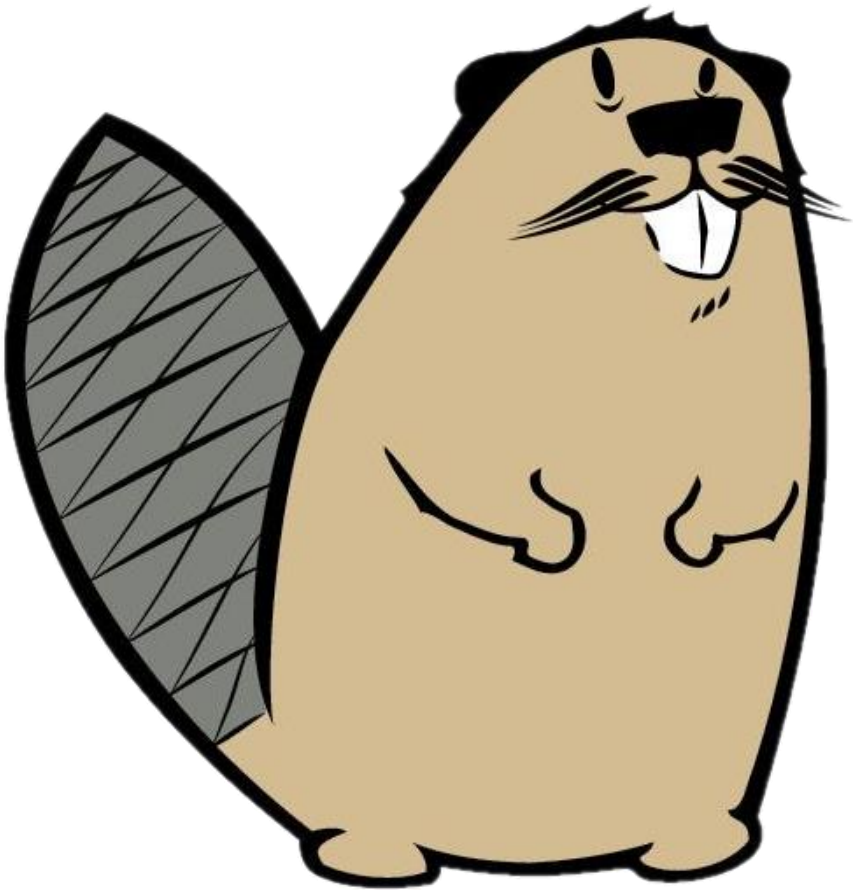
HÓDÍTSD MEG A BITEKET!

INFORMATIKAI GONDOLKODÁST TÁMOGATÓ, NEMZETKÖZI BEBRAS KEZDEMÉNYEZÉS MAGYAR MEGVALÓSULÁSA



ELTE | IK

Mi is az E-HÓD

Az e-HÓD/HÓDítsd meg a biteket a nemzetközi BEBRAS-kezdeményezés magyar partnere.

A nemzetközi Bebras, melyhez 2024-ben már több mint 70 ország kapcsolódott, 2015-ben elnyerte az Informatics Europe „Best Practices in Education” díját.

A kezdeményezés alapja Dr. Valentina Dagiene litván professzor által életre keltett verseny, melynek célja, hogy rövid, gyorsan (kb. 3 perc alatt) megérthető és megoldható feladatokkal megvalósítsa az alábbiakat:

- felkeltse az érdeklődést az informatika iránt;
- feloldja az informatikával kapcsolatos félelmeket, negatív érzéseket;
- megmutassa az informatika sokszínűségét, felhasználási lehetőségeit és területeit.

A kérdések három nehézségi szinten csak strukturált és logikus gondolkodást igényelnek, semmilyen különleges informatikai tudás nem szükséges a megválaszolásukhoz. A feladatok érdekes problémákat mutatnak be. Nem tesztek, inkább szórakoztató gondolkodtató feladványok.

Magyarországon 2024-ben tizennegyedik alkalommal, öt korcsoportban vehettek részt a diákok 4-től 12. osztályig.

A versenyt az ELTE IK és az NJSZT Közoktatási Szakosztálya szervezi.

Az alábbi dokumentumban a 2024-es magyar verseny feladatai és megoldásai találhatóak.

További információkért látogasson el a e-Hód | HÓDítsd meg a biteket! weboldalra, vagy írjon email-t az info@e-hod.elte.hu címre.

Részvétel

A részvétel mindenki számára ingyenes.

A verseny november második és harmadik hetében kerül lebonyolításra, osztályonként kiválasztható, hogy az adott héten melyik napon mikor oldják meg a feladatokat (8:00 és 16:00 között). Ezzel biztosítható, hogy akár egy tanóra keretein belül tudjanak részt venni egész osztályok.

A résztvevő diákoknak egy-egy internet kapcsolattal rendelkező számítógépre van szükségük. A feladatok megjelenítése és elküldése minden böngészőn működik. A verseny befejezése után, a hód hetet követően kerülnek nyilvánosságra a megoldások, melyek lehetőség szerint átbeszélhetők ugyancsak akár egy tanóra keretein belül.

Szabályok

- A verseny lebonyolítása iskolai helyszíneken történik.
- A résztvevők online kapják meg és válaszoljak meg a kérdéseket;
- A versenyre fordítandó idő 45 perc, 18 feladat három nehézségi szinten: könnyű, közepes és nehéz (legkisebb korosztályban 12 feladat);
- A verseny alatt semmilyen más számítógépes program, alkalmazás nem használható;
- A verseny során nyugalmas környezetet kell biztosítani;
- A terem a verseny során nem hagyható el;
- Az esetleges számítógéppel, internettel kapcsolatos észrevételeket a kontakt személynek kell összegyűjtenie és továbbítania a szervezők felé;
- A verseny célja: minél több pont összegyűjtése helyes válaszok megjelölésével, helytelen válaszok eseten pontlevonás történik;
- A kérdések tetszőleges sorrendben megválaszolhatók;
- A kérdések, problémák megértése a feladat részét képezi. Ezért a feladatok megbeszélése és értelmezéssel kapcsolatos kérdések nem megengedettek;
- A megoldások a verseny befejezése után, a hód hetet követően kerülnek nyilvánosságra.

Értékelés, Pontozás

A Kishód korcsoportban 15, minden más korcsoportban 18 feladatot kell megoldani három nehézségi szinten. Minden helyes válasz pontot ér, minden helytelen válaszért pontlevonás jár.

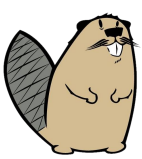
Nem megválaszolt kérdés esetében az összpontszám változatlan marad.

Az alábbi táblázat mutatja, hogy a feladatok nehézségétől függően hány pont kerül jóváírásra, illetve levonásra:

	Könnyű	Közepes	Nehéz
Helyes válasz	6 pont	9 pont	12 pont
Helytelen válasz	-2 pont	-3 pont	-4 pont

Összesen (18 feladat esetében) maximum 216 pont érhető el, mivel mindenki 54pontról indul.

Tartalom



Feladatok

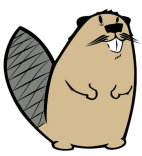
Kishód: 2024-BR-04, 2024-CH-03b, 2024-ID-04, 2024-IE-01c, 2024-KR-02
2024-FI-01, 2024-KR-03a, 2024-NL-02, 2024-PL-03, 2024-SK-01b
2024-DE-02, 2024-DE-03, 2024-MT-02, 2024-MY-03, 2024-PL-04

Benjamin: 2024-CZ-05, 2024-FI-01, 2024-KR-03a, 2024-NL-02, 2024-PL-03, 2024-SK-01b
2024-CH-03a, 2024-DE-02, 2024-DE-03, 2024-MT-02, 2024-MY-03, 2024-PL-04
2024-DE-04a, 2024-DE-07, 2024-DE-08, 2024-HU-02, 2024-IN-04, 2024-SI-01

Kadét: 2024-CH-03a, 2024-DE-02, 2024-DE-03, 2024-MT-02, 2024-MY-03, 2024-PL-04
2024-DE-04a, 2024-DE-07, 2024-DE-08, 2024-HU-02, 2024-IN-04, 2024-SI-01
2024-AT-03, 2024-BE-01a, 2024-CA-02, 2024-DE-06a, 2024-HU-03, 2024-TW-03

Junior: 2024-DE-04a, 2024-DE-07, 2024-DE-08, 2024-HU-02, 2024-IN-04, 2024-SI-01
2024-AT-03, 2024-BE-01a, 2024-CA-02, 2024-DE-06a, 2024-HU-03, 2024-TW-03
2024-AU-03, 2024-BG-01b, 2024-IT-01b, 2024-LT-05, 2024-PK-03, 2024-SK-04

Senior: 2024-AT-03, 2024-CA-02, 2024-DE-06a, 2024-HU-03, 2024-SI-01, 2024-TW-03
2024-BG-01b, 2024-DE-04b, 2024-IT-01b, 2024-LT-05, 2024-PK-03, 2024-SK-04
2019-VN-04, 2024-AU-01, 2024-BE-01b, 2024-DE-05, 2024-FI-03, 2024-HU-04

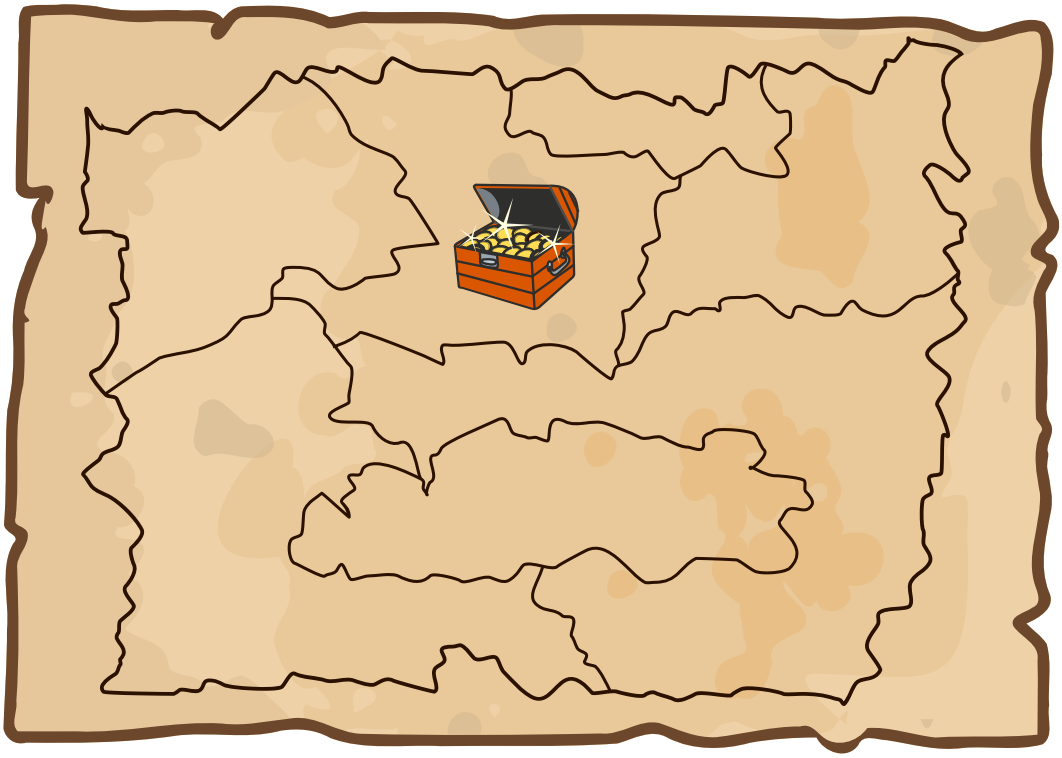


Hódkirály kincse (2019-VN-04)

SENIOR – NEHÉZ

A Hódkirály hét tartomány felett uralkodik, amelyek határait az alábbi térkép mutatja.

A kincset, e tartományok egyikében rejtette el.

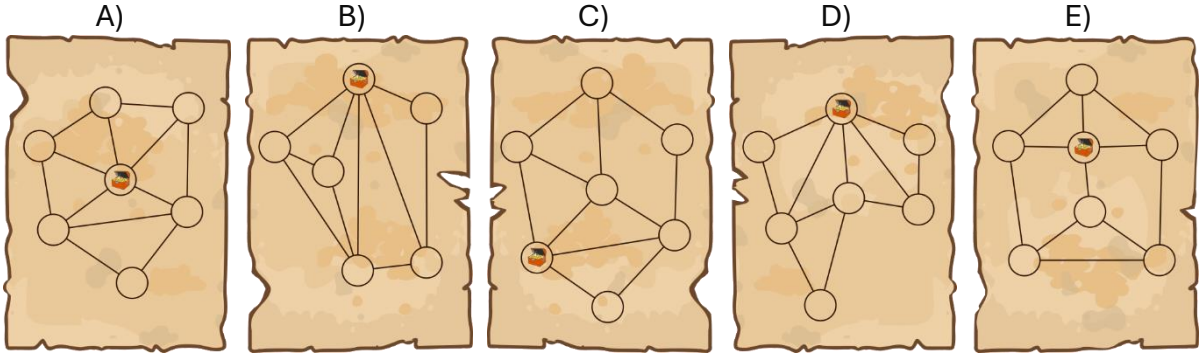


A király készíttet egy kincses térképet, amelyen a tartományok körökként vannak ábrázolva. Megjelölte azt a tartományt, ahol a kincs van. Két kör mindig akkor kapcsolódik össze, ha a megfelelő tartományoknak közös a határa.

Hogy a rablók ne tudják ellopni a kincset, a király négy hamis kincses térképet is készíttetett.

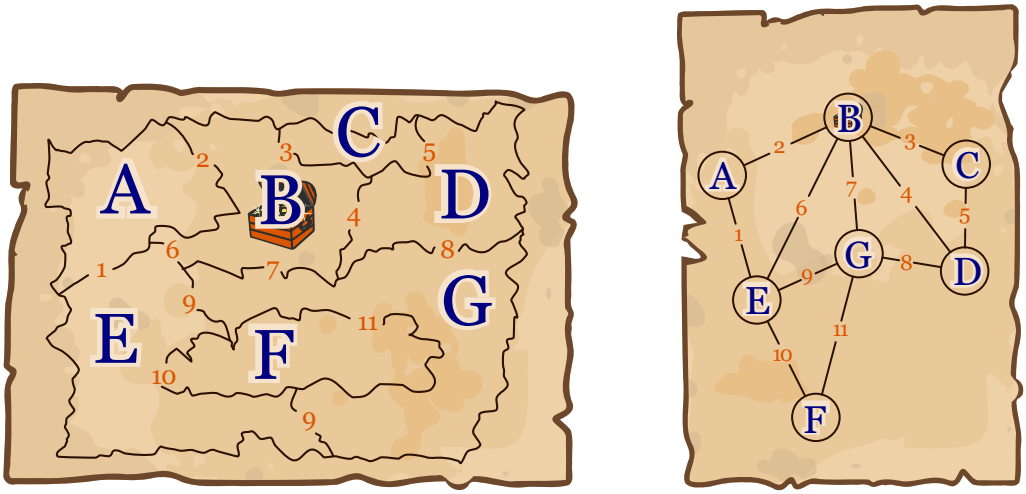
Melyik a helyes kincses térkép?

A) B) C) D) E)



A helyes válasz a D)

A következő nézetben az egyes területeket az A, B, C, D, E, F és G betűkkel jelöltük meg. Az egyes határterületeket pedig 1-től 11-ig beszámoztuk. Mivel az E és a G terület esetében két külön határ is van, de a meghatározás arra vonatkozott, hogy van közös határuk, ezért ugyanúgy 9-cel jelöltük mindkettőt. A jelölések a segítségével gyorsan megbizonyosodhatunk róla, hogy melyik a helyes kincses térkép. Az A) válasz hamis: az A, C és F három tartomány mindegyikének csak két szomszédja van. Ezért a kincses térképen három olyan körnek kellene szerepelnie, melyekből csak két vonal indul ki (két másik tartománnyal van összekötve). De itt csak egy olyan kör szerepel. A B) válasz sem lehet helyes, mivel csak hat kör szerepel, de hét tartományunk van. A C) válasz sem helyes: a kincset tartalmazó tartománynak öt határos szomszédja van (A, C, D, E és G), tehát öt vonalnak kell kiindulnia a kinccsel jelölt körből. De ezen a térképen csak 4 kiindulás szerepel.



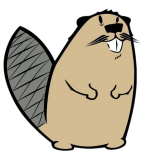
Az A) válasz hamis: az A, C és F három tartomány mindegyikének csak két szomszédja van. Ezért a kincses térképen három olyan körnek kellene szerepelnie, melyekből csak két vonal indul ki (két másik tartománnyal van összekötve). De itt csak egy olyan kör szerepel. A B) válasz sem lehet helyes, mivel csak hat kör szerepel, de hét tartományunk van. A C) válasz sem helyes: a kincset tartalmazó tartománynak öt határos szomszédja van (A, C, D, E és G), tehát öt vonalnak kell kiindulnia a kinccsel jelölt körből. De ezen a térképen csak 4 kiindulás szerepel.

MIÉRT INFORMATIKA

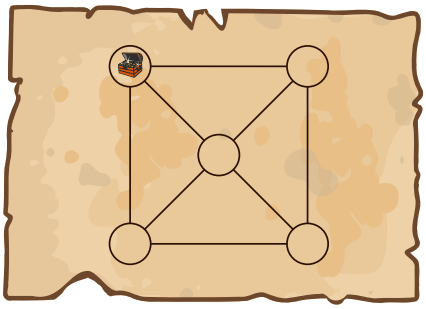
Ebben a feladatban egy térképet gráf segítségével reprezentálunk. Egy ilyen gráf a térkép absztrakcióját, leképezését jeleníti meg.

Mint minden absztrakciónál, így itt is, a lényegtelen információk elvesznek: pl. az egyes tartományok földrajzi elhelyezkedése. Habár a B tartomány a D tartománnyal azonos vonalban (magasságban) helyezkedik el, a gráfban különböző szinteken szerepelnek.

A fontos információk, mint a határok, továbbra is elérhetőek lesznek – az absztrakt modell tartalmazni fogja azokat. De egy helyes absztrakciós modellben is benne van az a



veszély, hogy egy látszólag lényegtelen információ elvész: a következő gráf (melyet egy másik államról készítettünk), forgásszimmetrikus, tehát nem világos, hol lehet elrejtve a kincs.



Egy gráf csomópontokból (ebben a feladatban a körök) és élekből (ebben a feladatban a vonalak) áll. Ebben a feladatban a gráf élei arra szorítkoznak, hogy két csomópontot kössenek össze. Lehetőségünk lenne arra is, hogy két csomópontot több éllel kössünk össze, így megjeleníthetnénk az E és G tartományok közötti mindkét határszakaszt.

Az informatikában a gráfokat rendszeresen használjuk információk elvonatkoztatására. Sok esetben maga az absztrakció (elvonatkoztatás) adja a probléma megoldását.

WEBOLDALAK

Gráf – Wikipédia

Absztrakció – Wikipédia

KULCSSZAVAK

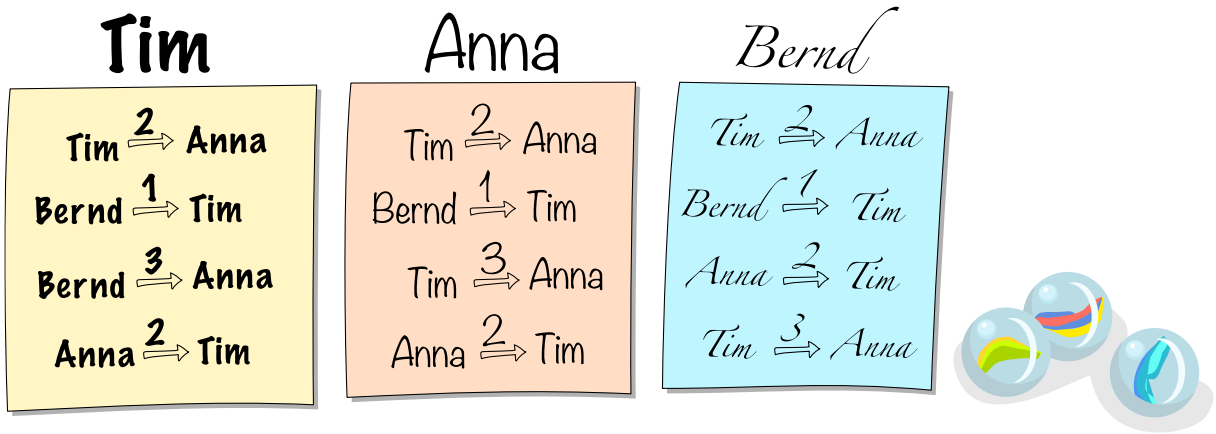
Gráf, térkép, reprezentáció

Elosztott lista (2024-AT-03)

KADÉT – NEHÉZ
JUNIOR – KÖZEPES
SENIOR – KÖNNYŰ

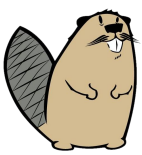
Anna, Bernd és Tim időnként kölcsönadják egymásnak az üveggolyóikat. Hogy biztosan ne hibázzanak, mindenki felírja, hogy ki kinek hány üveggolyót adott kölcsön.

Egy hét múlva összehasonlítják a feljegyzéseiket.



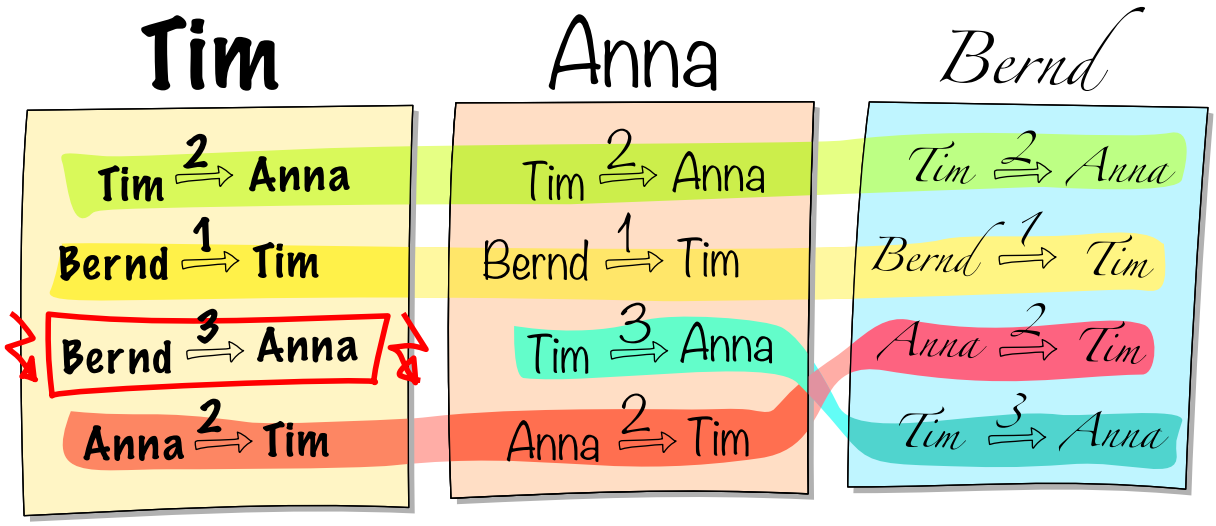
A három barát azt gyanítja, hogy hiba történt. Ha igen, ki követte el a hibát?

A) Anna
B) Bernd
C) Tim
D) Senki sem hibázott.



A helyes válasz a C), Tim.

Ha senki sem hibázott volna, minden papírra ugyanazok a bejegyzések kerültek volna. Három bejegyzés minden papíron ugyanaz. Ezeket a képen zöld, sárga és narancssárga vonal jelöli.



Csupán egy különbség van: a

$$\text{Tim} \overset{3}{\Rightarrow} \text{Anna}$$

bejegyzés 2 papíron (Annáén és Berndén) is megtalálható. Tim azonban az alábbit

$$\text{Bernd} \overset{3}{\Rightarrow} \text{Anna}$$

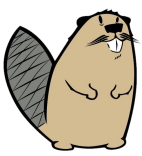
jegyezte fel.

Mivel azt feltételeztük, hogy csak egyvalaki hibázott, ez így Tim volt.

MIÉRT INFORMATIKA?

Az informatika nyelvén azt mondanánk, hogy a történetünkben szereplő gyerekek úgy döntöttek, hogy nem egy közös adatbázisban kezelik adataikat (a golyók kölcsönzésének bejegyzéseit), hanem saját adatbázist vezetnek. Ennek eredményeként most több különálló adatbázis van, amelyeknek mégis ugyanazt a tartalmat kell megjeleníteniük.

De miért ez a látszólag felesleges többszörös adattárolás? Ennek akkor van értelme, ha nélkülözni akarjuk a központi rendszert, például azért, mert az meghibásodhat, vagy nem megbízhatóan működhet. Összességében itt egy peer-to-peer (P2P, magyarul kb. egyenrangú) hálózat példáját látjuk: minden gyereknek (peer-nek) ugyanazok a jogai és kötelezettségei (nevezetesen a kölcsönzési adatok helyes tárolása). A gyerekek kommunikálhatnak egymással az adatok szinkronizálása érdekében, ahogyan a számítógépek is ké-



pesek erre egy hálózatban. Egy egyenrangú hálózatban a szinkronizálás csak akkor működik, ha a legtöbb egyenrangú fél becsületes és hiba nélkül dolgozik. Nézeteltérés esetén „szavaznak” arról, hogy melyik adatbázis a helyes. A feladatban Anna és Bernd nyerné ezt a szavazást.

A blokk-lánc technológia alapgondolatai közé tartozik az egyenrangúság és az adatok megosztása. A feladatban említett megközelítés csak akkor működik, ha a legtöbb résztvevő becsületes, hiba nélkül dolgozik. Ha nézeteltérések vannak, akkor a társak szavaznak arról, hogy melyik a helyes.

A blokklánc-technológia egy lépéssel tovább megy egy úgynevezett konszenzus algoritmus megvalósításával, amikor például egy rosszindulatú, vagy hibázó szereplőnek több számítási teljesítményt kellene adnia, mint az összes többi résztvevőnek együttvéve.

Ez de facto lehetetlen, ha a hálózat elég nagy. Ez a feladat azonban már megmutatja, hogyan lehet a bizalmat erősíteni egy nem eleve megbízható környezetben a hatalomnak az összes résztvevő közötti elosztásával.

WEBOLDAL

Adatbázis – Wikipédia

Peer-to-peer – Wikipédia

Blokklánc – Wikipédia

KULCSSZAVAK

Adatbázis, Blokklánc, Peer-to-peer hálózat, Hálózat

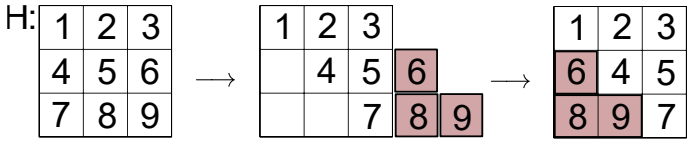
Képtitkosítás (2024-AU-01)

SENIOR – NEHÉZ

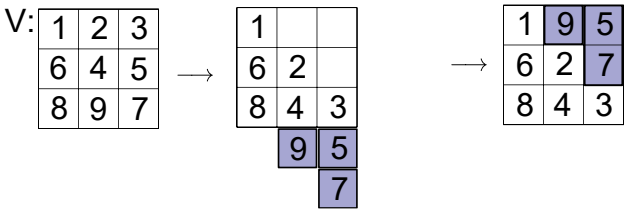
Leo új módszert talált ki a képek titkosítására a H (horizontális, vízszintes) és V (vertikális, függőleges) művelettel. Egy kép mindig egy téglalap, amely színes pontok soraiból és oszlopaiból, azaz pixelekből áll.

A ***H művelet*** hatására az első sor minden egyes képpontja ugyanabban a pozícióban marad, a második sor minden egyes képpontja egy oszlopot tolódik jobbra, a harmadik sor minden egyes képpontja két oszlopot tolódik jobbra és így tovább. Az n-edik sor minden egyes képpontja tehát n-1 oszloppal jobbra tolódik.

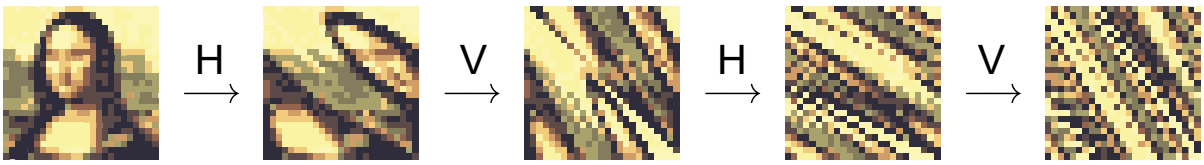
A kép jobb szélén túlra áthelyezett pixelek a kép bal szélén ugyanabba a sorba kerülnek vissza, anélkül, hogy sorrendjük megváltozna. Íme a H vázlatos ábrázolása egy 3x3-as képen, amelynek pixeleit 1-től 9-ig számoztuk:



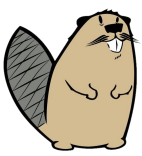
A **V művelet** hasonló: az n-edik oszlop minden egyes képpontja n-1 sorral mozog lefelé, és a kép alsó szélén túlra került képpontokat újra beillesztjük ugyanabba az oszlopba, a kép tetejére a sorrend megváltoztatása nélkül. Íme a V művelet sematikus ábrázolása:

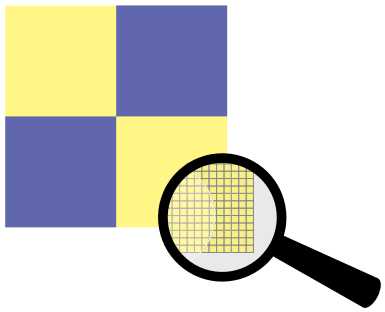


A következő példában a Mona Lisa 25x25-ös képét titkosítjuk a H V H V művelettekkel egymás után:

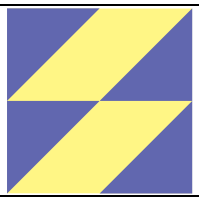
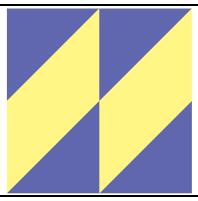
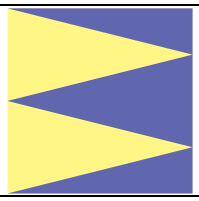
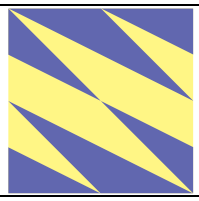


Leo a következő 1000 x 1000 pixelből álló képet a V H műveleteket egymás után alkalmazva titkosította.





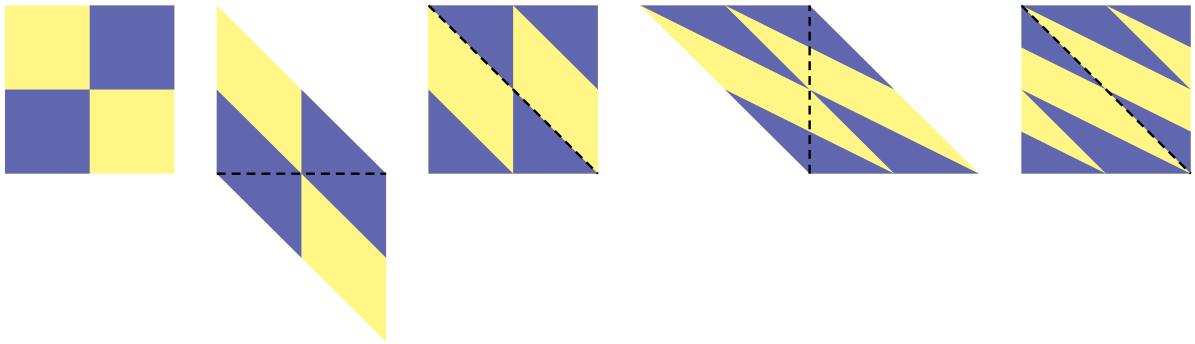
Melyik a titkosított kép?

A	B	C	D	E
				

Megoldás/ A helyes válasz az E.

A példában a Mona Lisa képének torzulása azt jelzi, hogy a H művelet geometriailag vízszintes nyírást, a V művelet pedig függőleges nyírást jelent.

A négy színes négyzetet tartalmazó képet a V művelet első lépése először lefelé nyírja. Az eredmény egy torzított paralelogramma. Az alsó szélénél lévő túlnyúló háromszöget beillesztjük a felső részbe (párhuzamos eltolással), így ismét egy téglalap alakú kép jön létre. Ezután a H művelet segítségével hasonló jobbra nyírást hajtunk végre. A túlnyúlás balra kerül beillesztésre:



MIÉRT INFORMATIKA?

Az információ továbbításakor fontos lehet az adatok titkosítása, hogy az információt elfogó személy ne tudja könnyen azonosítani az eredeti adatokat. A képek titkosítási módszere, amely a H és V műveleteken alapul, azzal az előnnyel jár, hogy egyszerűen és könnyen használható. Ha a címzett ismeri a műveletek sorozatát (pl. H V), könnyen vissza tudja fejteni. Azonban a titkosított képet szemmel már nem lehet felismerni.

Minél magasabb egy titkosítási módszerrel szemben támasztott biztonsági követelmény, annál nagyobb erőfeszítést igényel általában a titkosítás és a visszafejtés (pl. a titkosított információ feladója és címzettje közötti összetett titkos kulcs továbbítása). Ez az eljárás könnyen elvégezhető, de könnyen „feltörhető” is lehet, ha ismerjük a kulcsot.

A titkosítás különlegessége, hogy geometriai leképezéseken (nyírás és eltolás) alapul.

WEBOLDAL

Titkosítás – Wikipédia

A kriptográfia története – Wikipédia

KULCSSZAVAK

Titkosítás, Geometriai transzformáció, Képek titkosítása

Rajzoló robot (2024-AU-03)

JUNIOR – NEHÉZ

Ati rajzoló robotja két alapvető utasítást ismer:

`E` mellyel egy egységet előre mozog
`J` mellyel 90 fokot fordul jobbra

Ezen kívül, minden nem használt betű hozzárendelhető rövidítésként több utasítás sorozatához. Például:

`A = EEE` a 3 egységgel való előrehaladás rövidítése.

`B = AJA` az `A` (azaz `EEE`), majd a jobbra fordulás, majd az `A` (azaz `EEE`) ismét.

Egy programban tetszőleges számú sor lehet a különböző rövidítések meghatározására. Az utolsó sor mindig a „`Rajzolj`” utasítással kezdődik, amely a végső rajzot készíti el.

Ati a következő rajzot szeretné megrajzoltatni, ahol minden rövid vonalszakasz 1 egység hosszú.

Ehhez a következő programot készítette:

1. sor: `W=EJ`
2. sor: `X=WJJ`
3. sor: `Y=WXWX`
4. sor: `Rajzolj YYYY`

Sajnos azonban az egyik sorban elrontott pontosan egy valamit, ami miatt nem azt rajzolja le a robot, amit Ati szeretne.

Melyik sorban rontott el pontosan egy valamit?

A) 1. sorban
B) 2. sorban
C) 3. sorban
D) 4. sorban

Megoldás/ A helyes válasz a C: a 3. sorban.

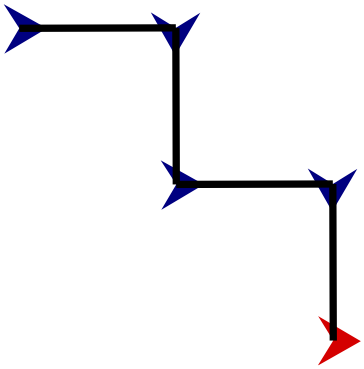
Tegyük fel, hogy a robot jobbra néz. Menjünk végig az egyes kódsorokon:

Az 1. sorban (`W=EJ`) leírtak szerint a robot egy egységet megy előre, majd jobbra fordul

A 2. sorban (`X=WJJ`) végrehajtja a W-t, azaz előre megy egy egységet és jobbra fordul; majd kétszer ismét jobbra fordul:

Ezzel tulajdonképpen előre ment és balra fordult 90 fokot.

A 3. sorban (`Y=WXWX`) a két előző sorban meghatározott betűket használja fel felváltva: azaz előre megy, jobbra fordul, előre megy, balra fordul…



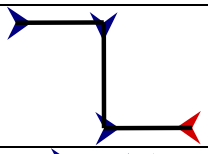
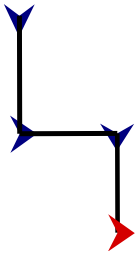
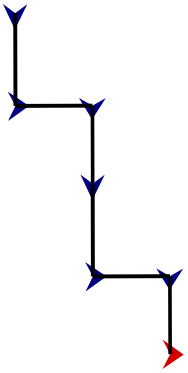
A 4. sorban (`Rajzolj YYYY`) ezt ismételné meg négyszer egymás után. Ezzel azonban egy lépcsőt rajzol a kereszt helyett.

Vizsgáljuk meg alaposabban a megrajzolandó képet: ezen egy 12 egységnyi részből álló vonal látható és a 4. sorban ugyanazt ismételjük meg négyszer, így egyértelmű, hogy az Y utasítás a rajz egynegyedét kell megrajzolja. Ha a 4. sor lenne a hibás, a hiba azt jelentené, hogy nem négyszer, hanem csak háromszor kellene megismételni a 3. sorral rajzoltakat. De a minta ettől még a lépcső-forma maradna. Emiatt nem a 4. sorban kell keresnünk a hibát.

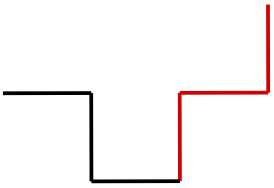
A négy ismétlés azonban azt jelenti, hogy a rajz negyedéd, azaz 3 szakaszt kell megrajzolnunk egy lépésben az `Y` utasítással. Emiatt biztosan itt kell a hibát keresnünk, és egy felesleges „szakaszt” eltávolítani.

Ezt kétféleképpen tehetjük meg:

1. Az X meghatározásából kellene kitörölnünk a W-t. Mivel a W-ből nem távolíthajuk el az előre lépést (hiszen akkor egyáltalán nem haladnánk, csak forognánk), az X-ből vesszük ki a W-t (előre lépést is tartalmazó utasítást). Ez azonban a 3. sorban kétszer szerepel, emiatt nem egy, hanem két szakasszal rajzolnánk kevesebbet minden lépés során. Emiatt ez nem jó megoldás.
2. A 3. sorban kell változtatnunk. De mit?
   Azt már beláttuk, hogy a 12 szakasz miatt egy negyed rajzon csak 3 szakasznak kellene szerepelnie – emiatt nem az X-et kell W-re vagy a W-t X-re cserélnünk (maradna a 4 szakasz). E-re nem cserélhetjük le egyiket sem, mivel akkor a következő utasítás (W vagy X) miatt megint előre mennénk, és így két hosszú szakasz keletkezne, ami nem lehetséges (nem szerepel a képen).
   Ha valamelyik betűt J-re cserélnénk, akkor

<table>
<tr><td>(Y=WXWJ)</td><td rowspan="3">  </td><td rowspan="3">A J utáni utasításban egy szakaszt újra rajzolunk, így biztosan kevés lesz a szakaszok száma…</td></tr>
<tr><td>(Y=WXJX)</td></tr>
<tr><td>(Y=WJWX)</td></tr>
<tr><td>(Y=JXWX)</td><td>  </td><td>Lefele „nézve” kezdenénk megrajzolni a képet.
 
A következő Y-nál azonban megint jobbra fordulással kezdenénk, ami miatt
 
a folytatás nem közelítene ahhoz, amit szeretnénk.</td></tr>
</table>

Marad az a lehetőség, hogy a 3. sorban eggyel több betűt gépeltünk. Ha elképzeljük a „negyedeket” egymás után, már a másodiknál jól megmutatkozik, hogy egy előre-jobbra (`W`), majd egy előre-balra (`X`) utasítás után ismét egy előre-balra (`X`) a megoldás.



Azaz a 3. sor helyesen: `Y=WXX`

Az `XWW` is helyes sorrend lehetne, de ahhoz nem tudunk egy törléssel, módosítással eljutni.

MIÉRT INFORMATIKA?

Ebben a hódfeladatban egy programozási nyelvet ismerhetsz meg. Ez a nyelv egymás után hajtja végre az utasításokat, de azt is lehetővé teszi, hogy egy utasításkészlethez *eljárásokat, függvényeket* definiálj. Az eljárások vagy függvények használata egy programban sokszor nem csak azt teszi lehetővé, hogy ugyanúgy hajtsd végre az utasításokat. Legtöbbször paraméterek alkalmazásával kisebb változásokat is megenged – pl. mennyit menjünk előre, mennyit forduljunk. Az eljárások, függvények használata olvashatóbbá is teheti a programodat. Ebben az esetben az eljárások elnevezési konvenciója valójában megnehezíti a program olvashatóságát.

A feladat a programozók másik fontos feladatát is megmutatják: a *hibakeresést és annak kijavítását*. A hibakeresés első lépése az, hogy leteszteljük – azaz megnézzük, hogy a várt eredményt adja-e. Ha nem, igyekszünk lépésenként visszakövetni, hol „csúszhatott” félre a program. A programozók sokféle tesztelési, hibakeresési módszert alkalmaznak. Itt megmutatkozott, hogy ugyan tudtuk, hogy csak egy helyen lehet hiba, de az nem volt egyértelmű, hogy másik betűt (utasítást) használtunk, vagy eggyel többet/kevesebbet.

WEBOLDAL

Debuggolás - Mi az? - Prog.Hu

Függvény (programozás) – Wikipédia

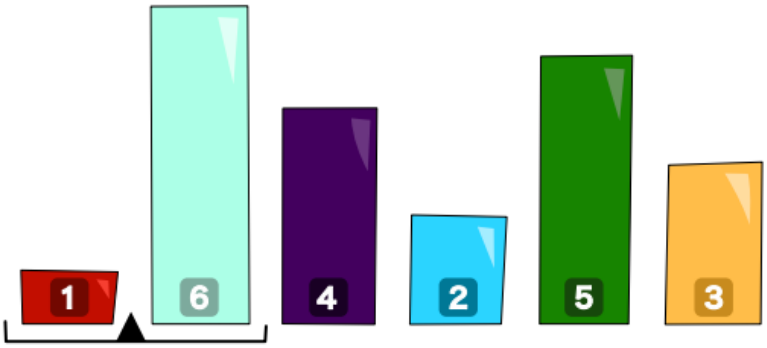
KULCSSZAVAK

Függvény, Eljárás, Hibakeresés

Építőkocka mozgató (2024-BE-01a)

NEHÉZ – KADÉT
KÖZEPES – JUNIOR

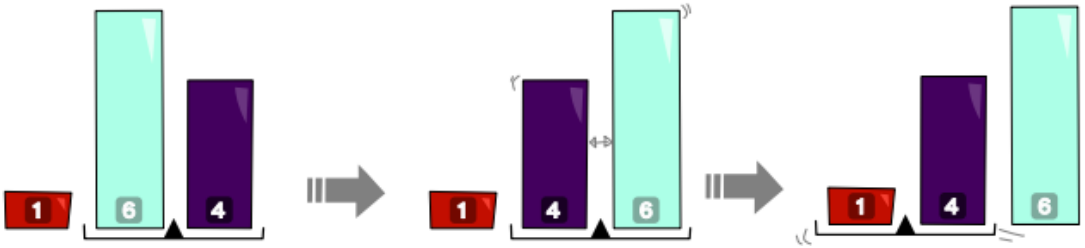
Egy gép hat különböző magasságú építőkockát tud mozgatni. Ehhez egy jelölőt használ, ami mindig két építőkocka között áll. Kezdetben így:



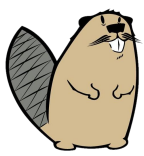
A gép ezután az alábbi szabályok szerint működik:

- Ha a jelölőtől balra lévő építőkocka alacsonyabb, mint a tőle jobbra lévő, akkor a jelölő jobbra mozog.

- Ha a jelölőtől balra lévő építőkocka magasabb, mint a tőle jobbra lévő, akkor az építőkockákat kicseréli. Ezután pedig balra mozog, de csak akkor, ha továbbra is két építőkocka között marad.

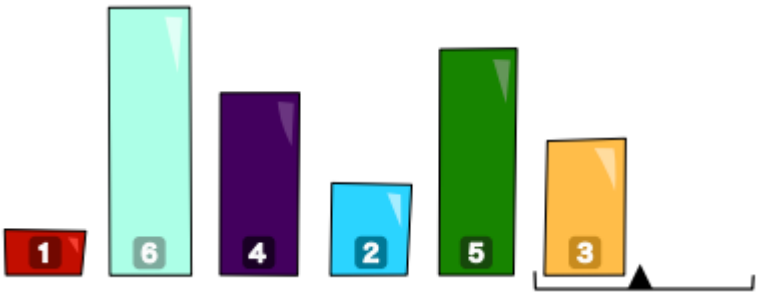


A gép leáll, amikor a jelölő egészen jobbra kerül, és már nem áll két építőkocka között.

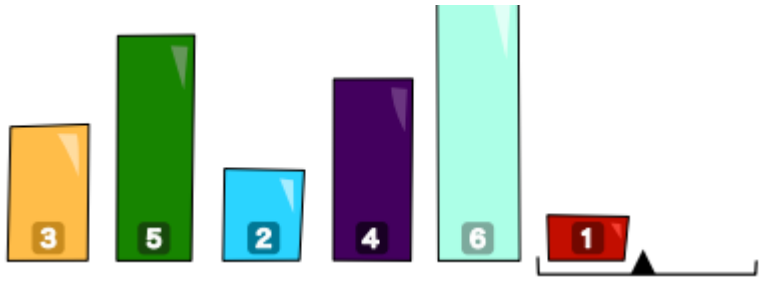


Hogy állnak az építőkockák, amikor a gép leáll?

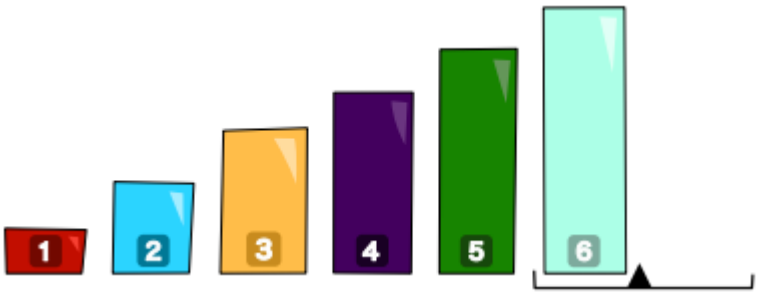
A)



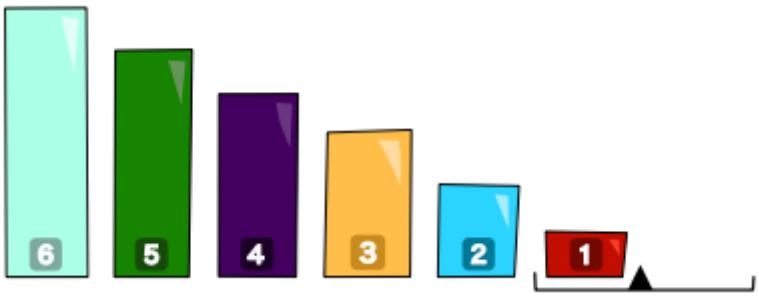
B)

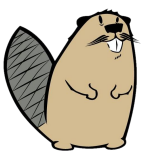


C)

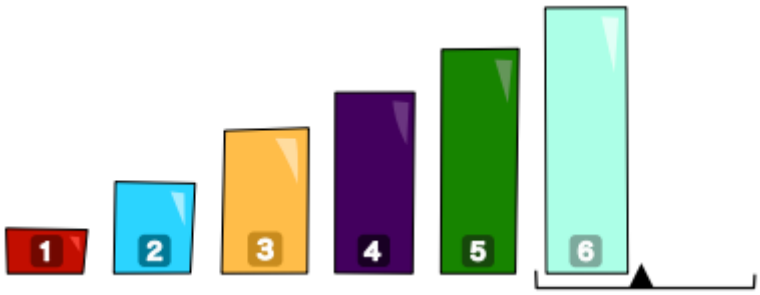


D)





Megoldás/ A helyes válasz a C.



Alapötlet: A jelölőtől balra lévő építőkockák mindig rendezettek. Kezdetben ez egyértelműen látszik, a két bemutatott lépés esetében. Ezután már csak azt kell bizonyítanunk, hogy a gép megáll, azaz „kifut” jobbra: Egy balra visszalépés után (mindegy, hogy hol állunk éppen), a jelölő mindig visszamegy jobbra, legalább egy építőkockával (az előző – balra lépés előtt végzett – csere miatt).

Természetesen lépésről lépésre is követhetnéd a gép munkáját, de egyszerűbb felismerni, hogy a gép magasság szerint válogatja az építőkockákat. A jelölő csak akkor mozdul jobbra, ha a jelölőtől balra lévő építőkocka alacsonyabb, mint a jobbra lévő. Ha egy alacsonyabb építőkockát kicserélünk a bal oldali szomszédjával, akkor a jelölő is balra mozog. Ez egyértelművé teszi, hogy a jelölőtől balra lévő összes építőkocka már magasság szerint növekvő sorrendbe van rendezve.

Egy alacsonyabb építőkocka addig cserélődik tovább balra, amíg magasabb nem lesz, mint a tőle balra lévő építőkocka (és így az összes tőle balra lévő építőkockáka). Ekkor a jelölő ismét jobbra kezd mozogni egészen addig, amíg egy újabb alacsonyabb építőkocka nem jelenik meg a sorban. Ily módon az építőkockák lépésről lépésre magasságuk szerint növekvő sorrendbe kerülnek.

Miután az összes építőkockát magasság szerint növekvő sorrendbe rendeztük, a jelölő fokozatosan jobbra mozog, amíg az építőkockák jobb oldalára nem kerül, és a gép megáll.

A C válasz mutatja ezt a magasság szerint rendezett állapotot.

MIÉRT INFORMATIKA?

A számítógépek adatokat dolgoznak fel - gyakran sok adatot. Fontos, hogy az adatok rendszerezve legyenek. Az informatikában a rendszerezés helyett általában a rendezés kifejezést használják: Például a számokat növekvő méret szerint, az emberekre vonatkozó adatokat születési dátum szerint, vagy a könyvekre vonatkozó adatokat ISBN-szám szerint lehet rendezni. A rendezési folyamat végén az adatok a kívánt sorrendben vannak. Mivel a számítógépeknek folyamatosan nagy mennyiségű adatot kell rendezniük, az informatikusok többek között azzal foglalkoznak, hogy minél **hatékonyabb rendezési módszereket** találjanak, amelyek kevés időt és kevés tárhelyet igényelnek.

Az ebben a hód feladatban használt rendezést „**Gnomesort**”-nak, azaz kertitörpe rendezésnek nevezik. Ez hasonló a korábbi években szerepeltetett buborékos rendezéshez.

Ugyan könnyen érthető és leprogramozható, de nem különösebben hatékony, azaz bizonyos helyzetekben sokat mozog és cserél feleslegesen. Ha azonban az adatok már megközelítőleg rendezettek, mindkét módszer nagyon gyors - és talán gyorsabb, mint az olyan rendezési módszerek, mint a quicksort vagy a mergesort, amelyek általában sokkal hatékonyabbak.

WEBOLDAL

Kertitörpe-rendezés – Wikipédia

Gnome Sort Algorithm Animation - algostructure.com

Rendezés (programozás) – Wikipédia

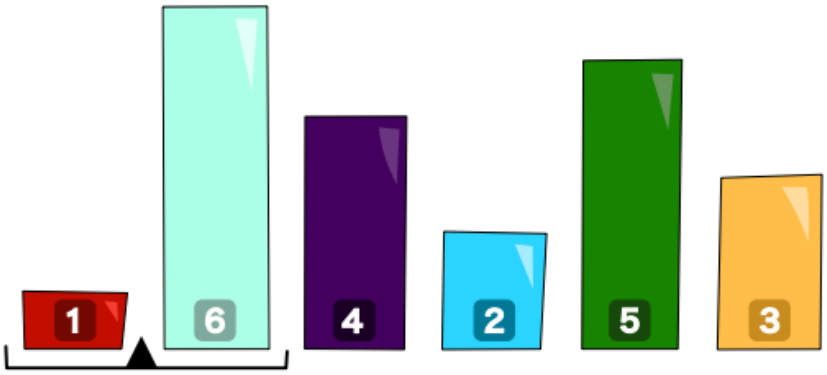
KULCSSZAVAK

Rendezés, Kertitörpe-rendezés

Építőkocka mozgató (2024-BE-01b)

NEHÉZ – SENIOR

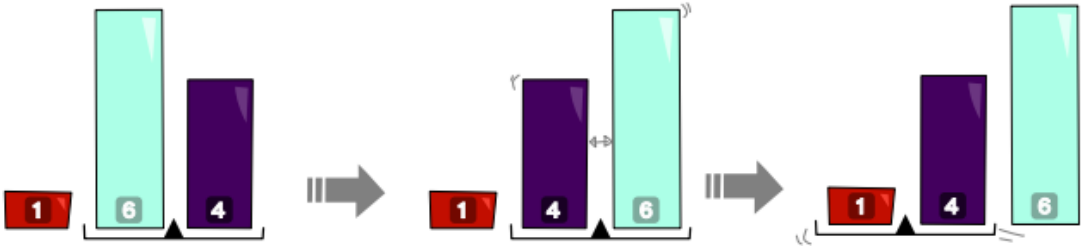
Egy gép hat különböző magasságú építőkockát tud mozgatni. Ehhez egy jelölőt használ, ami mindig két építőkocka között áll. Kezdetben így:



A gép ezután az alábbi szabályok szerint működik:

- Ha a jelölőtől balra lévő építőkocka alacsonyabb, mint a tőle jobbra lévő, akkor a jelölő jobbra mozog.

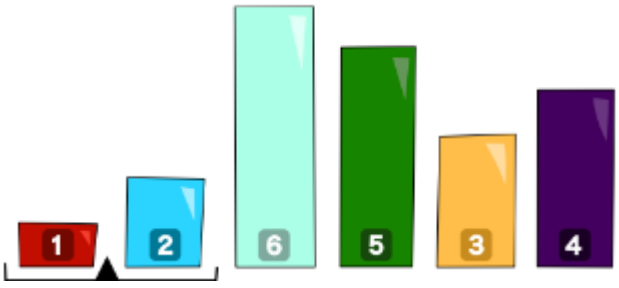
- Ha a jelölőtől balra lévő építőkocka magasabb, mint a tőle jobbra lévő, akkor az építőkockákat kicseréli. Ezután pedig balra mozog, de csak akkor, ha továbbra is két építőkocka között marad.

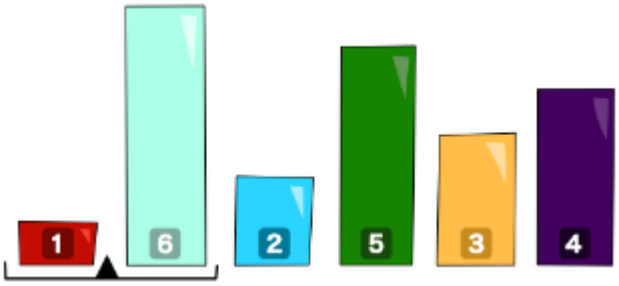


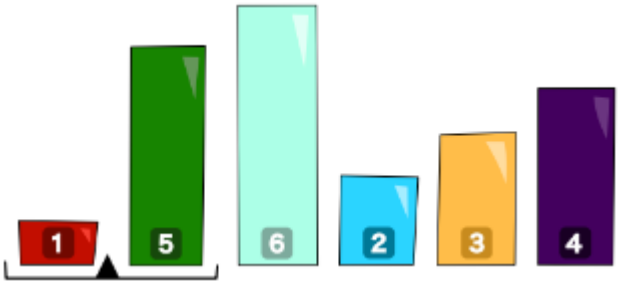
A gép leáll, amikor a jelölő egészen jobbra kerül, és már nem áll két építőkocka között.

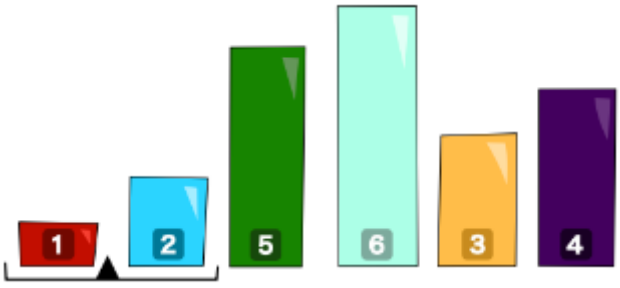
Attól függően, hogy az építőkockák kezdetben milyen sorrendben állnak, a gép más-más lépésszám (jelölőmozgatás) után áll le.

Az alábbi építőkocka sorrendek melyikénél mozog a jelölő a legkevesebbszer?

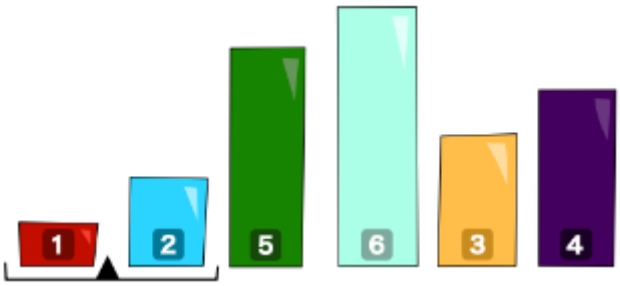
A) 

B) 

C) 

D) 

Megoldás/ A helyes válasz a D.



A helyes válasz meghatározásához szimulálhatjuk a gép munkáját minden egyes lehetséges megoldás esetén és megszámolhatjuk, hogy a jelölő hányszor mozog jobbra vagy balra.

A négy kezdő sorrendet azonban könnyebb összehasonlítani a mozgások számának szempontjából anélkül, hogy megszámolnánk őket. Az egyszerűség kedvéért a sorrendeket csak számokkal (az építőkockák magasságával) ábrázoljuk.

A	B	C	D
1 ▲ 2 6 5 3 4	1 ▲ 5 6 2 3 4	1 ▲ 6 2 5 3 4	1 ▲ 2 5 6 3 4

Ha a B válaszban szereplő sorrendből indulunk, két lépés után elérjük az A válaszban szereplő sorrendet:

1 ▲ 6 2 5 3 4 → 1 6 ▲ 2 5 3 4 → 1 ▲ 2 6 5 3 4

Tehát a B válaszban több lépésre van szükség, mint az A válaszban. Emiatt a B válasz kizárható.

Ha az A válaszban szereplő sorrendet kezdjük el feldolgozni, az első 3 lépés a következő:

1 ▲ 2 6 5 3 4 → 1 2 ▲ 6 5 3 4 → 1 2 6 ▲ 5 3 4 → 1 2 ▲ 5 6 3 4

De ezt érjük el a D válaszban szereplő sorrend feldolgozásánál egy lépésből.

1 ▲ 2 5 6 3 4 → 1 2 ▲ 5 6 3 4.

Azaz az A válasz sem a legkevesebb lépést tartalmazza.

Ha a C választ vizsgáljuk meg, ott négy lépés alatt jutunk el a D válaszban szereplő sorrendig:

1 ▲ 5 6 2 3 4 → 1 5 ▲ 6 2 3 4 → 1 5 6 ▲ 2 3 4 → 1 5 ▲ 2 6 3 4 → 1 ▲ 2 5 6 3 4

Tehát a C válasz sem a legkevesebb lépést igényli.

MIÉRT INFORMATIKA?

A számítógépek adatokat dolgoznak fel - gyakran sok adatot. Fontos, hogy az adatok rendszerezve legyenek. Az informatikában a rendszerezés helyett általában a rendezés kifejezést használják: Például a számokat növekvő méret szerint, az emberekre vonatkozó adatokat születési dátum szerint, vagy a könyvekre vonatkozó adatokat ISBN-szám szerint lehet rendezni. A rendezési folyamat végén az adatok a kívánt sorrendben vannak. Mivel a számítógépeknek folyamatosan nagy mennyiségű adatot kell rendezniük, az informatikusok többek között azzal foglalkoznak, hogy minél **hatékonyabb rendezési módszereket** találjanak, amelyek kevés időt és kevés tárhelyet igényelnek.

Az ebben a hód feladatban használt rendezést „**Gnomesort**”-nak, azaz kertitörpe rendezésnek nevezik. Ez hasonló a korábbi években szerepeltetett buborékos rendezéshez. Ugyan könnyen érthető és leprogramozható, de nem különösebben hatékony, azaz bizonyos helyzetekben sokat mozog és cserél feleslegesen. Ha azonban az adatok már megközelítőleg rendezettek, mindkét módszer nagyon gyors - és talán gyorsabb, mint az olyan rendezési módszerek, mint a quicksort vagy a mergesort, amelyek általában sokkal hatékonyabbak.

WEBOLDAL

Kertitörpe-rendezés – Wikipédia

Gnome Sort Algorithm Animation - algostructure.com

Rendezés (programozás) – Wikipédia

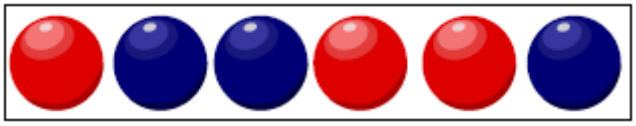
KULCSSZAVAK

Rendezés, Kertitörpe-rendezés

Labdák (2024-BG-01b)

NEHÉZ – JUNIOR
KÖZEPES – SENIOR

Adott egy kék és piros labdákból álló sorozat:



Minden labdánál megszámoljuk, hogy hány kék labda áll tőle jobbra (beleértve az adott labdát is):

```
3, 3, 2, 1, 1, 1
```

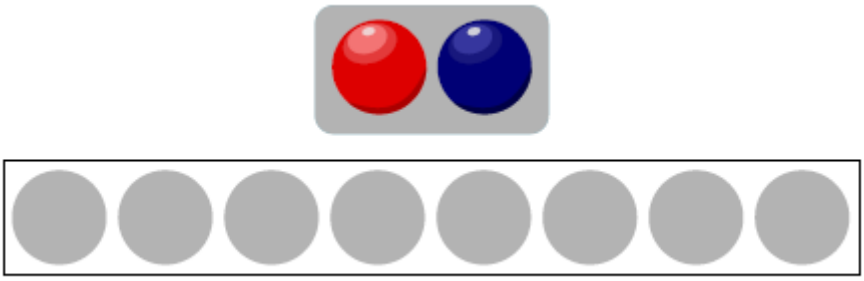
A következő lépésben ezt még egyszer leírjuk, de ha a szám páros volt, akkor 0-t, ha páratlan, akkor 1-et írunk. Így a következő sorozatot kapjuk:

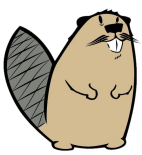
```
1, 1, 0, 1, 1, 1
```

Állítsd össze az eredeti, labdákból álló sorozatot, amelyből a fenti két lépésben a

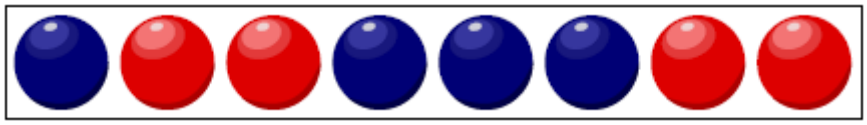
```
0, 1, 1, 1, 0, 1, 0, 0
```

sorozat állt elő!





A megoldás:



A választ többféleképpen is megtalálhatjuk. Az egyik lehetőség az, hogy a számsorozatban jobbról balra (visszafele) haladva logikusan következtetünk arra, hogy az adott labdának pirosnak vagy kéknek kell-e lennie.

A jobbról a legelső szám 0, emiatt a labdának pirosnak kell lennie, mivel csak így lehet páros a kék labdák száma:

0	1	1	1	0	1	0	0
?	?	?	?	?	?	?	**piros**

Jobbról a második helyen ismét 0-t találunk, így az előző okoskodáshoz hasonlóan itt is piros labda kell legyen:

0	1	1	1	0	1	0	0
?	?	?	?	?	?	**piros**	piros

Hátulról a harmadik helyen 1-et találunk, ami azt jelenti, hogy a kék labdák száma páratlan, azaz itt egy kék labdának kell állnia, hiszen eddig nem volt csak két piros labdánk.

0	1	1	1	0	1	0	0
?	?	?	?	?	**kék**	piros	piros

Hátulról a negyedik helyen megint egy 0-t találunk. Ez azt jelenti, hogy a kék labdák száma páros kell legyen. Mivel eddig 1 kék labdánk volt, itt is kék labdának kell lennie.

0	1	1	1	0	1	0	0
?	?	?	?	**kék**	kék	piros	piros

A következtetéseket tovább folytatva észrevehetjük, hogy amikor 0-ról 1-re vagy 1-ről 0-ra váltunk, akkor kell egy új kék labdának bekerülnie a sorba, hiszen ekkor változik a számuk párosról páratlanra, vagy páratlanról párosra.

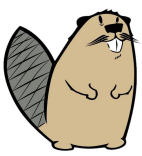
0	1	1	1	0	1	0	0
kék	piros	piros	kék	kék	kék	piros	piros

Ha a köztes sorozatot is szeretnénk leírni, az a 4, 3, 3, 3, 2, 1, 0, 0 lesz.

MIÉRT INFORMATIKA?

Ez a hódfeladat az informatika három központi problémáját is megmutatja:

A *kódolás* összetett tények olyan módon történő ábrázolására szolgál, hogy a számítógép könnyen feldolgozhassa azokat. Gyakran irreleváns (lényegtelen) információk elhagyhatóak anélkül, hogy a lényeges információ elveszne. Ez a feladat olyan kódot javasol, amely



a kék és a piros labdák pontos számát és helyzetét tárolja anélkül, hogy a tényleges színeket tárolni kellene. A választott kódolással vissza tudjuk állítani az eredeti információt (az eredeti labdák színét és helyzetét).

Bizonyos körülmények között a kódolással memóriaterületet is megtakaríthatunk, amit *tömörítés*nek nevezünk. A bináris információ (például, hogy egy szám páros vagy páratlan-e) ábrázolásához elegendő egyetlen bit információ, míg bármely szám vagy szín tárolásához több memóriára van szükség. Ha az utolsó sorozatból elhagyjuk a vesszőket, máris egy-egy bitként tárolható a sorozatunk minden eleme.

A feladatban az első köztes lépésként itt egy *kumulatív gyakoriságot* határoztunk meg. Ennek során kiszámítjuk azoknak az elemeknek az összegét, amelyek egy bizonyos jellemzőt hordoznak (ebben az esetben az aktuális pozícióból származó kék labdák számát). A kumulatív gyakoriság egy olyan statisztikai eszköz, amelyet különösen jellemez, hogy az így kapott érték információt halmoz fel (ebben az esetben nemcsak az aktuális labda színéről, hanem a mögötte lévő labdák színéről is információt szolgáltat). A kumulatív gyakoriságot használó problémamegoldásoknak gyakran előnye, hogy sok lekérdezés gyorsan feldolgozható anélkül, hogy az eredeti adatokhoz hozzá kellene férni.

WEBOLDAL

Kód (informatika) – Wikipédia

Adattömörítés – Wikipédia

Prefix sum - Wikipedia

KULCSSZAVAK

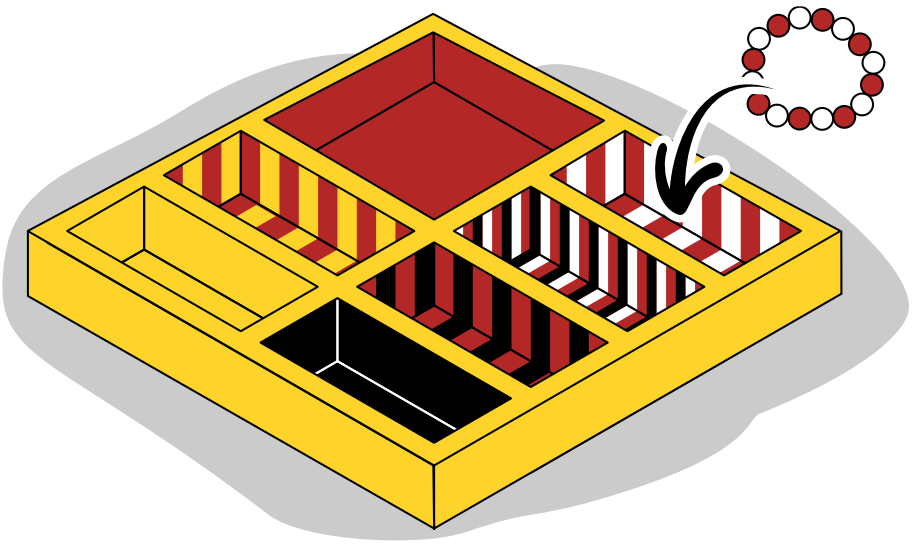
Kódolás, Tömörítés, Prefix sum array, Kumulatív gyakoriság

Karkötők rendszerezése (2024-BR-04)

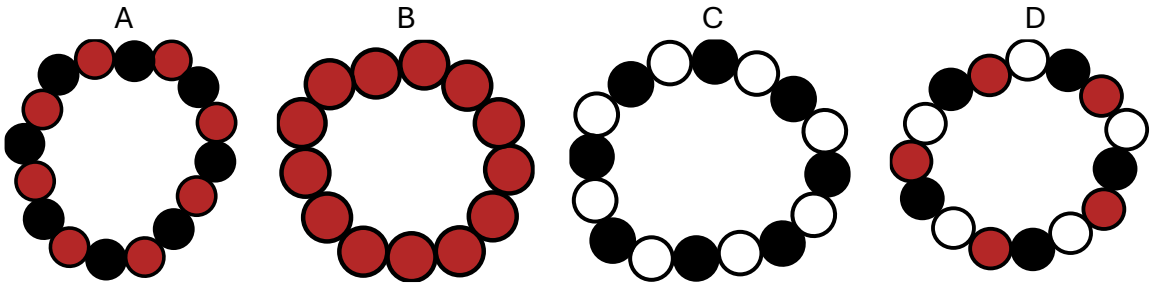
BENJAMIN – KÖNNYŰ

Viktória egy hétrekeszes dobozban tartja a karkötőit. Minden karkötőt csak olyan rekeszbe tesz, aminek a színei, mintája megegyezik a karkötőével.

Itt egy példa egy karkötőre és a hozzáillő rekeszre:

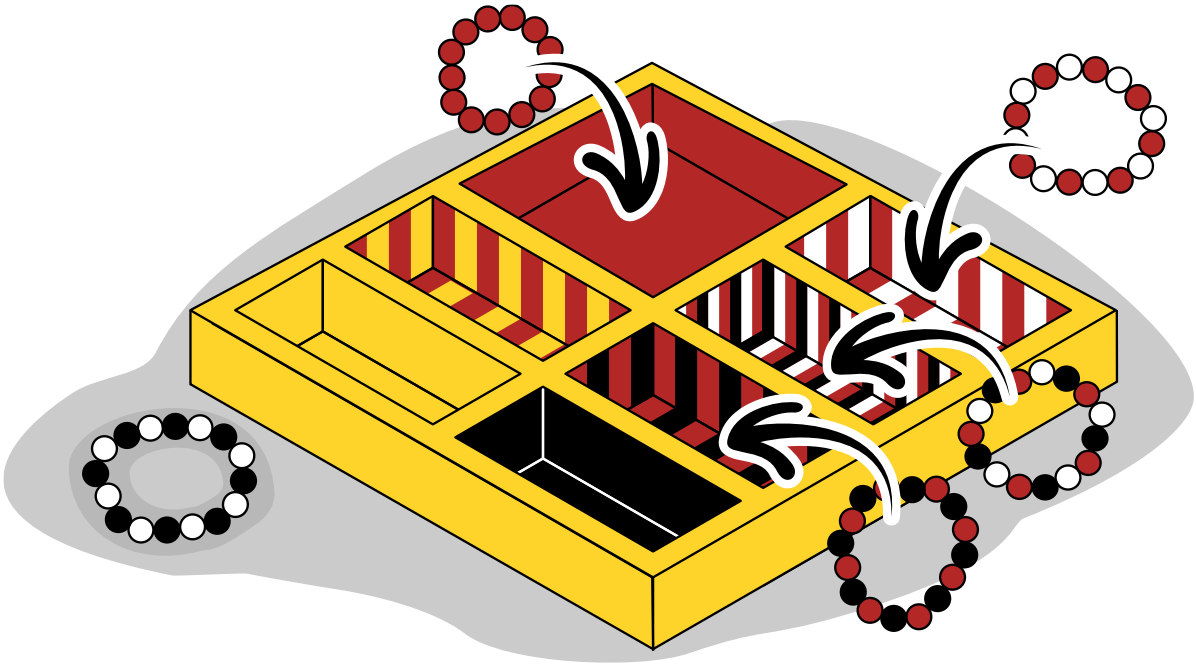


Az alábbi karkötők közül melyik nem illik egyik rekeszbe sem?



A C karkötő nem illik egyik rekeszbe sem.

Az alábbi képen látható, hogy az A, B és D karkötők melyik megegyező színű és mintájú rekeszbe illenek:



Nincs a C karkötővel megegyező felváltva fekete-fehér színmintás rekesz.

MIÉRT INFORMATIKA?

Gyakran előfordul, hogy az informatikai problémák megoldása során nagy mennyiségű adat áll rendelkezésünkre, de ezeknek csak egy kis része releváns egy adott kérdés szempontjából. Ennél a feladatnál is a karkötők egy tulajdonságára vonatkozó információra van szükségünk ahhoz, hogy a megfelelő rekeszbe tegyük őket. A karkötők sok más tulajdonságáról szóló információ nem segít a feladat megoldásában, csak a színeket és a felfűzés sorrendjét (minta) kell megnéznünk, a gyöngyök alakját vagy számosságát nem. Fontos, hogy kiszűrjük a nem szükséges adatokat. Így csak olyan adatokkal dolgozunk, amelyek a feladat megoldásához vezetnek.

Ezt az elvet a mesterséges intelligenciában is alkalmazzák, például az objektumok felismerésekor az adatokból csak néhány lényegest kell használnunk, ezzel növelve a számítás gyorsaságát.

WEBOLDAL

Adat – Wikipédia

Szűrő (programozás) – Wikipédia

KULCSSZAVAK

Mintaillesztés, adat

Szólánc (2024-CA-02)

KADÉT – NEHÉZ
JUNIOR – KÖZEPES
SENIOR – KÖNNYŰ

Hód a három karakterből álló kódokat úgy rendezi sorba, hogy az egyik kódból csak egy karakter változik a következőre lépve. Ezzel kódláncokat hoz létre.

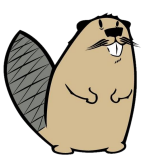
Például ezeket a kódokat teheti bele egy kódláncba: XUG → XUD → XED → KED

Hód kilenc kódot hozott létre: TOF, XEW, TEF, CET, COF, TEW, COT, CEF és XEF.

Ezeket három-három kódból álló kódláncokba helyezi úgy, hogy a kilenc kód mindegyike csak egy kódláncban szerepeljen.

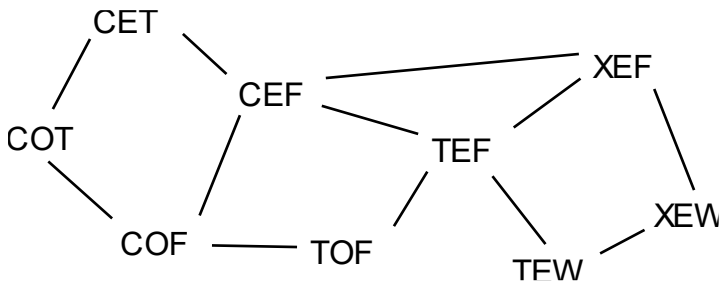
Az alábbi kódláncok egyike sem sérti meg a szabályokat, de az egyik lehetetlenné teszi, hogy három kódláncot készítsünk anélkül, hogy bármelyik kódot többször is felhasználjuk. Melyik?

A) XEW → TEW → TEF
B) COF → COT → CET
C) TEF → CEF → COF
D) CEF → CET → COT

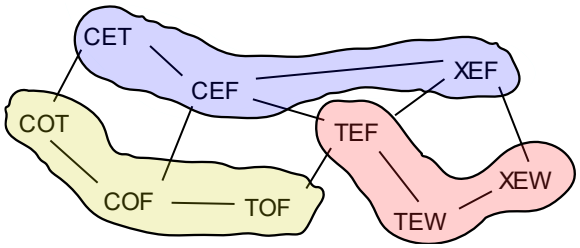


Megoldás/ A helyes válasz a C: TEF → CEF → COF.

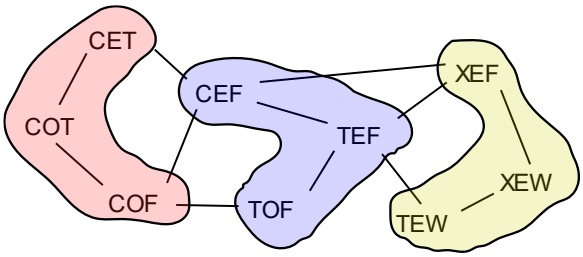
A probléma megoldásához segítségünkre lehet, ha rajzolunk egy olyan ábrát, ahol a kódokat összekötjük egy vonallal, ha az egyik követheti a másikat egy kódláncban.

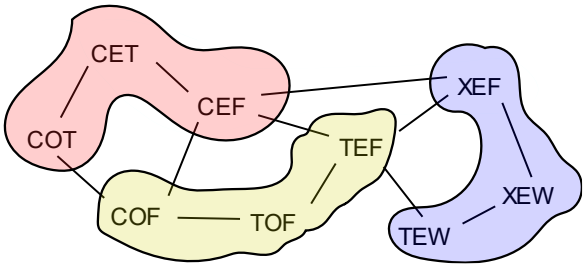


Az A) XEW → TEW → TEF esetben a másik két kódlánc a SAT → CAT → CAB és a COB → COT → BOT lehet

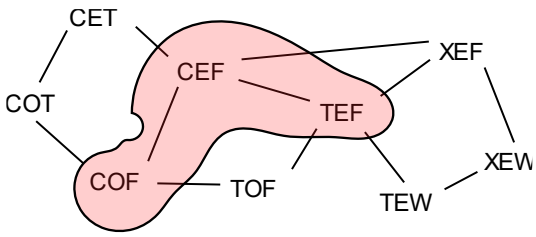


A B) és a D) esetben is tudunk másik két kódláncot alkotni 3-3 kóddal:





A C) TEF → CEF → COF esetben azonban a CET és a COT kódok egyike sincs egy harmadik kóddal összekötve, a TOF pedig csak a már felhasznált kódokba alakítható. Emiatt maximum a XEF → XEW → TEW kódlánc alakítható ki, de egy harmadik már nem.



MIÉRT INFORMATIKA?

A megoldás magyarázatához használt rajzokon minden kód egy csomópont. A két csomópont közötti él (vonalként rajzolva) pedig azt mutatja, hogy ez a két kód pontosan egy pozícióban különbözik egymástól. Ez a két csomópont tehát szomszédos.

Az informatikában a csomópontokból és élekből álló konstrukciókat általában gráfnak nevezik, és a csomópontok által reprezentált objektumok közötti kapcsolatoknak az élek segítségével történő ábrázolására használják.

Ebben a hódfeladatban a gráfot három összefüggő (azaz élekkel összekötött) részre kell felvágnunk. Ha ez sikerül, automatikusan létrejött a kódlánc minden részből.

A gráfok népszerű eszközei az informatikusok által használt komplex helyzetek modellezésének. A programok számos feladatot meg tudnak oldani gráfok segítségével. A navigációs rendszerek például olyan programok, amelyek nagyon nagy gráfok segítségével lépésről lépésre vezetik az autókat a kiindulási ponttól a célállomásig. Minden egyes lépésnél az autó egy élen halad át az egyik csomóponttól a szomszédos csomópontig.

WEBOLDAL

Gráf – Wikipédia

Szomszédság (gráfelmélet) – Wikipédia

KULCSSZAVAK

Gráf, Szomszédság

Ricca kártyák (2024-CH-03a)

NEHÉZSÉG 1 – LEÍR BEKEZDÉSSEL, ÚJ SORBAN FELSOROLVA
NEHÉZSÉG 2 – LEÍR BEKEZDÉS FOLYTATÁSA ÚJ SORBAN.

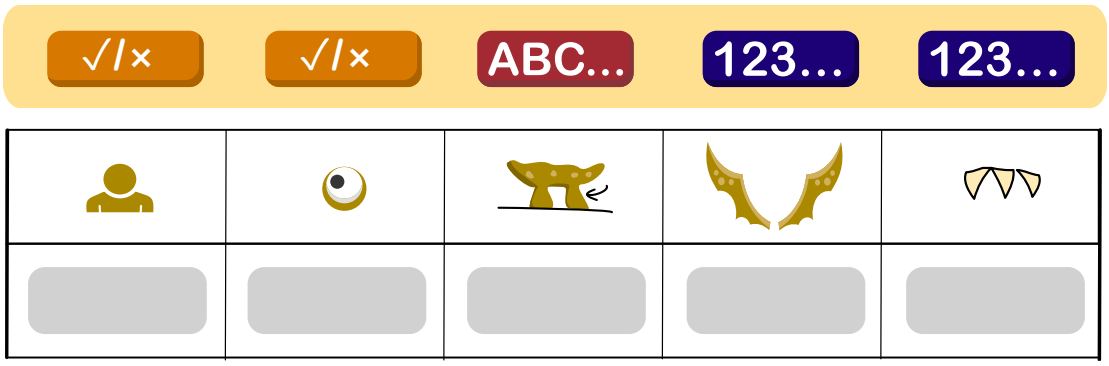
Barbara szörnyeket ábrázoló kártyákat, „riccákat” gyűjt. Itt vannak a Ricca-kártyái:

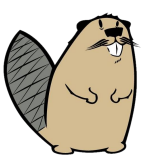
Minden kártyán szerepelnek tulajdonságok, például a ricca neve ( ) vagy az, hogy vannak-e fogai ( ). Ezeknek a tulajdonságoknak értékei vannak. A 2. kártyán például a név ( ) értéke MONI, a fogaké ( ) pedig ✓: A kártyán szereplő riccát tehát MONI-nak hívják és vannak fogai.

Barbara rájön, hogy a tulajdonságok csak háromfélék. Ezeket „típusoknak” nevezi:

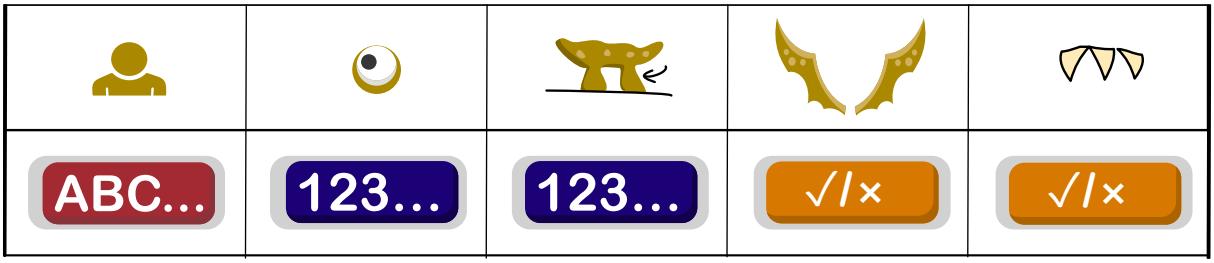
ABC...	több betű egymás után
123...	pl. 1, 2, ...
√/×	✓ és X, amik «igen»-t és «nem»-et jelentenek

Húzd a típusokat a tulajdonságok alá úgy, ahogy a kártyákon látható





A megoldás:



Ha megnézzük a kártyákat, látszik, hogy melyik tulajdonság mellett, milyen értékek szerepelnek.

kártya	Név ( )	Szemek ( )	Lábak ( )	Szárnyak ( )	Fogak ( )
1	JOSI	3	2	X	✓
2	MONI	3	2	X	✓
3	KILI	2	2	✓	✓
4	BENI	2	2	X	✓
5	LORI	2	5	X	✓
6	PHIL	2	2	X	✓

A táblázatból is leolvasható, hogy a lábaknál és a szemeknél számok szerepelnek, ezért ezek szám típusúak. A szárnyak és fogak esetében csak ✓ vagy X szerepel, emiatt ezek igen/nem típusúak.

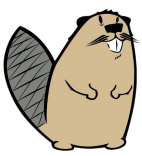
Ugyan a szemeket is lehetne igen/nem-mel jelölni (van szeme vagy szemtelen), de a kártyákról nem ez olvasható le.

MIÉRT INFORMATIKA?

A Ricca-kártya olyan, mint egy számítógépen tárolt adatrekord. A számítógépes memóriák elektronikája azonban csak 1-eket és 0-kat tud tárolni. A számítógépes programok ezért „értéktípusokat” is használnak, amelyeket az informatikában adattípusoknak neveznek, hogy meghatározzák, hogyan értelmezze és kezelje a számítógép ezeket az 1-eket és 0-kat. Az adattípusok segítségével azt is ellenőrizni lehet, hogy a tárolt adatok kombinálhatók-e egymással (pl. összeadhatóak, összehasonlíthatóak-e), és ha igen, hogyan.

Az értéktípusok ebből a hódfeladatból megfelelnek a programozás fontos adattípusainak:

- A szöveget a programozási nyelvekben gyakran „sztring”(string) néven emlegetik: Az ilyen típusú adatok karakterek (betűk, számjegyek és egyéb karakterek) sorozatai.
- A számokból több típus is létezik. A feladatban az egész szerepel, ami megfelel az „integer” típusnak: Az ilyen típusú adatok egész számok, azaz ... -2, -1, 0, 1, 2, ...



- Igen/nem megfelel a „boolean” típusnak: Ez az adattípus csak két különböző értéket ismer, amelyeket gyakran „true” és „false” (igaz és hamis) értékeknek neveznek.

A kártyákon szereplő tulajdonságok és azok értékei a riccákat írják le. A képek azonban sokkal több információt tartalmaznak a Riccákról, mint ami a kártyákra van írva. Például nemcsak azt láthatjuk, hogy egy riccának van-e foga vagy sem, hanem azt is, hogy hány foga van. Fontos, hogy alaposan átgondoljuk, milyen típusként tároljuk az adatokat, hiszen a későbbiekben a hiányzó információkat, adatokat már nem tudjuk akkor visszakeresni.

WEBOLDAL

Adattípus – Wikipédia

String – Wikipédia

Boolean – Wikipédia

KULCSSZAVAK

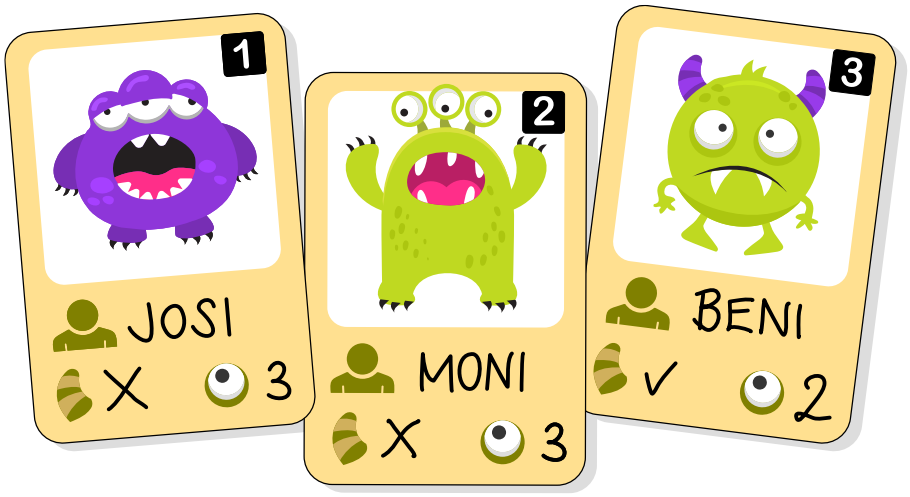
Adattípus, Szöveg (String), Logikai (boolean) típus

Ricca kártyák (2024-CH-03b)

KÖNNYŰ – KISHÓD

Barbara szörnyeket ábrázoló kártyákat, „riccákat” gyűjt.

Minden kártyán szerepelnek tulajdonságok, például a ricca neve ( ) vagy az, hogy a hány szeme van ( ), vagy van-e szarva ( ). Itt vannak Barbara Ricca-kártyái:



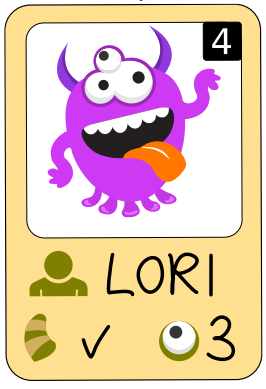
Minden tulajdonsághoz meghatározott, milyen érték tartozhat.

	Tulajdonság	Megengedett érték
	név	szöveg (több betű)
	szemek száma	szám
	van szarva	✓ vagy X

Barbara kap négy új kártyát, de csak az egyiken van minden tulajdonságnak megengedett értéke. Melyiken?

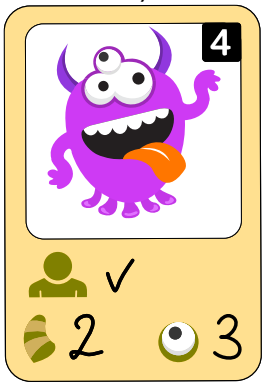
A)

B)



C)

D)



A helyes válasz a B)

Nézzük meg a kártyákat, melyik tulajdonság mellett, milyen értékek szerepelnek.

	A	B	C	D
	LORI	LORI	LORI	✓
	2	✓	2	2
	✓	3	3	3

Csak a B képen van minden tulajdonságnak megengedett értéke. Az A, C és D kártyákon egy szám áll a szarv () mellett, ami ugyan igaz, hiszen a riccának 2 szarva () van, de nem megengedett érték, mivel a  tulajdonság csak ✓ vagy X állhat. Az A kártyán a szem (), a D kártyán pedig a név () nem megengedett.

MIÉRT INFORMATIKA?

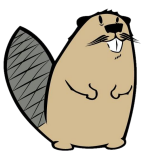
A Ricca-kártya olyan, mint egy számítógépen tárolt adatrekord. A számítógépes memóriák elektronikája azonban csak 1-eket és 0-kat tud tárolni. A számítógépes programok ezért „értéktípusokat” is használnak, amelyeket az informatikában adattípusoknak neveznek, hogy meghatározzák, hogyan értelmezze és kezelje a számítógép ezeket az 1-eket és 0-kat.

Az adattípusok segítségével azt is ellenőrizni lehet, hogy a tárolt adatok kombinálhatók-e egymással (pl. összeadhatóak, összehasonlíthatóak-e), és ha igen, hogyan.

Az értéktípusok ebből a hódfeladatból megfelelnek a programozás fontos adattípusainak:

- A szöveget a programozási nyelvekben gyakran „sztring”(string) néven emlegetik: Az ilyen típusú adatok karakterek (betűk, számjegyek és egyéb karakterek) sorozatai.
- A számokból több típus is létezik. A feladatban az egész szerepel, ami megfelel az „integer” típusnak: Az ilyen típusú adatok egész számok, azaz … -2, -1, 0, 1, 2, … .
- Igen/nem megfelel a „boolean” típusnak: Ez az adattípus csak két különböző értéket ismer, amelyeket gyakran „true” és „false” (igaz és hamis) értékeknek nevezznek.

A kártyákon szereplő tulajdonságok és azok értékei a riccákat írják le. A képek azonban sokkal több információt tartalmaznak a riccákról, mint ami a kártyákra van írva. Például nemcsak azt láthatjuk, hogy egy riccának van-e szarva vagy sem, hanem azt is, hogy hány van. Fontos, hogy alaposan átgondoljuk, milyen típusként tároljuk az adatokat, hiszen a későbbiekben a nem tárolt információkat, adatokat már nem tudjuk akkor visszakeresni.



WEBOLDAL

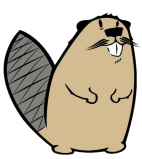
Adattípus – Wikipédia

String – Wikipédia

Boolean – Wikipédia

KULCSSZAVAK

Adattípus, Szöveg (String), Logikai (boolean) típus



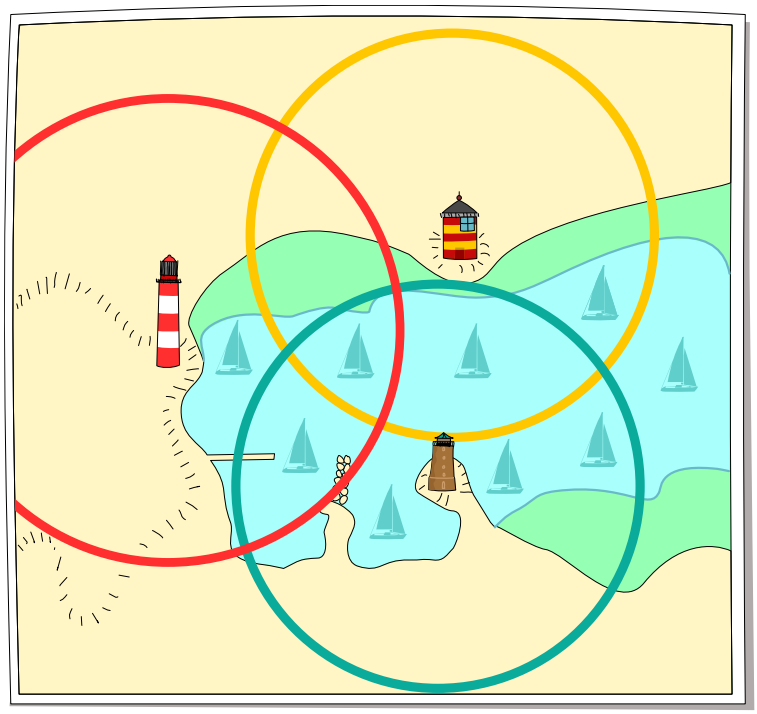
Világítótornyok (2024-CZ-05)

KÖNNYŰ – KISHÓD

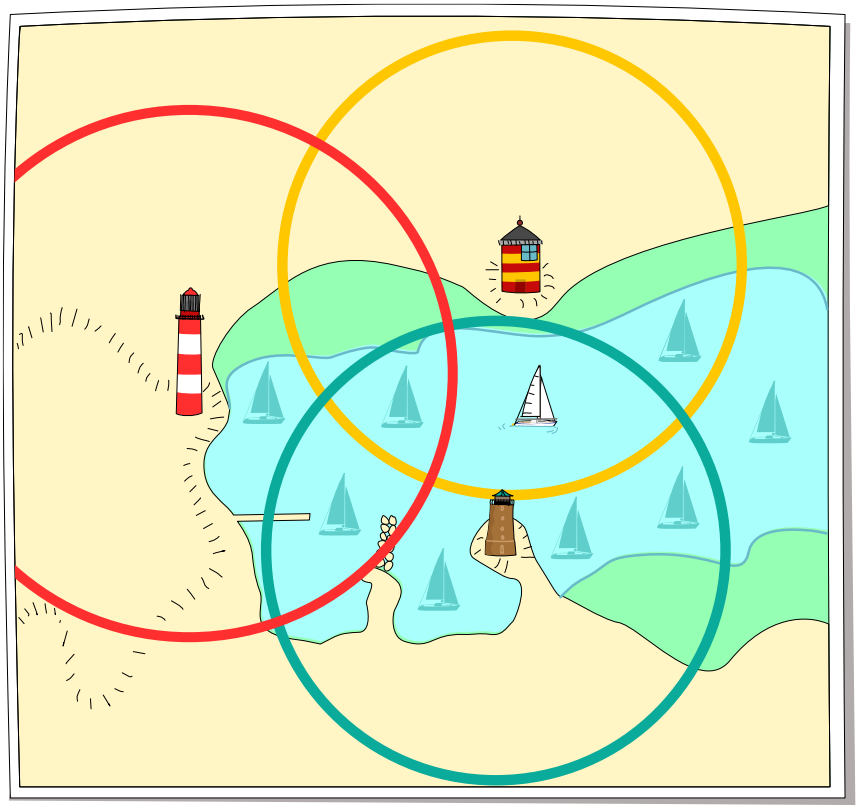
Bence vitorlázik (), és nála van egy térkép a kikötővel és a világítótornyokkal. Minden világítótorony köré rajzol egy-egy kört. Ha Bence hajója az adott körön belül van, akkor látja az ahhoz tartozó világítótornyot.

Bence látja a  és a  világítótornyot, de nem látja a  -t.

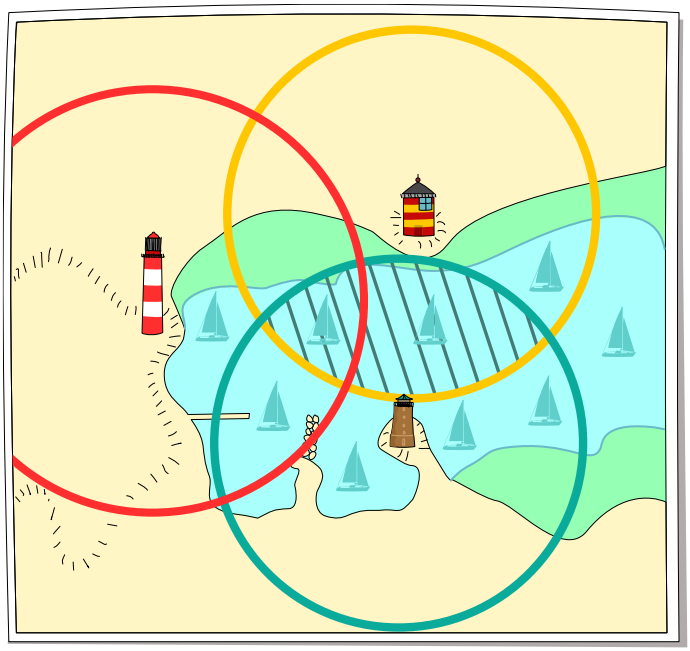
Melyik Bence hajója?



A megoldás:

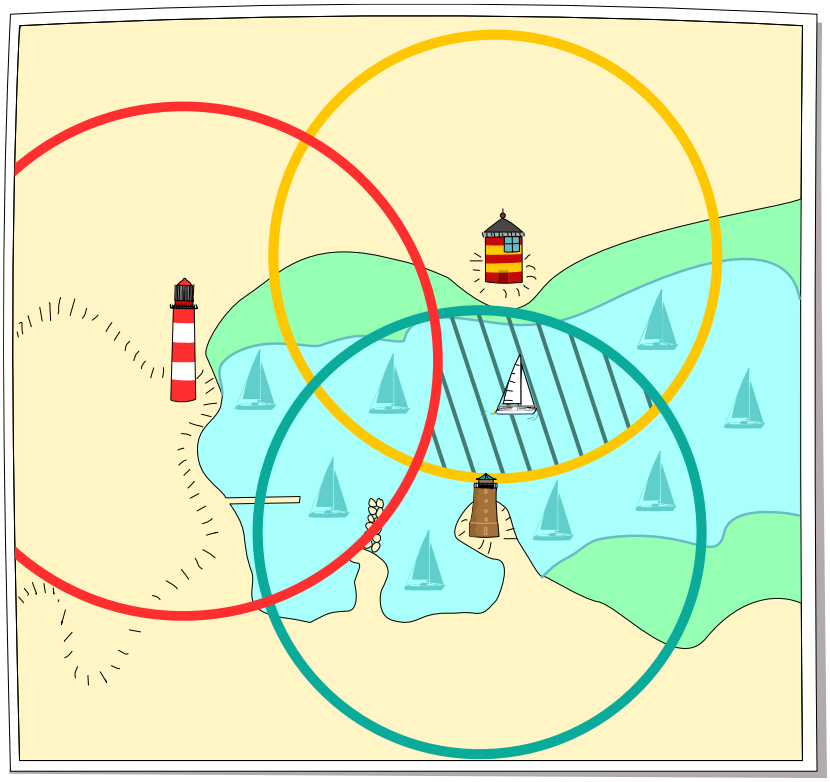


Mivel Bence látja a  és a  világítótornyot, azon a területen van, ami a két kör mindegyikében benne van. A képen a satírozott rész.



De Bence nem látja a  világító tornyot, azaz nincs az ehhez tartozó körben.

Marad a satírozott terület a képen, ahol csak egy hajó van.



.

MIÉRT INFORMATIKA?

A vízen nem csak a csillagokat megfigyelve, de a parton jól látható pontok segítségével is tájékozódhatunk (ezt nevezik szárazföldi navigációnak). Ma már a legtöbb hajó műholdas navigációval van felszerelve, és a jól látható pontok helyett műholdas jeleket használnak, a helyzetet pedig számítógép segítségével számítják ki. Mindazonáltal minden hajós megtanul műholdjelek nélkül is navigálni, hiszen előfordulhat, hogy azok nem állnak rendelkezésre.

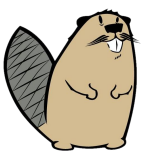
Az informatikában nagy jelentősége van a probléma pontos leírásának és a megoldási módszernek. Ennek hasznos eszköze a formális logika: Bence három állítással helyezhető el a térképen, amelyek mindegyike lehet igaz vagy hamis:

Az A állítás a következő: „Az A világítótorony látható”.

A B állítás: „A B világítótorony látható”.

A C állítás: „A C világítótorony látható”.

Bence esetében B és C igaz, A hamis (Bence látja a B és C világítótornyot, de az A világítótornyot nem). A logikai állításban ezt így írjuk le: „B ÉS C ÉS NEM A”. A térképen például az „A ÉS B ÉS C” annak a területnek felel meg, ahol mindhárom kör átfedi egymást, mert



csak ott látható mindhárom világítótorony. A „NEM A ÉS NEM B ÉS NEM C” annak a területnek felel meg, amely a három körön kívül esik. Ott a három világítótorony egyike sem látható.

Általánosságban elmondható, hogy az informatikában a logikai állításokat ismeretek reprezentálására, programok ellenőrzésére és mindenekelőtt digitális áramkörök modellezésére használják.

WEBOLDAL

Ítéletlogika – Wikipédia

Boole-algebra (informatika) – Wikipédia

Navigáció – Wikipédia

KULCSSZAVAK

Logika, Boole-algebra, Navigáció

Pizza Party (2024-DE-02)

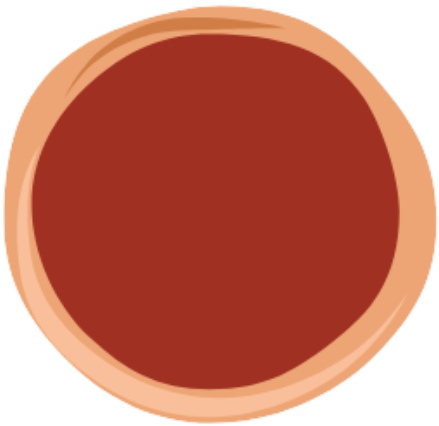
BENJAMIN – NEHÉZ
KADÉT – KÖZEPES
JUNIOR – KÖNNYŰ

János pizzapartit rendez. Ismeri barátainak a kedvenc feltéteit:

Anna			
Bálint			
Cecil			
Diána			

Egyetlen pizzát szeretne készíteni, 3 feltéttel. Ezeket úgy akarja kiválasztani, hogy összességében barátainak a legtöbb kívánsága teljesüljön.

Segíts Jánosnak kiválasztani 3 feltétet a pizzához!



A helyes válasz:

Mindenki szereti a gombát  , hárman (majdnem mindenki) szeretik a sajtot  , ketten pedig a hagymát  . Tehát ezeket kell választania.

Hogy találjuk meg a megoldást? Megszámolhatjuk, hogy melyik feltétet hányan szeretik, majd kiválasztjuk azt a hármat, amit a legtöbben szeretnek. Nézzük a megadott adatokat és oldjuk meg a feladatot lépésről lépésre.

1. lépés: Számoljuk meg az egyes feltétek gyakoriságát:

						
Anna	1	1			1	
Bálint		1	1			1
Cecil		1		1	1	
Diána		1			1	1
Összesen	1	4	1	1	3	2

Az utolsó sorban összeszámoltuk, melyik feltétet hányan szeretik.

2. lépés: Határozzuk meg a három legnépszerűbb feltétet

Azaz keressük meg az utolsó sorban a legnagyobb számokat. A gomba a legnépszerűbb feltét, mind a 4 barát kedvence, ezt követi a sajt (3-szor említve) és a hagyma (2 említéssel). A legkevesebbet a pirospaprikát, az ananászt és a rukkolát említették, mindet csak egyszer

Ez az eljárás akkor is alkalmazható, ha a barátok vagy a feltétek száma sokkal több.

MIÉRT INFORMATIKA?

Ez a hódfeladat egy jó példa egy egyszerű optimalizálási problémára.

A megoldás megtalálásához adatokat kell gyűjteni (a barátok kívánságai), új adatokat kell kiszámítani (a kívánságok gyakorisága), feltételeket kell figyelembe venni (csak egy pizza három feltétellel) és döntéseket kell hozni (a feltét kiválasztása).

Ahhoz, hogy egy optimalizálási probléma megoldása során döntést hozhassunk, képesnek kell lennünk arra, hogy a lehetséges válaszokat vagy megoldásokat összehasonlítsuk egymással. Ehhez szükség van egy mértékre, hogy mennyire jó egy megoldás. Ebben a hódfeladatban a teljesített kívánságok száma az optimalizálási mérték: minél több kívánság teljesül, annál jobb. Az optimalizáló algoritmus az egyes feltétekkel kapcsolatos kívánságok összeadásával közvetlenül ki tudta választani azt a megoldást, amelyik a legjobb értéket adta az optimalizációs mértékre. Ebben az esetben tehát nem volt szükség a különböző megoldások egymással való összehasonlítására.

Az optimalizációs algoritmusokat az informatikában számos célra fejlesztik ki, például az ipari termelési folyamatokhoz vagy a csomagküldő vagy egyéb kézbesítő szolgálatok szállítójárműveinek útvonalainak kiszámításához. Az optimalizálási mértékek nagyon különbözőek lehetnek, például a gyártási folyamat sebessége vagy az útvonal hossza. Egy dolog azonban mindig ugyanaz: át kell gondolni, hogyan lehet a lehető legügyesebben kiválasztani a sok lehetséges megoldás közül a legjobbat. Ez ritkán olyan egyszerű, mint ebben a hódmezővásárhelyi feladatban.

WEBOLDALAK

Matematikai optimalizálás – Wikipédia

Algoritmus – Wikipédia

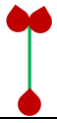
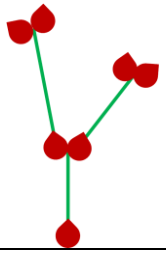
KULCSSZAVAK

Optimalizáció, Algoritmus

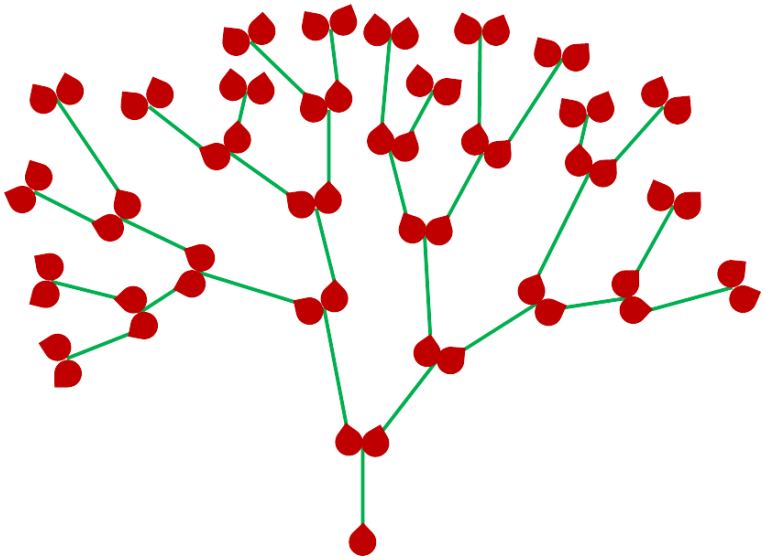
Csodavirág (2024-DE-03)

KISHÓD – NEHÉZ
BENJAMIN – KÖZEPES
KADÉT – KÖNNYŰ

Napkeltekor a csodavirág minden új rügyéből egy szár nő. A szár a nap folyamán tovább növekszik. Napnyugtakor a szár két új rügyre ágazik és éjszakára megáll a növekedésben. Ez folytatódik napról napra és a csodavirág egyre pompásabb lesz.

Rügy az első napkelte előtt	Csodavirág napnyugta után (első nap)	Csodavirág napnyugta után (második nap)
		

Hány napja nő ez a csodavirág?



A helyes válasz: 5 nap

Az ágak azokból a szárakból állnak, amelyek mentén az első bimbótól egy olyan külső bimbóig haladunk, amelyből még nem nőtt ki szár. Az 1. nap végén csak egy ág van, és az 1 szár hosszú. A második nap végén már két ág van, és ezek két szár hosszúak. Ahogy a csodavirág növekszik, az ágak minden nap egy szárral hosszabbak lesznek.

Tehát csak egy ágat kell követned, pl. a jobb szélső ágat, és megszámolnod, hogy hány szár hosszú. Az ág 5 szár hosszú, így tudod, hogy a csodavirág 5 napig nőtt.

MIÉRT INFORMATIKA?

A csodavirágok egyetlen szabály szerint nőnek. A szabály egyszerű, kimondja, hogy egy rügyből egy szár és két új rügy keletkezik. Ily módon a szabály önmagát tartalmazza és a csodavirág elméletileg a végtelenségig nőhet. Az ilyen önmagukat tartalmazó szabályokat rekurzív szabályoknak nevezzük.

A programozásban a rekurzív függvény olyan függvény, amely önmagát hívja meg: a függvény meghívja magát, elindul ugyanannak a függvénynek egy új példánya, és meghívja magát, az új példányt... és így tovább. Fontos, hogy legyen bennmaradási feltétel, vagyis a függvény csak akkor hívja meg magát, ha valamilyen feltétel teljesül. Ebben az esetben bizonyos számú hívás után a folyamat akkor áll le, amikor a feltétel már nem teljesül.

A rekurzív programok érthetők és könnyen olvashatók lehetnek. A sok függvényhívás miatt azonban általában lassabb a futási idejük és sok memóriát igényelnek.

A rekurzív programokkal nagyon természetes kinézetű, fraktáloknak nevezett grafikákat is létrehozhatunk.

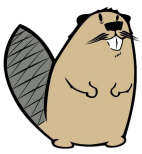
A rekurzió másik példája a francia hód képe. A pólóján a francia hód képe látható, aki pólót visel a francia hód képével stb. A kép önmagát tartalmazza.

WEBOLDAL

Rekurzió – Wikipédia

KULCSSZAVAK

Rekurzió, Grafika



Napsütéses napok (2024-DE-04a)

BENJAMIN – NEHÉZ
KADÉT – KÖZEPES
JUNIOR – KÖNNYŰ

Tomi, a hód azt mondja: „Napos napokon minden tóban úszik legalább egy hód.”

A nővére, Kata így válaszol: „Ez nem igaz! A múlt vasárnap egy példa arra, hogy bebizonyítsam, tévedsz.”

Tegyük fel, hogy Katának igaza van. Egészítsd ki a következő mondatot!

A múlt vasárnap sütött a nap és _____.

A) minden tavacska tele volt úszó hódokkal

B) a Jeges Vízeséses tóban nem úszott hód

C) Misi hód úszott minden tóban

D) Misi hód egyáltalán nem úszott

A helyes válasz a B): a Jeges Vízeséses tóban nem úszott hód.

Ebben az esetben nem minden tóban úszkál hód, mert a Jeges Vízesés tóban nem. Így Tomi téved, amikor azt mondja, hogy minden tóban van egy hód.

A többi válaszlehetőség nem bizonyítja, hogy Tomi téved:

Az A) lehetőség szerint „A múlt vasárnap sütött a nap, és minden tó tele volt úszó hódokkal.” Ez azt jelenti, hogy minden tóban hódok úszkáltak, ami megfelel Tomi kijelentésének.

A C) lehetőség szerint „Múlt vasárnap sütött a nap, és Misi hód úszott minden tóban.” Ez azt jelenti, hogy minden tóban legalább Misi úszott egyet, ami ismét megfelel Tomi kijelentésének.

A D) lehetőség szerint „Múlt vasárnap sütött a nap, és Misi hód egyáltalán nem úszott.” Azonban előfordulhat, hogy minden tóban más hódok is úszkálnak, ebben az esetben Tomi kijelentése továbbra is teljesülhetne.

MIÉRT INFORMATIKA?

Tomi és Kata megvitatják Tomi kijelentésének igazságértékét, belemerülve a logika birodalmába, ahol az állításokat igaznak vagy hamisnak minősíthetik. A "logika" az egyszerű és bonyolult állítások tudománya, ami az ÉS, VAGY, NEM, és a MINDEN és LEGALÁBB EGY kifejezésekkel is leírható.

A logika a vitákban segít elkerülni a félreértéseket, hibákat strukturált módszert kínálva az állítások vizsgálatára.

A logika alapvető fontosságú az informatikai rendszerekben. Minden számítógép olyan apró fizikai elemekből áll, amelyek logikai műveleteket tudnak végrehajtani csak igaz és hamis értékek helyett 1 és 0 szerepel.

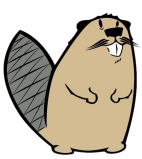
A számítógépek programozására használt nyelvek is használnak logikai utasításokat, például: "HA nyár van ÉS süt a nap, akkor kapcsolja be a légkondicionálást".

WEBOLDALAK

Logika – Wikipédia

KULCSSZAVAK

Logika, logikai operátorok



Napsütéses napok (2024-DE-04b)

SENIOR – KÖZEPES

Tomi, a hód azt mondja: „Napos napokon minden tóban úszik legalább egy hód.”

A nővére, Kata így válaszol: „Ez nem igaz! A múlt vasárnap egy példa arra, hogy bebizonyítsam, tévedsz.”

Tegyük fel, hogy Katának igaza van. Egészítsd ki a következő mondatot!

Múlt vasárnap, _____ és _____

a) sütött a nap	a) minden tó tele volt úszó hódokkal
b) nem sütött a nap	b) a Jeges Vízeséses tóban nem úszott hód
	c) Misi hód úszott minden tóban
	d) Misi hód egyáltalán nem úszott

A helyes válasz: A múlt vasárnap sütött a nap és a Jeges Vízeséses tóban nem úszott hód.

Tomi a napsütéses napokról állít valamit. Tehát, ha be akarjuk bizonyítani, hogy helytelen, amit a napsütéses napokról mondott, akkor meg kell mutatnunk, hogy van olyan napsütéses nap, amely nem teljesíti az állítását. Tomi nem mond semmit a nem napos napokról. Állítása tehát attól függetlenül lehet igaz, hogy mi történik a nem napos napokon. Tehát az első vonalra a „sütött a nap" kell kerüljön.

A gondolatmenetet folytatva Tomi akkor téved, ha van olyan tó, amiben nem úszkál hód. Ezt a „a Jeges Vízesés tóban nem úszott hód" rész adja meg.

A többi válaszlehetőség nem bizonyítja, hogy Tomi téved:

A „Múlt vasárnap sütött a nap, és minden tó tele volt úszó hódokkal." azt jelenti, hogy minden tóban hódok úszkáltak, ami megfelel Tomi kijelentésének.

A „Múlt vasárnap sütött a nap, és Misi hód úszott minden tóban." azt jelenti, hogy minden tóban legalább Misi úszott egyet, ami ismét megfelel Tomi kijelentésének.

A „Múlt vasárnap sütött a nap, és Misi hód egyáltalán nem úszott." megint nem cáfolja meg az eredeti állítást, hiszen előfordulhat, hogy minden tóban más hódok is úszkálnak, ebben az esetben Tomi kijelentése továbbra is teljesülhetne.

MIÉRT INFORMATIKA?

Tomi és Kata megvitatják Tomi kijelentésének igazságértékét, belemerülve a logika birodalmába, ahol az állításokat igaznak vagy hamisnak minősíthetik. A "logika" az egyszerű és bonyolult állítások tudománya, ami az ÉS, VAGY, NEM, és a MINDEN és LEGALÁBB EGY kifejezésekkel is leírható.

A logika a vitákban segít elkerülni a félreértéseket, hibákat strukturált módszert kínálva az állítások vizsgálatára.

A logika alapvető fontosságú az informatikai rendszerekben. Minden számítógép olyan apró fizikai elemekből áll, amelyek logikai műveleteket tudnak végrehajtani csak igaz és hamis értékek helyett 1 és 0 szerepel.

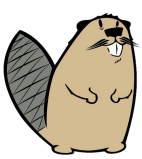
A számítógépek programozására használt nyelvek is használnak logikai utasításokat, például: "HA nyár van ÉS süt a nap, akkor kapcsolja be a légkondicionálást".

WEBOLDALAK

Logika – Wikipédia

KULCSSZAVAK

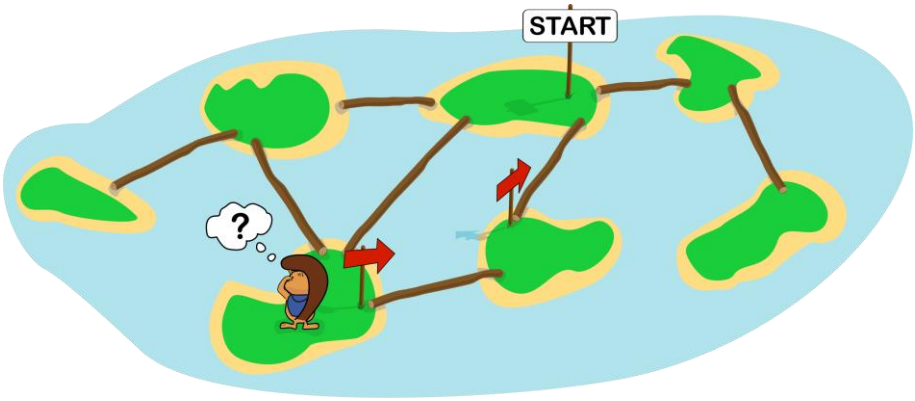
logika



Felfedezés (2024-DE-05)

SENIOR – NEHÉZ

Huba egy fatörzsekkel összekötött szigetcsoportot látogat meg. Minden szigetre legalább egyszer el akar jutni. Ehhez táblákat (Start és nyilak) visz magával. Amikor egy szigeten megáll, látja az összes szomszédos szigetet és megnézi, hogy van-e rajta tábla:



A fenti képen látjuk, hogy már három szigeten járt. A következő utasítások szerint mozog:

Állj egy tetszőleges szigetre és helyezd le a START táblát.

Minden alkalommal, amikor egy szigetre lépsz, tedd a következőket:

Ha ezen a szigeten még nincs tábla, **akkor**:

Helyezz le egy nyíl táblát, ami arra a szigetre mutat, ahonnan jöttél.

Ha van olyan szomszédos sziget, amin < nincs tábla /van nyíl tábla/START tábla van >, **akkor**:

Menj erre a szigetre.

Különben

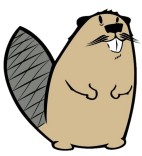
Ha egy szigeten, amin állsz, van nyíl tábla, **akkor**:

Menj arra a szigetre, < ahonnan jöttél/amire a nyíl mutat >

Különben:

Maradj a szigeten, hiszen a szigetcsoport összes szigetére eljutottál már.

Egészítsd ki a hiányos utasításokat úgy, hogy ha Huba e szerint mozog, minden szigetre legalább egyszer eljusson.



Megoldás:

Állj egy tetszőleges szigetre és helyezd le a START táblát.

Minden alkalommal, amikor egy szigetre lépsz, tedd a következőket:

Ha ezen a szigeten még nincs tábla, **akkor**:

Helyezz le egy nyíl táblát, ami arra a szigetre mutat, ahonnan jöttél.

Ha van olyan szomszédos sziget, amin < **nincs tábla** /van nyíl tábla/START tábla van >, **akkor**:

Menj erre a szigetre.

Különben

Ha egy szigeten, amin állsz, van nyíl tábla, **akkor**:

Menj arra a szigetre, < ahonnan jöttél/**amire a nyíl mutat** >

Különben:

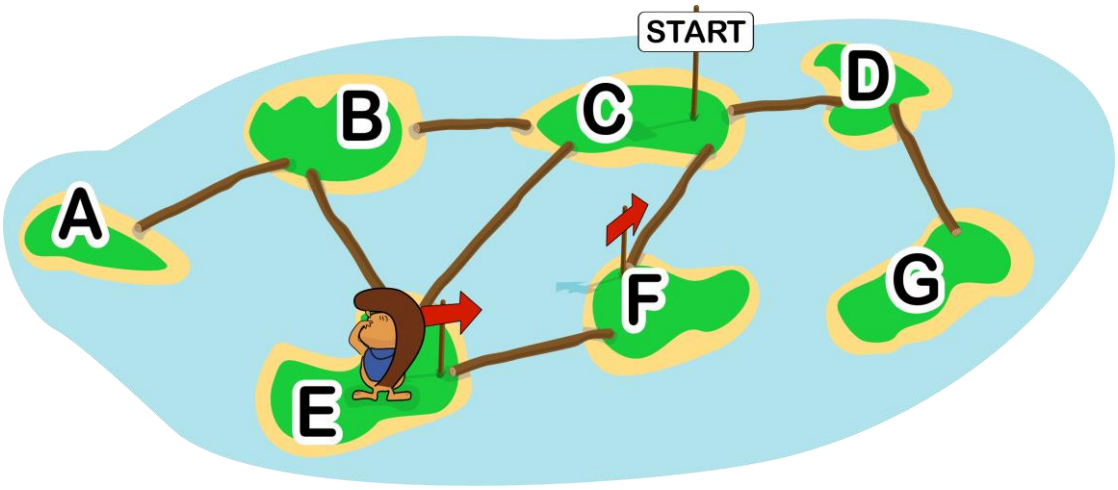
Maradj a szigeten, hiszen a szigetcsoport összes szigetére eljutottál már.

Első kitöltendő hely:

Huba mindig letesz egy táblát, amikor egy tábla nélküli szigetre ér.

Emiatt az összes szigeten, ahol Huba már járt, van tábla és az összes szigeten, ahol még nem, nincs tábla. A helyes szöveg tehát az, hogy „nincs tábla”. Ez biztosítja, hogy tudja, mikor megy olyan helyre, ahol még nem járt.

Például a képen látható helyzetben a B szigetre kell mennie, mert ez az egyetlen szomszédos sziget, ahol nincs tábla, így még nem járt ott.



Ha a „van nyíl tábla” szöveget választanánk, az azt jelentené, hogy Huba csak azokra a szigetekre megy, ahol már járt. Emiatt nem tudna mindegyikre eljutni.

A „a START tábla van” szintén hibás, mert a következőt jelentené: Miután Huba átmegy a második szigetre, mindig visszamegy arra a szigetre, ahol a START tábla van és ott is marad. Ekkor még nem járt az összes szigeten.

Második kitöltendő hely:

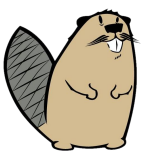
Huba a nyilakkal tudja visszakövetni az utat arra a helyre, ahol elkezdte az útját.

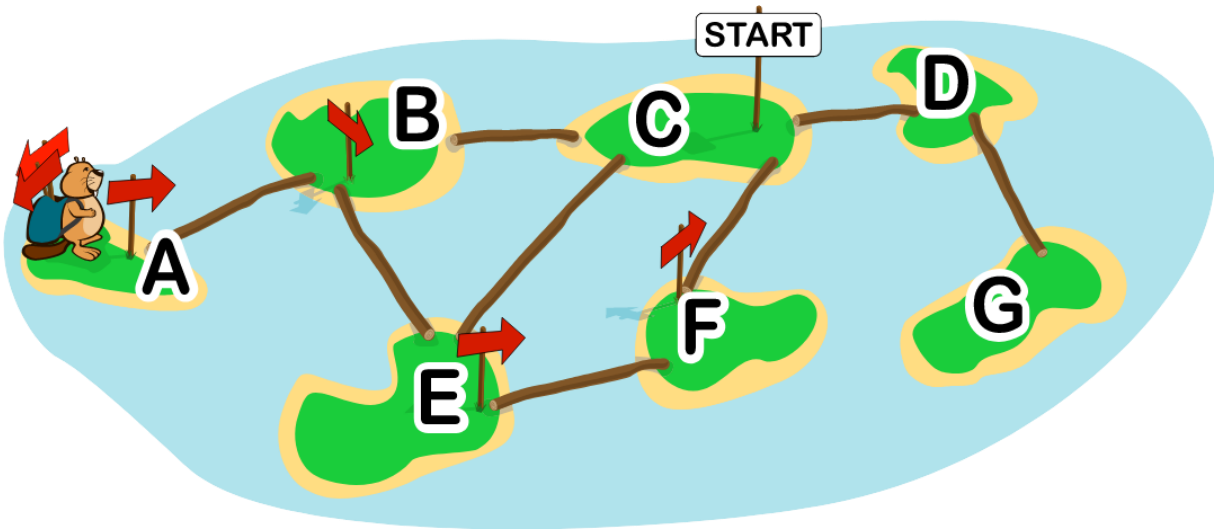
A helyes szövegrész az *„amire a nyíl mutat”*. Ez biztosítja, hogy Huba megtalálja a útját a táblákkal egészen addig, amíg egy olyan szigetre nem ér, amelynek szomszédos szigetén még nem járt. Ezt a technikát „visszalépésnek” nevezzük.

Az *„ahonnan jöttél”* azt jelentené, hogy Huba örökké ingázna két sziget között anélkül, hogy szomszédos szigeteket érintene.

Például a következő helyzetben Huba az A szigeten tartózkodik, amelynek nincs szomszédos, tábla nélküli szigete. Az első („ahonnan jöttél”) szövegrésszel a következő lépésben B-re menne, majd vissza A-ra, majd vissza B-re stb. És soha többé nem hagyná el ezt a két szigetet.

A helyes megoldásunk („amire a nyíl mutat”) azonban a nyilakat követve visszamegy a C szigetre, majd a D szigetre, mert ott még nincs tábla.





MIÉRT INFORMATIKA?

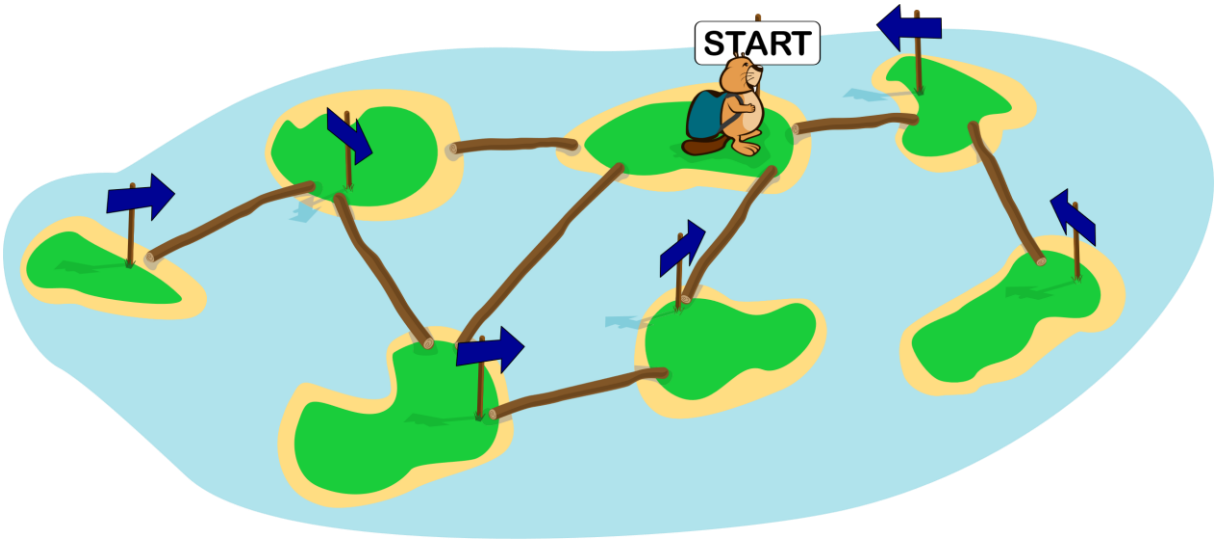
Az egymással összekapcsolt szigetek hálózata egy gráffal modellezhető.

A gráf egy adatstruktúra. Csomópontokból (a szigetek) és élekből (a szigeteket összekötő fatörzsek) áll.

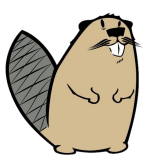
A valóságban számos rendszer ábrázolható gráfokkal. Például az embereket barátságok, a vasútállomásokat sínpálya, a sakkjátszmák lehetséges pozícióit pedig érvényes lépések kötik össze.

Néha fontos, hogy egy gráfot teljesen bejárjunk, akár nagy gráfok esetén is, amelyekben sok csomópont és él van. Szükség van egy olyan stratégiára, amely biztosítja, hogy valóban minden csomópont elérhető és nem hagyunk ki egyetlen csomópontot sem.

Az ebben a hódfeladatban alkalmazott stratégiát az informatikában *mélységi bejárásnak* nevezik. Miután az algoritmus teljes végrehajtása után a hód visszatér arra a szigetre, ahonnan elindult és a szigetcsoport a következőképpen néz ki.



A végrehajtandó algoritmust úgynevezett pszeudokódban írtuk le.



A pszeudokód a természetes nyelv és a programozási nyelv keveréke. Az utasításokat úgy fogalmazzuk meg, hogy azok érthetőek legyenek. A behúzásokat (a Python programozási nyelvhez hasonlóan) arra használják, hogy kifejezzék, mely utasítások tartoznak össze.

WEBOLDAL

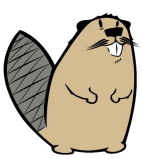
Gráf – Wikipédia

Mélységi keresés – Wikipédia

Pszeudokód – Wikipédia

KULCSSZAVAK

Gráf, Mélységi bejárás, Mélységi keresés, Pszeudokód



Lufigép (2024-DE-06a)

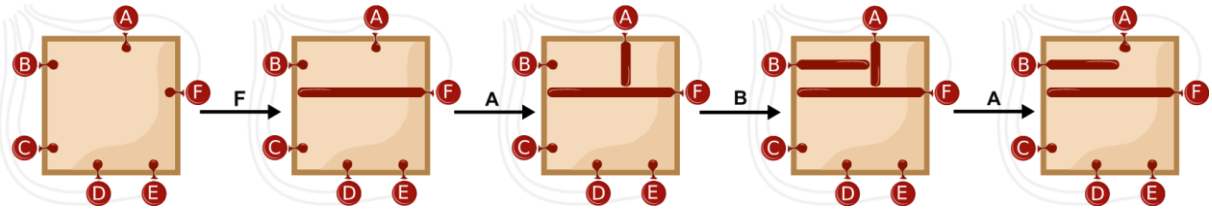
KADÉT - NEHÉZ
JUNIOR – KÖZEPES
SENIOR – KÖNNYŰ

A lufigéppel képeket készíthetünk egy négyzet alakú keretben lévő lufik felfújásával.

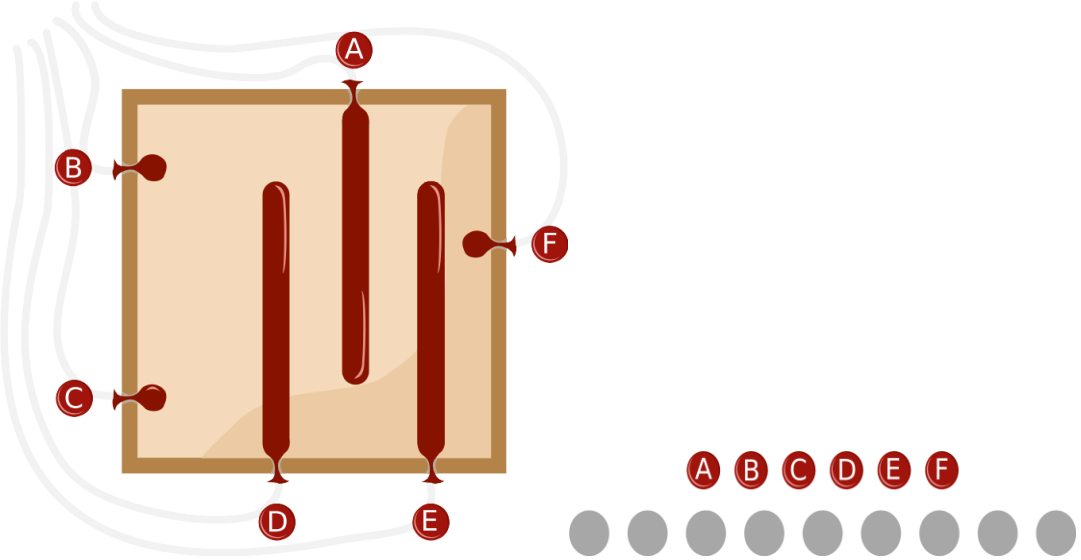
A lufikat A, B, C, D, E és F betűkkel jelöltük. A gép a betűket balról jobbra haladva egymás után olvassa be. Amikor beolvas egy betűt, a következőket teszi:

- Ha az aktuális betűhöz tartozó lufi üres (nincs felfújva), akkor felfújja addig, amíg vagy hozzá nem ér egy másik lufihoz, vagy nem éri el a keret túlsó szélét.
- Ha az aktuális betűhöz tartozó lufi fel van fújva, akkor leereszti.

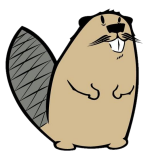
Például: ha kezdetben az összes lufi üres, és a gép az F, A, B, majd A betűt kapja, akkor a következőket fogja tenni:



Kezdetben minden lufi üres, és a gép kilenc betűt olvas be, balról jobbra. Az eredményt az alábbi kép mutatja:

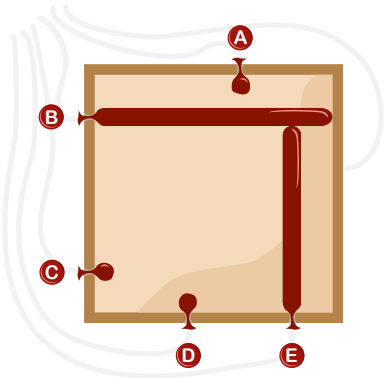


Húzd be a betűket a kilenc helyre a megfelelő sorrendben! Egy betűt többször is használhatsz.



A helyes megoldás: BEBCACBDB vagy BECBACBDB vagy BEBCABCDB vagy BECBABCDB

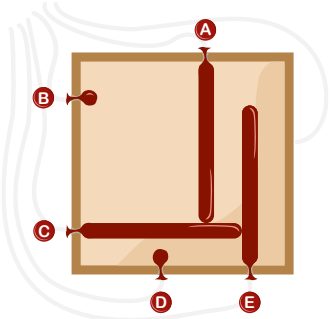
Okoskodjuk ki az egyik megoldást.



A képről leolvasható, hogy B-t E előtt fel kell fújni, hogy B blokkolni tudja az E teljes felfújását. Ezeket elvégezve a helyzetet a jobb oldali kép mutatja.

A B-t ezek után le kell ereszteni, mert különben blokkolná A felfújását. Ez azt is megmutatja, hogy nem kezdhettük volna az A felfújásával, hiszen akkor a B-vel nem tudtuk volna E-t megállítani.

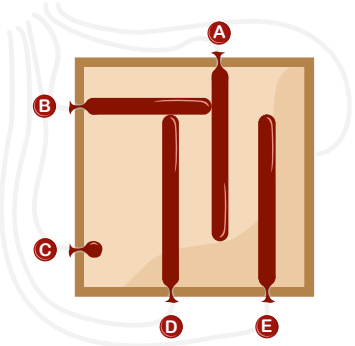
Tehát BEB az első 3 betűnk.



Az A sincs teljesen felfújva, tehát blokkolnunk kell, mégpedig a C-vel. Tehát a CA a következő lépés.

Hasonlóan az előző hármashoz, a D miatt, a C-t le kell ereszteni.

Azaz a BEBCAC lépéseknél tartunk.



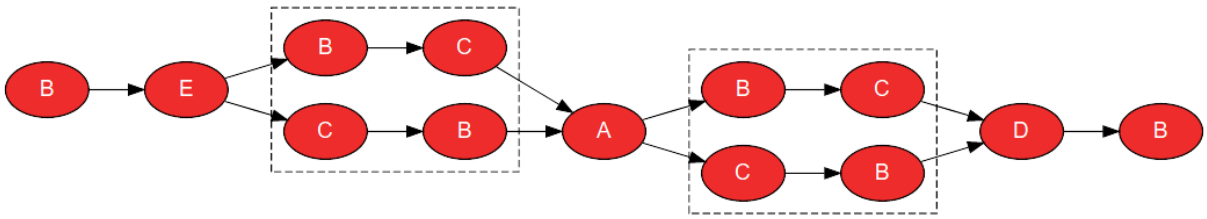
Ezek után a D-t szeretnénk felfújni, de az sem ér el a keretig, így előbb a B felfújásával blokkolnunk kell.

Így a BEBCACBD sornál tartunk.

Ezek után már csak a B „eltüntetése” hiányzik a képről, ezért azt le kell eresztenünk.

Az eredmény tehát BEBCACBDB.

Az összes helyes válasz megtalálásához hasznos, ha a problémát ábrázoljuk, a felfújási sorrendeket figyelve. Jelöljük az egyes lufikat a betűkkel és az egyikből a másikba mutató nyíl jelezze, hogy az egyiket előbb kell felfújni, mint a másikat. Például a B-ből E-be mutató nyíl azt jelzi, hogy a B lufit az E lufi előtt kell felfújni.



Két olyan eset van, amikor a B és a C sorrendje felcserélhető (ezeket a szaggatottan jelölt téglalapok jelzik).

Minden helyes válasz (a felfújható lufik sorrendje) egy lehetséges útvonal az ábránkban. A feladatban négy lehetséges útvonalunk van, tehát négy lehetséges helyes megoldás létezik.

Az is leolvasható, hogy a felfújandó lufiknak (itt A, D és E) páratlan sokszor kell megjelennie, míg a blokkoláshoz használható, a végén üres lufiknak (itt B, C és F) egyszer sem, vagy páros sokszor kell megjelenniük.

MIÉRT INFORMATIKA?

Ebben a hódfeladatban használt betűsorozat egy számítógépes program, amely egy gépet vezérel. Minden egyes betű egy utasítás, amely a gépet egy lufi felfújására vagy leeresztésére utasítja. Mint a legtöbb számítógépes program esetében, az utasítások sorrendje döntő fontosságú. Például a BE sorozat más képet eredményez, mint az EB sorozat. Az informatikában ezt szekvenciának nevezik.

A megoldásnál használt ábra egy gráf, amiben a megelőzési kapcsolatok a nyilak. Ez irányított (azaz fontos, melyik csomópontból mutat melyikbe) és nem tartalmaz ismétlődő köröket (a nyilak mentén nem lehet visszatérni egy kiindulási ponthoz).

Az ilyen gráfoknak számos alkalmazása van az informatikában és más területeken is, például a feladatütemezés, a verziószabályozás vagy az általános projekttervezés területén.

Az irányított, körmentes gráfok feldolgozására pedig gyors algoritmusok léteznek.

WEBOLDAL

Algoritmus – Wikipédia

Strukturált programozás – Wikipédia

Gráf – Wikipédia

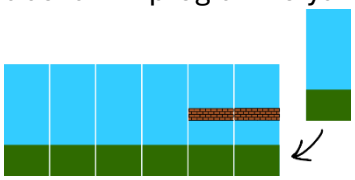
KULCSSZAVAK

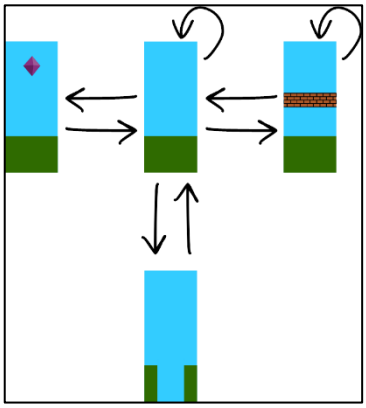
Algoritmus, Szekvencia, Irányított gráf

Szuperhód (2024-DE-07)

KADÉT – NEHÉZ
JUNIOR – KÖZEPES
SENIOR – KÖNNYŰ

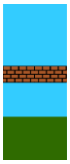
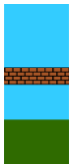
A Szuperhód számítógépes játékban a háttér lapkák sorozatából áll. A program folyamatosan új lapkát helyez a sorozat jobb oldalára, és ezzel egyidejűleg eltávolít egy lapkát a bal oldalról. Így a program a mozgás illúzióját kelti.





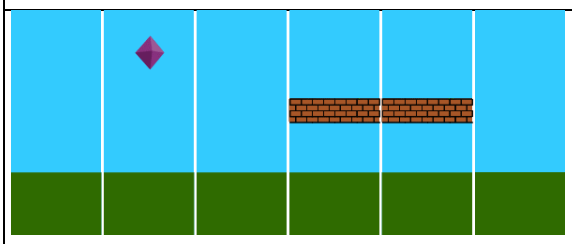
A program az alábbi ábra segítségével választja ki a következő új lapkát:

megkeresi az előző lapkát, és ellenőrzi az onnan kiinduló nyilakat. Ezután véletlenszerűen kiválaszt egy olyan lapkát, amelyre az egyik nyíl mutat.

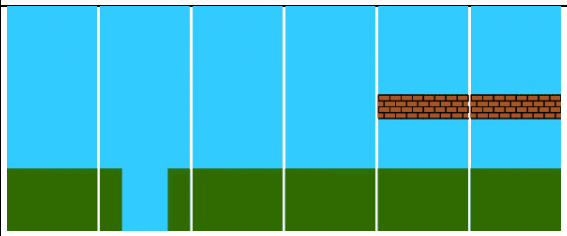
Például a  lapka után a program vagy ismét a  lapkát vagy a  lapkát választhatja.

Az alábbi képek egyike NEM érvényes Szuperhód háttér. Melyik?

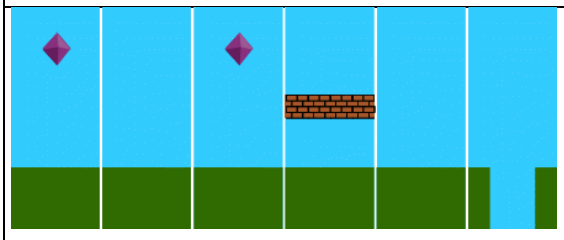
A)



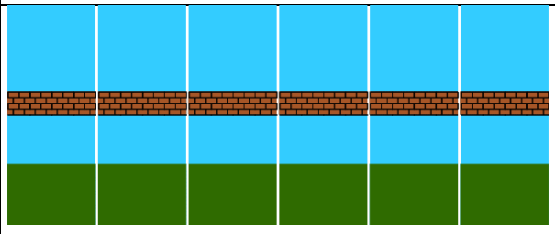
B)



C)

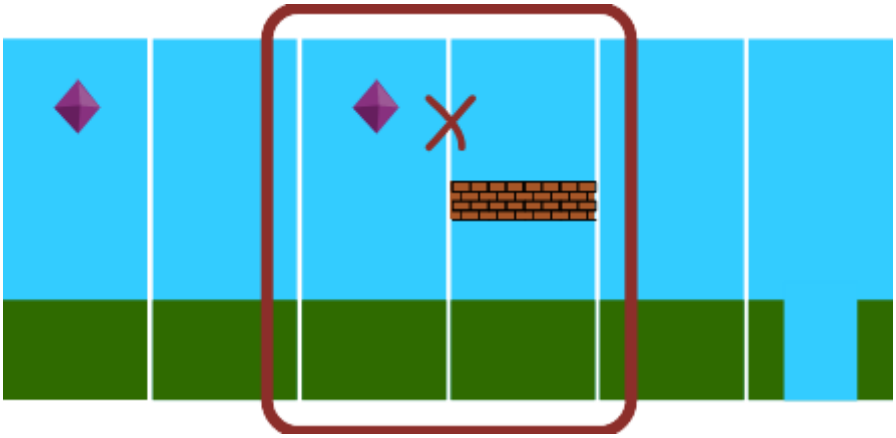


D)



A helyes válasz a C)

A C válaszban szereplő háttér nem teljesen egyezik a szabályok szerint építhető ábrával.



Egy gyémántot (rombuszt) tartalmazó lapkát csak egy lap ( )követhet, ami itt nem így van.

Többféleképpen is meg lehet találni a megoldást erre a feladatra. Egy egyszerű stratégia az, hogy vesszük az adott háttérképet és összehasonlítjuk az egyes lapkákat az ábrával, megnézzük, hogy a sorrend érvényes-e. Ennek egy gyorsabb változata, ha azokkal a lapkákkal kezdjük, amelyekből csak egy nyíl indul.

MIÉRT INFORMATIKA?

Egyes számítógépes játékok - például a végtelen szaladgálós, ugrálós (úgynevezett *platform) játékok* - háttere vízszintesen mozog, azt az illúziót keltve, hogy a játékos egy fantáziavilágban halad előre. Néha a háttér nem egy állandó kép, hanem a játék automatikusan hozza létre.

Ezt nevezik procedurális generálásnak. Általában a kisebb elemeket véletlenszerűen kombinálják, hogy sokféle hátteret hozzanak létre. Az elemek kombinációja azonban nem lehet teljesen véletlenszerű, hanem követnie kell bizonyos szabályokat, hogy elkerülhetők legyenek olyan helyzetek, amelyekkel a játékos nem tud mit kezdeni. A feladatban ezeket a szabályokat egy lapkákból és nyilakból álló gráf ábrázolja. Ezeket irányított gráfoknak is nevezik.

Az elemeket csomópontoknak, a nyilakat pedig irányított éleknek nevezzük. Irányított gráfokat állapotmodellezésre is használnak.

A véges automaták egy lépéssel tovább mennek: állapotokkal, kezdőállapottal, végállapotokkal és az egyik állapotból a másikba való átmenetre szabályokkal rendelkeznek. A véges automatákat gyakran használják modellezésre az informatikában.

WEBOLDAL

Procedural generation - Wikipedia

Platformjáték – Wikipédia

Nemdeterminisztikus véges állapotú gép – Wikipédia

KULCSSZAVAK

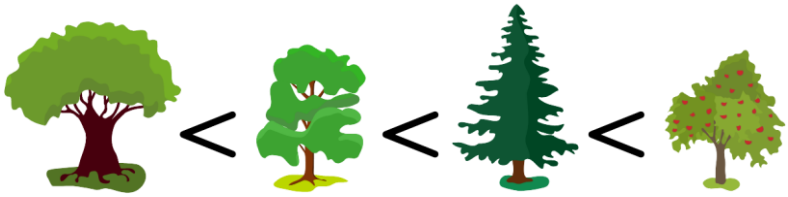
Állapotautomata, Platformjáték fejlesztés

Túraterv az erdőben (2024-DE-08)

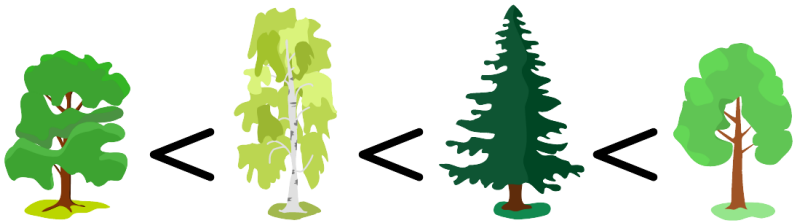
BENJAMIN - NEHÉZ
KADÉT – KÖZEPES
JUNIOR – KÖNNYŰ

Bea kirándulásokat szervez a közeli erdőbe, és mesél a meglátogatott fákról. Korábbi túráiból tudja, mely fák különösen népszerűek a vendégek körében.

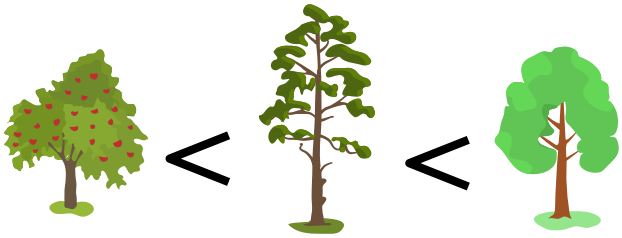
Első túra



Második túra

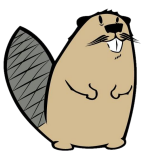


Harmadik túra



Itt az `fa1<fa2` azt jelenti, hogy az `fa1` kevésbé népszerű, mint a `fa2`.

Bea új túrát tervez, melyen a legnépszerűbb fákat akarja megmutatni, a korábbi túrák népszerűségi sorrendjét követve, a népszerűbbeket később.



Például ha Bea a  és a  fákat szeretné megmutatni, akkor előbb kell a  -t, mint a  -t, mert a harmadik túrán  kevésbé volt népszerű, mint a  .

A következő túráján Bea be akarja mutatni a  ,  ,  ,  és  fákat.

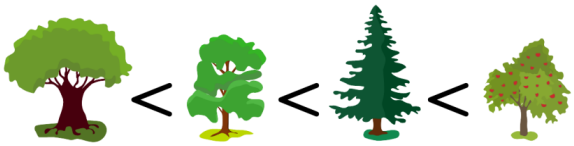
Mi a helyes sorrend a tervezett túrán?

Segíts Beának és húzd a fákat a megfelelő (népszerűségi) sorrendbe!

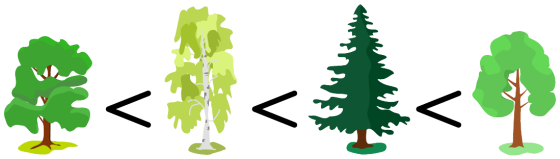
A megoldás:

A megfelelő sorrend nem mondhat ellent a fák népszerűségének a korábbi túrákon:

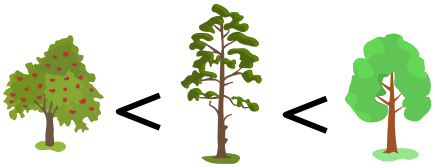
Első túra



Második túra

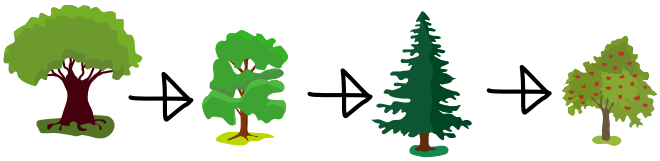


Harmadik túra

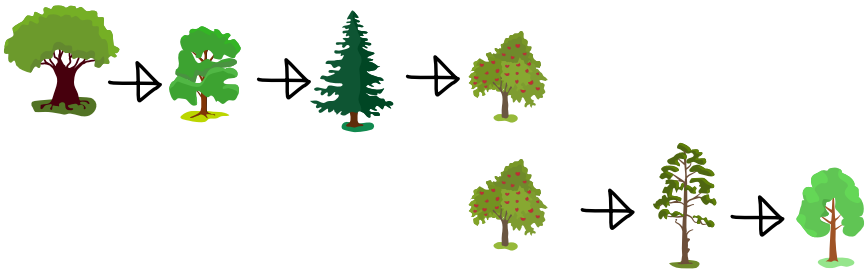


Próbáljuk meg a három túrát összekapcsolni, és a túrákat, a fák népszerűségét együtt ábrázolni. Ezen a két fát mindig egy nyíl köti össze, ha az egyik fát előbb kell meglátogatni (népszerűbb), mint egy másikat.

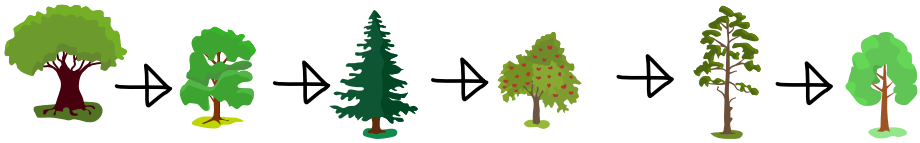
Az első túra ábrája:



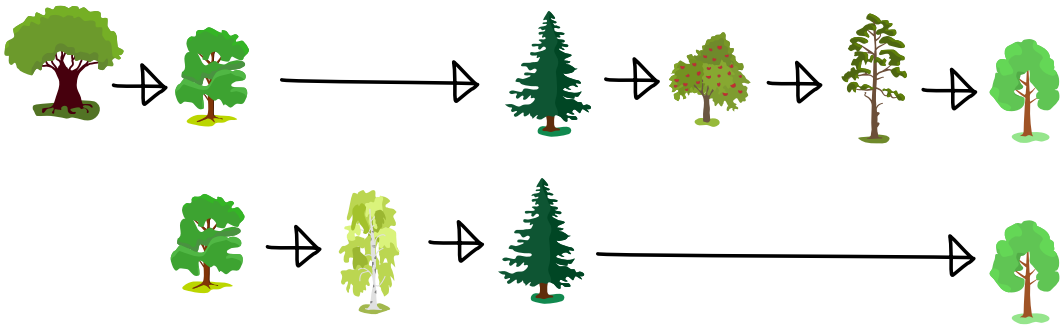
Ez összekapcsolható a harmadik túrával, mivel mindkettő tartalmazza a  fát.



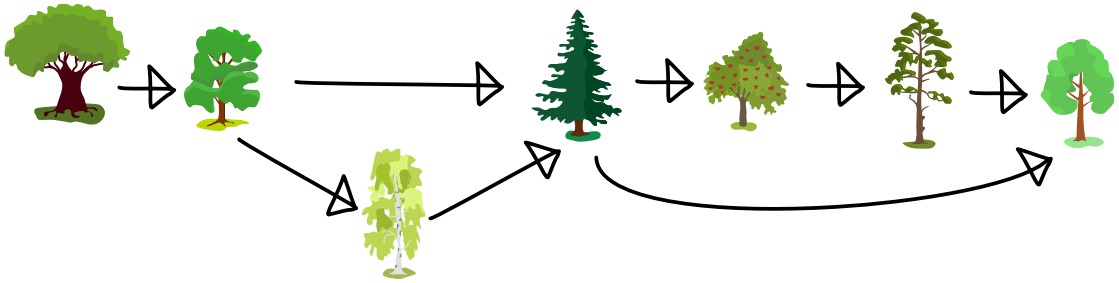
azaz:



Végül tegyük be a második túra adatait is. Szerencsére csak egy új fa jelenik meg (ami nem volt az első és a harmadik túrán):



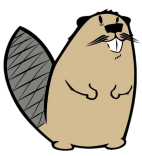
Ezt összerakva megkapjuk a következő ábrát:



Az ábra segítségével most már nagyon könnyen megoldhatjuk a feladatot: A nyilak mentén balról jobbra haladva meglátogatjuk az ábrán szereplő fákat, és így felépítjük a séta során figyelembe veendő sorrendet. Vegyük észre, hogy nem minden fát kell bevonni a sétába.

MIÉRT INFORMATIKA?

Ebben a feladatban fákat kell növekvő népszerűségi sorrendbe rendezni.



Néhány fáról van páronkénti összehasonlításból információnk. Más párok esetében, mint például  és  nincs közvetlen összehasonlítás.

De végül ezek a fák is összehasonlíthatók lesznek, mivel az összehasonlíthatóságnak két fontos tulajdonsága is van:

1. tranzitív, azaz ha az fa1<fa2 és a fa2<fa3, akkor az fa1<fa3.

2. egyértelmű, tehát ellentmondásmentes, azaz a már megadott három túra nem mond ellent egymásnak. Például probléma lenne, ha az egyik túrán a fa1<fa2 és ugyanakkor egy másik túrán a fa2<fa1.

Ezeknek a tulajdonságoknak köszönhetően a feladat matematikai értelemben egyértelműen meghatározott sorrendet eredményez. Kezdetben úgy tűnhet, hogy ez a sorrend részleges, mivel nem látjuk az összes fát páronként összehasonlítva. Valójában a feladatban szereplő fák teljesen rendezett sorrendbe tehetők (ezt *topologikus* rendezésnek nevezzük). Van egy viszonylag egyszerű algoritmus egy ilyen topologikus rendezés meghatározásához. Ehhez azonban az kell, hogy az eredeti összehasonlításokban ne legyenek ellentmondások.

Közelebbről megvizsgálva még azt is megállapíthatjuk, hogy a fák népszerűsége egy teljes rendezés, és a tranzitivitásnak köszönhetően egyetlen helyes megoldást kapunk az összes fa topologikus rendezésére.

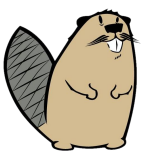
WEBOLDAL

Topologikus sorrend – Wikipédia

Tranzitív reláció – Wikipédia

KULCSSZAVAK

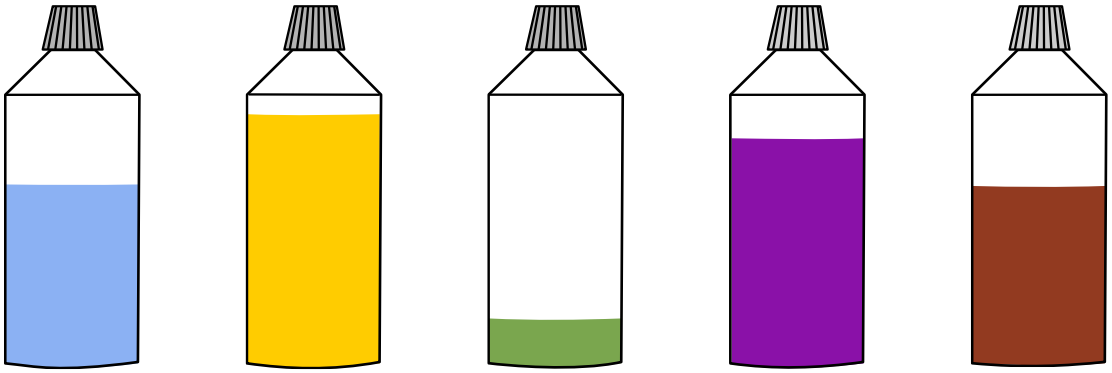
Gráf, Topologikus rendezés, Tranzitivitás



Kifestő (2024-FI-01)

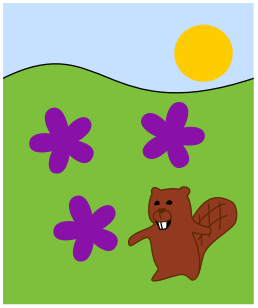
KISHÓD – KÖZEPES
BENJAMIN – KÖNNYŰ

Bea fest egy képet. Öt teli tubus festékkel kezd. Több festékre van szüksége a kép nagy területein, mint a kisebb területeken. Amikor Bea befejezi a festést, a tubusokban még mindig ennyi festék van:

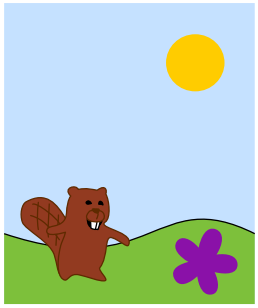


Melyik képet festhette Bea?

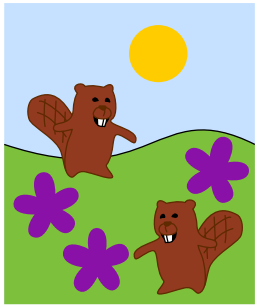
A)



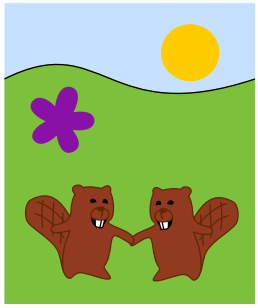
B)



C)



D)



A helyes válasz a D)

A helyes válasz megtalálásához hasonlítsuk össze a képen látható színes területeket a festékes tubusok tartalmával. A színes tubusok alapján láthatjuk, hogy a zöldet használta fel a legtöbbet, és a kéket a második legtöbbet. Ez egyezik az A, C és D képekkel. Ezeken a képeken a zöld terület a legnagyobb, a kék terület pedig a második legnagyobb. A B képet kizárhatjuk, mert a kék terület sokkal nagyobb, mint a zöld terület.

A színes tubusok azt is megmutatják, hogy Bea közel ugyanannyi sárgát használt, mint lilát. Az A és C képen a teljes lila terület (3 virág) körülbelül háromszor akkora, mint a sárga terület (nap). Csak a D képen a sárga és a lila terület (nap és virág) körülbelül egyforma méretű. A barna szín használata szintén a D képhez illeszkedik: Bea ugyanannyi barnát használt, mint kéket, és az összes barna terület (2 hód) körülbelül ugyanolyan méretű, mint a kék terület.

Bea tehát csak a D képet festhette.

MIÉRT INFORMATIKA?

Ebben a hódfeladatban a színek használatát vizsgálva találtad meg a helyes képet. Például a következő kérdéseket tetted fel magadnak: Milyen színeket használnak? Melyik színt használják a legtöbbet? Mely színeket használják kevéssé? Egy színt többet használnak, mint egy másik színt?

E kérdések megválaszolásához meg kellett különböztetned a különböző területeket a képen, meg kellett becsülnöd a területek méretét, össze kellett hasonlítanod a területértékeket és döntéseket kellett hoznod.

Az ilyen elemzéseket számítógépes programok automatikusan is el tudják végezni. A számítógépes programok segíthetnek a mobiltelefonon tárolt fényképek megtalálásában. A színek vizsgálatával egy számítógépes program képes megkülönböztetni a tájképi fotókat, amelyeken felül sok kék, alul pedig sok zöld van, a portréfotóktól. A műholdképek színének vizsgálatával a számítógépek meg tudják állapítani, hogy egy terület mennyire sűrűn beépített, vagy hogy egy természetvédelmi területen illegálisan irtottak-e erdőt.

Az informatikában létezik egy fontos terület, amely a képek automatikus elemzésével foglalkozik: A számítógépes látás.

WEBOLDAL

Digitális képfeldolgozás – Wikipédia

Hisztogram – Wikipédia

Gépi látás – Wikipédia

KULCSSZAVAK

Színhisztogram, Számítógépes képfeldolgozás, Gépi látás

Téglafal (2024-FI-03)

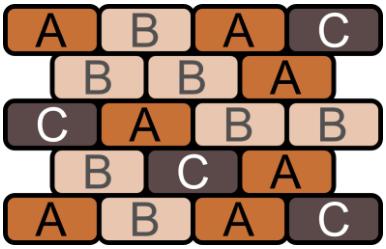
SENIOR – NEHÉZ

Bob háromféle téglából épített falat. Sajnos rossz helyre építette, ezért át kell helyeznie.

Ezt téglánkként teszi meg a következőképpen:

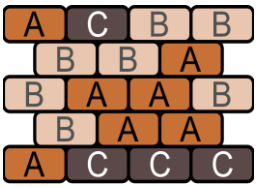
- levesz egy olyan téglát a falról, amin nincs másik tégla;
- hozzáteszi a téglát az új falhoz, vagy a földre, vagy az új falon két másik téglára, de sohasem valamelyik tégla alá.

A következő ábrán Bob eredeti fala látható.

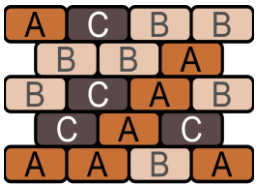


Az alábbi lehetőségekből válaszd ki az összes olyan falat, amelyet Bob fel tud építeni.

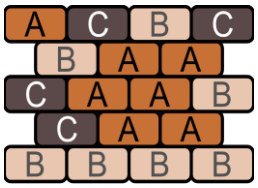
A)



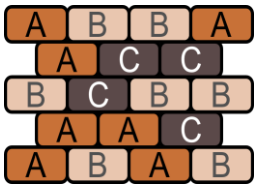
B)

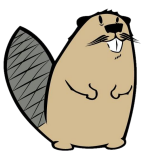


C)



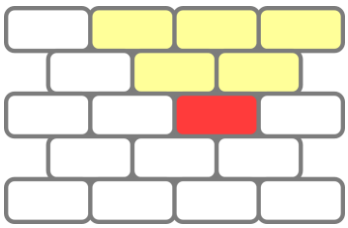
D)



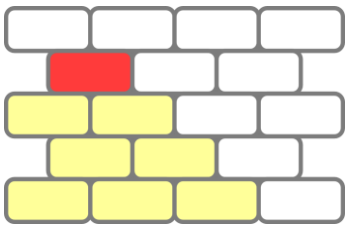


A helyes válasz D)

Először is, tegyünk egy alapvető megfontolást. A bal oldali képen a piros tégla egy megfordított háromszög csúcsa, amelyet a hat sárga tégla alkot. Ezek pontosan azok a téglák, amelyeket a piros tégla előtt át kell helyezni. Minden tégla az első falban egy ilyen „előtte levő” háromszög része. A jobb oldali képen a sárga téglák azok, amelyeket a piros tégla előtt az új falra kell helyezni; ez az „előtte rajta lévő” háromszög, amely minden téglához tartozik.



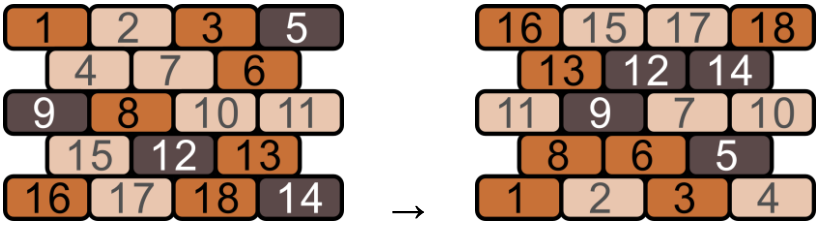
első fal



új fal

Ha a téglákat abban a sorrendben számozzuk, ahogy át vannak helyezve, akkor a következő két áthelyezési feltétel érvényes: Minden tégla az első falban nagyobb számú, mint az „előtte rajta levő” háromszög többi téglája. Hasonlóképpen, minden tégla az új falban nagyobb számú, mint a „előtte rajta lévő” háromszög többi téglája.

Az alábbi kép azt mutatja, hogy a D válasz fala hogyan hozható létre az első falból Bob módszere szerint. A téglák abban a sorrendben vannak számozva, ahogy át lettek helyezve, és az áthelyezési feltételek érvényesek:

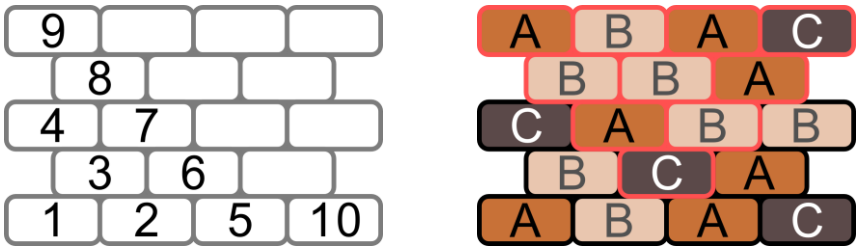


Most meg kell mutatni, hogy az A, B és C válaszok fala nem hozható létre Bob módszere szerint.

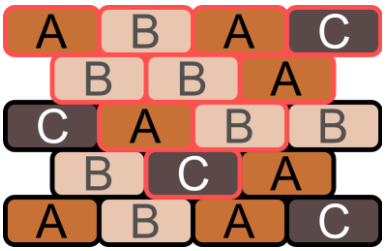
A legkönnyebben a **C válasz** esetében cáfolható a szabály: Az alsó sorban négy B tégla található. Egy második B tégla legkorábban a harmadik téglaként helyezhető át. A második áthelyezett tégla tehát nem lehet B tégla, de kötelezően az új fal alsó sorába kell kerülnie. Tehát lehetetlen, hogy az új fal alsó sorában négy B tégla legyen.

Az **A válasz** falában három C tégla található az alsó sorban. Az új fal alsó sorában a harmadik C legfeljebb a tizedik áthelyezett tégla lehet (bal oldali kép). Ha a negyedik sorban levő C téglát akarjuk elvenni az első falból, akkor Bobnak először az „előtte rajta lévő” há-

romszög összes kilenc tégláját át kell helyeznie (jobb oldali kép). Ezért ez a C tégla legkorábban a tizedik téglaként helyezhető át, és így a második C tégla lesz az új falban. Egy harmadik C tégla tehát legkorábban a tizenegyedik téglaként helyezhető át. Tehát lehetetlen, hogy az új fal alsó sorában három C tégla legyen.



Most nézzük meg a **B válasz** falát: A B tégla, amely negyedikként kerül az új falra, a lenti képen jelölt pozíciókban lehet. Attól függően, hogy ezek közül melyik pozíciót nézzük és hogy melyik három B téglát helyeztük el előtte, legalább 5 A téglának vagy legalább 3 C téglának kellett korábban az új falra kerülnie. Bob első falában a bal oldali második sorból a második tégla az a B tégla, amely negyedikként helyezhető át, hogy a lehető legtöbb másik tégla kerüljön előtte. Őt előbb a sorában lévő más téglák és az alsó sor téglái elé kell helyezni. Így tehát legfeljebb 4 A tégla és 2 C tégla helyezhető el előtte. Ez ellentmondásban áll a fent említett megfigyeléssel a B válasz falára, ezért ez nem lehet az új fal.



MIÉRT INFORMATIKA?

Mivel Bob egy megadott módszer szerint dolgozik, nem tudja a téglákat bármilyen sorrendben újra rakosgatni. A téglák helyzete az első falban sok függőséget eredményez két tégla között a sorrend tekintetében. Például a felülről második sorban balra lévő téglához a felső sorban balra lévő téglát és a tőle jobbra lévő téglát is át kell rakni, mielőtt az átrakható lenne. Viszont ezt a téglát is át kell rakni a harmadik sorban balra lévő tégla és a tőle jobbra lévő tégla előtt. Ez egy általános „újratelepítési sorrendet” hoz létre. Sok tégéapár esetében meghatározza, hogy az egyik téglát a másik előtt kell újra rakni - de nem minden lehetséges téglapár esetében.

Az informatikában gyakran kell olyan sorrendekkel foglalkoznunk, amelyek éppoly hiányosak, mint ebben a hód feladatban. Ha egy gyártási folyamat munkalépéseinek időrendi sorrendjét számítógéppel tervezzük meg, akkor bizonyos munkalépés párok esetében ismert, hogy az egyiket a másik előtt kell végrehajtani - de nem minden pár esetében. Amikor egy új szoftvercsomagot telepítünk egy számítógépre, általában több más csomagot is előbb kell telepíteni, és e csomagok egyes párjaira ismert a telepítési sorrend - de nem az összesre. Ennek ellenére a végén kell lennie egy olyan sorrendnek, amelyben minden sorrendben történik.

Az objektumok közötti függőségek általában függőségi gráf formájában fejezhetők ki. A lehetséges feldolgozási sorrend – ebben a feladatban a téglák eltávolításának és elhelyezésének sorrendje – megfelel a tárgyak topológiai sorrendjének.

WEBOLDAL

Topologikus sorrend – Wikipédia

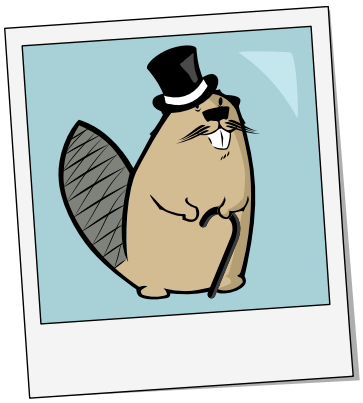
KULCSZSAVAK

Topológia, Gráf

Alfréd a bálban (2024-HU-02)

BENJÁMIN – NEHÉZ
KADÉT – KÖZEPES
JUNIOR – KÖNNYŰ

Amikor a híres Alfréd hód megjelent egy bálban, azonnal le is fényképezték.



Négy influenszer is írt erről, de nem minden hír igaz, ami a közösségia médiában megjelenik.

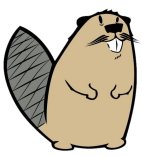
Melyik influenszer írt igazat az alábbiak közül?

A) Alfrédon cilinder volt és nem volt nála sétapálca.

B) Alfréd vagy nyakkendőt viselt vagy volt nála sétapálca.

C) Alfréd cilindert és csokornyakkendőt viselt.

D) Alfréd vagy csokornyakkendőt viselt vagy nem volt nála sétapálca sem.



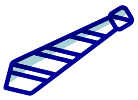
A helyes válasz a B)

Ebben a feladatban egy képpel kapcsolatos állítások igazságtartalmát (IGAZ vagy HAMIS) kell meghatározni. Minden állítás két rész-állításból áll, amelyeket vagy az „és” vagy a „vagy” kapcsol össze.

Egy „és”-sel összekapcsolt állítás csak akkor IGAZ, ha mindkét rész-állítás IGAZ.

Egy „vagy”-gyal összekapcsolt állítás csak akkor IGAZ, ha legalább az egyik rész-állítás IGAZ. Nézzük meg az állításokat:

A) Alfrédon cilinder  volt (IGAZ) **és** nem volt nála sétapálca  (HAMIS). → HAMIS az állítás

B) Alfréd vagy nyakkendőt  viselt (HAMIS) **vagy** volt nála sétapálca  (IGAZ). → IGAZ

C) Alfréd cilindert  (IGAZ) **és** csokornyakkendőt  (HAMIS) viselt. → HAMIS

D) Alfréd vagy csokornyakkendőt  viselt (HAMIS) **vagy nem** volt nála sétapálca  sem (HAMIS). → HAMIS

MIÉRT INFORMATIKA?

A logikai állítások logikai műveletek (operátorok) segítségével összekapcsolhatók és lehetővé teszik összetett döntések meghozatalát.

A logikai operátoroknak különböző típusai vannak. Ebben a feladatban 2 típust használunk:

- ÉS: Az összekapcsolt állítás csak akkor IGAZ, ha mindkét rész-állítás IGAZ.
- VAGY: Az összekapcsolt állítás akkor IGAZ, ha legalább az egyik rész-állítás IGAZ.

Az informatikában sok helyen használunk logikai állításokat és műveleteket. Például az e-mail programok spamszűrői logikai operátorokkal dolgoznak: a program csak akkor helyez át egy e-mail-t a spam mappába, ha az e-mail ismeretlen feladótól érkezik ÉS a tárgysorban bizonyos szavak szerepelnek. Tehát a program csak akkor mozgatja a levelet, ha mindkét részleges állítás IGAZ.

WEBOLDAL

Logikai művelet – Wikipédia

Boole-algebra (informatika) – Wikipédia

KULCSSZAVAK

Logikai műveletek, Bool algebra

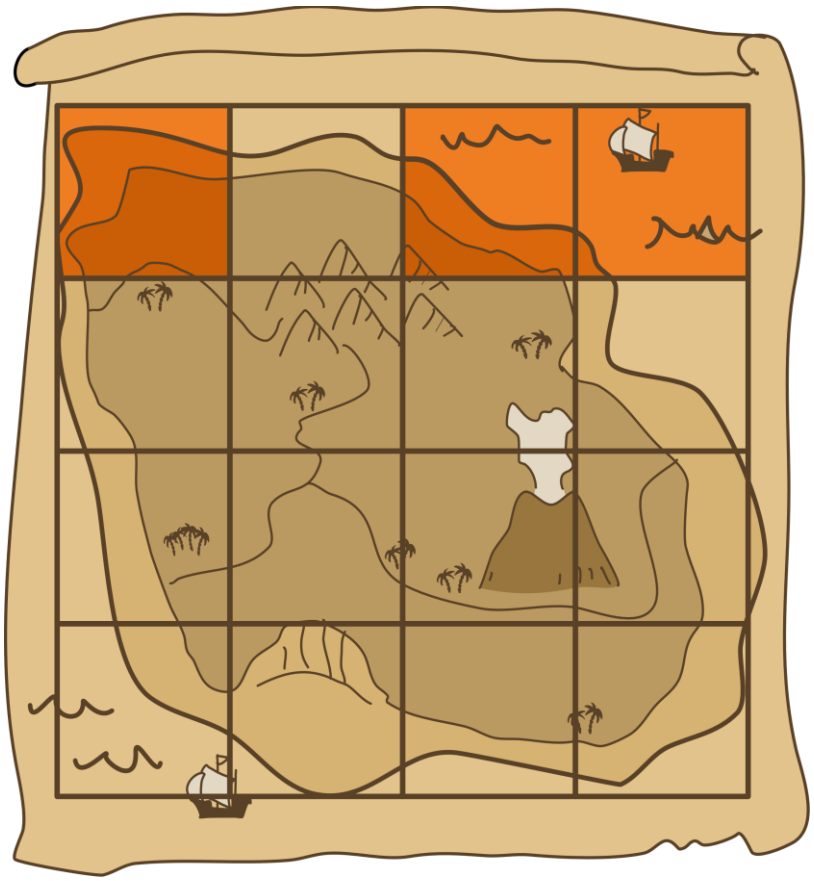
Kincsvadász (2024-HU-03)

KADÉT– NEHÉZ
JUNIOR – KÖZEPES
SENIOR – KÖNNYŰ

Hódszakáll, a kalóz, kincset keres egy elhagyatott szigeten. Van egy térképe a szigetről, ami 16 négyzetre van felosztva.

Szószátyár, Hódszakáll papagája, tudja, melyik négyzetben van a kincs. Hódszakáll egyszerre megkérdezheti Szószátyárt akárhány négyzetről. Szószátyár ekkor megmondja neki, hogy a kincs bármelyik megkérdezett négyzetben van-e vagy nincs.

Például Hódszakáll megkérdezi a három négyzetet, amelyek a képen jelöltek. Ha Szószátyár azt mondja, hogy „igen”, Hódszakáll tudja, hogy a kincs valahol ezen a három négyzeten belül van – de azt nem, hogy pontosan melyikben.



Hódszakáll szeretné a lehető leggyorsabban, biztosan megtudni, hogy melyik négyzetben van a kincs.

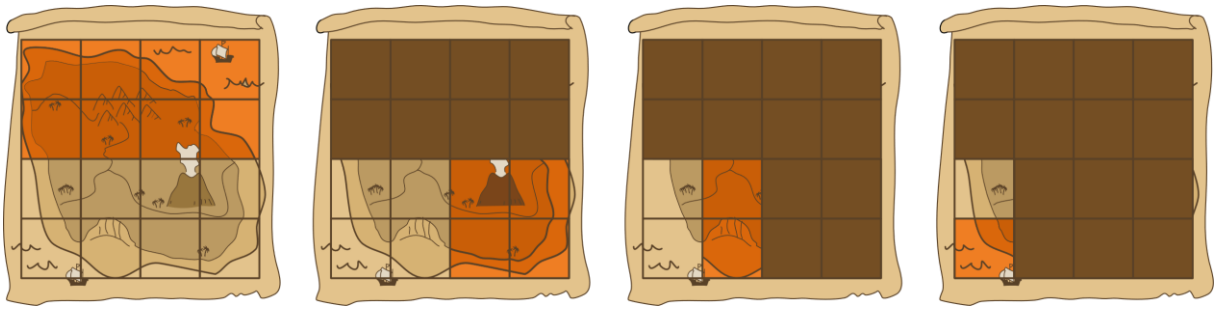
Hányszor kell legalább Hódszakállnak rákérdeznie a négyzetekre, hogy a lehető leggyorsabban megtalálja a kincset?

A helyes válasz: 4

Ahhoz, hogy Hódszakáll négy kérdés után biztosan megtudja, melyik négyzetben van a kincs, a következő stratégiát használhatja: mindig pontosan azokra a négyzetekre kérdez rá, amelyek a korábbi kérdések alapján még szóba jöhetnek, és ezek felét kérdezi le.

Az elején a kincs még mind a 16 négyzetben lehet. Az 1. kérdésben Hódszakáll tehát 8 négyzetet kérdez meg ezek közül. Ha Szószátyár azt mondja, hogy „igen”, a kincs ebben a 8 négyzetben lehet; ha „nem”, akkor a kincs a másik 8 négyzet valamelyikében lehet. A 2. kérdésben Hódszakáll ebből a fennmaradó 8 négyzetből 4-re kérdez rá, a 3. kérdésben 2 négyzetet kérdez meg a fennmaradó 4-ből, a 4. kérdésben pedig 1 négyzetet a maradék kettőből. A 4. kérdésre adott válasz alapján Hódszakáll pontosan tudni fogja, melyik négyzetben van a kincs.

A képek egy lehetséges menetet mutatnak be. Hódszakáll mindig összefüggő felekre osztja a fennmaradó négyzeteket. A kérdezett négyzetek világos pirossal vannak jelölve. Szószátyár mindig „nem”-mel válaszol, és négy kérdés után Hódszakáll tudja, hogy a kincs abban a négyzetben van, amelyik az utolsó képen fehér.



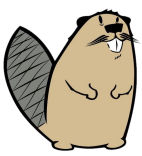
Kevesebb kérdéssel Hódszakáll nem boldogulhat. Ha nem két egyenlő részre osztaná a négyzeteket, a kincs a nagyobb részben is lehetne. Ennek a résznek a lekérdezéséhez legalább ugyanannyi kérdésre lenne szükség, mint egy felezés esetén.

MIÉRT INFORMATIKA?

A módszert, amit Hódszakáll a kincskeresésnél alkalmaz, az informatikában bináris keresésnek hívják. A "bináris" kifejezés a latin „bis” (kétszer) szóból származik. A bináris keresés során egy halmazban az objektum kereséséhez a halmazt egy sor kérdés segítségével folyamatosan félbe osztják, azaz két részre bontják – ezért „bináris”.

Egy halmaz akkor osztható jól két részre, ha az objektumok rendezhetők benne, például méret alapján; ez igaz többek között bármely számhalmazra. Ilyenkor van a halmazban egy középső elem, amit össze lehet hasonlítani a keresett tárggyal. Ha a középső elem nem az, amit keresünk, akkor legalább tudjuk, hogy a halmaz melyik felében van a keresett objektum, és ezt a felét újra binárisan átvizsgáljuk.

Ezzel a módszerrel nagyon gyorsan elérhetünk a keresett objektumhoz. 1.000 objektumnál körülbelül 10 keresési lépésre van szükség, 1.000.000 objektumnál pedig körülbelül 20-ra. Általánosan elmondható, hogy n objektum esetén körülbelül log(n) lépés szükséges; a log függvény a "kettes alapú logaritmus" vagy a logaritmus 2-es alappal. Mivel a



bináris keresés nagyon gyors, gyakran használják számítógépes programokban rendezett adatok keresésére.

Ebben a hódfeladatban a keresési tér a sziget térképén található négyzetek halmaza. A négyzetek rendezhetők például úgy, hogy felülről lefelé, majd balról jobbra számozzuk őket. Ebben az esetben azonban az is működik, ha a négyzetek halmazát, ahol a kincs még lehet, tetszőlegesen felezzük. Csak akkor valamivel nehezebb megjegyezni, hogy a következő keresési lépéshez melyik négyzetcsoport jöhet még szóba, és azt újra fel kell osztani.

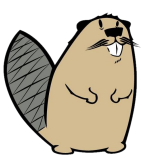
WEBOLDAL

Bináris keresés – Wikipédia

Keresőalgoritmus – Wikipédia (wikipedia.org)

KULCSSZAVAK

Keresés, Bináris keresés, Keresőalgoritmus



Palago (2024-HU-04)

SENIOR – NEHÉZ

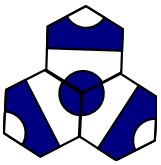
A Palago egy olyan társasjáték, amelyet azonos hatszögletű lapkákkal ketten játszanak. Mindegyik lapka kék és fehér színű, ahogy a képen látható.

Az egyik játékos a fehér, a másik a kék színnel van és a játék célja, hogy a játékosok egy zárt, legalább egy egyenest tartalmazó alakzatot alkossanak a saját színükből.

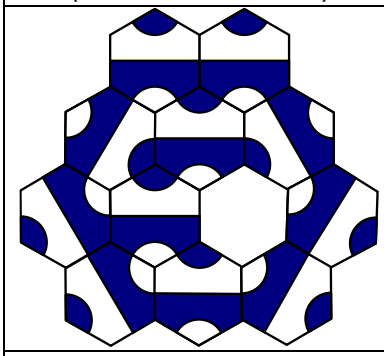
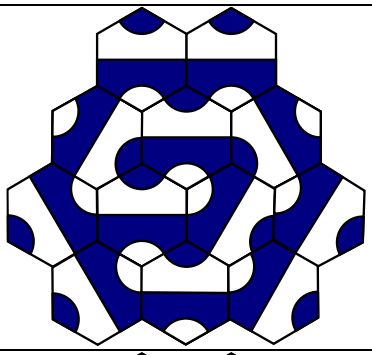
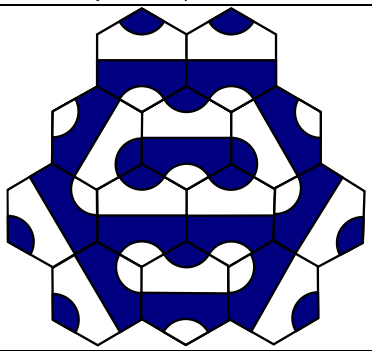
Az első játékos két lapkát helyez egymás mellé úgy, hogy a színek illeszkedjenek (kék csak kékkel folytatódhat a másik lapkán, fehér csak fehérrel). Ezután a játékosok felváltva helyeznek el két-két lapkát, a következő szabályokat követve:

1. A két új lapka közül legalább az egyiket egy már letett lapka mellé kell tenni.
2. A két új elhelyezett lapkának egymás mellett kell lennie, azaz legalább egy élüknek érintkezni kell.
3. Az érintkező élek mentén a színeknek továbbra is illeszkednie kell.

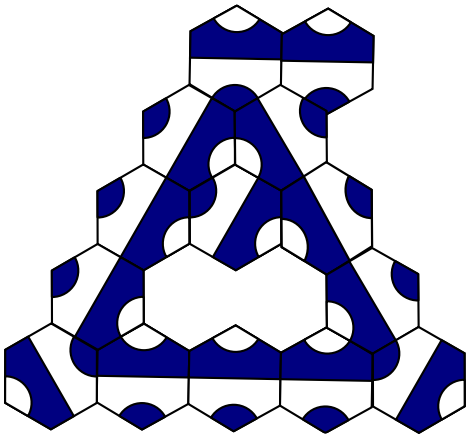
A játékos nyer, ha egy zárt alakzatot tud lerakni legalább egy egyenes oldallal a saját színében. Veszít azonban, ha ugyanakkor a másik színnel is keletkezik egy zárt, legalább egy egyenest tartalmazó alakzat. Ha csak egy lapka szükséges a győzelemhez, a másodikat nem kell lerakni.



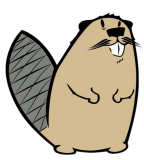
Ebben az esetben a kéknek ugyan van egy zárt alakzata, de nem tartalmaz egyenest. Emiatt a kék nem nyert (még). A fehér sem nyert, hiszen a fehérnek nincs zárt alakzata. Egy másik példa a következő oldalon látható:

Aktuális játékállás (a fehér következik)	A következő lapka lerakása után...	
		... a fehér nyert, mivel egy olyan fehér zárt alakzat keletkezett, amiben van egyenes.
		... a fehér vesztett, mivel nem csak egy fehér, de egy kék zárt (és egyenessel rendelkező) alakzatot is létrehozott.

Ki nyer a következő játékállásban, ha mindkét játékos hibázás nélkül játszik?

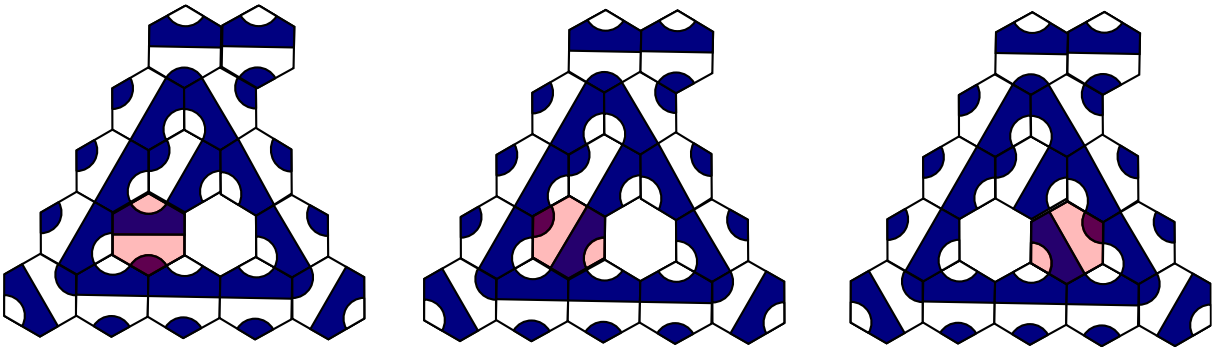


A) Kék nyer.
B) Fehér nyer.
C) Nem tudjuk biztosan megmondani.
D) Attól függ, melyik játékos következik.



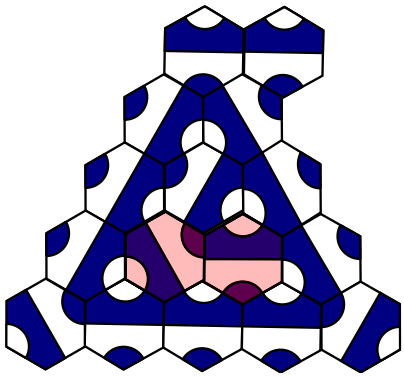
A helyes válasz a B): a fehér nyer.

Amikor fehér a következő lapkát lehelyezi, egyetlen lapkával bezárhat egy fehér alakzatot anélkül, hogy kék alakzatot is bezárna. Erre három lehetősége van:



A szabályok miatt nem kell a második lapkát elhelyeznie.

Ha a kék rakja le a következő lapkát, akkor kitöltheti a lyukat, és mindkét lapka felhasználásával bezárhatja az alakzatot. De akkor a fehérnek is bezárna egy alakzatot, így elveszítené a játékot.



Ha a kék nem zárja be a lyukat, a széleken nem tud a szabályoknak megfelelően máshol zárt alakzatot létrehozni. Azaz a játék folytatódik, a következő fordulóban a fehér megteheti a korábban leírtakat. A kék nem helyezhet csak egy lapkát a résbe, mert a két elhelyezett lapkának össze kell érnie. A kéknek nincs lehetősége arra, hogy a már kijátszott kövek körül alakzatot zárjon, így fehér legkésőbb a következő lépésben nyerhet.

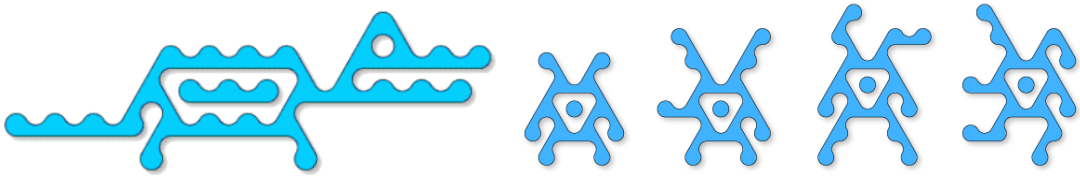
MIÉRT INFORMATIKA?

Egy ilyen többjátékos játékban (Palago) stratégiára van szükséged a győzelemhez, vagy ahhoz, hogy elkerüld a következő lépésekben a vereséget. Fontos, hogy lássuk a következő lehetséges lépéseket és azok kimenetelét, és ez alapján döntsünk.

A legtöbb stratégiai játék esetében ez szisztematikusan elemezhető egy játékfa segítségével. A játékfa gyökere a kiinduló pozíció, és az egyes pozíciókból kiinduló összes lehetséges lépés az ágak. Ily módon egy olyan fa épül fel, amely egy ilyen játékon belül az összes lehetséges játékállapotot reprezentálja. A játékfa felépítésekor megtalálhatjuk azt az utat, amelyet követve nyerhetünk anélkül, hogy az ellenfél előbb nyerne.

Egyes játékok (sakk, Go és Palago) hatalmas vagy akár végtelenül nagy játékfával rendelkeznek, így csak részleges játékfával dolgozhatunk, és a teljes fának csak egy részét tudjuk egyszerre elemezni, például csak a következő 5 lépést. Alternatívaként egy játékhelyzetet kívülről is kiértékelhetünk, és csak az ígéreteseket követhetjük nyomon.

A Palago egy kreatív művészeti kirakójáték is, amelyet Cameron Browne tervezett. Nem csak azért kell Palagóval játszanod, hogy nyerj. Több játékos is megpróbál a téglákból palagóniait (Palagónia, Palago világának polgára) vagy állatokat alkotni:



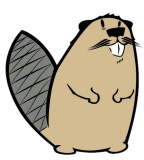
WEBOLDAL

Palago - Wikipedia

Játékelmélet – Wikipédia

KULCSSZAVAK

Játék stratégia, Játékfa, Játékelmélet



Az erdőben (2024-ID-04)

KISHÓD – KÖNNYŰ

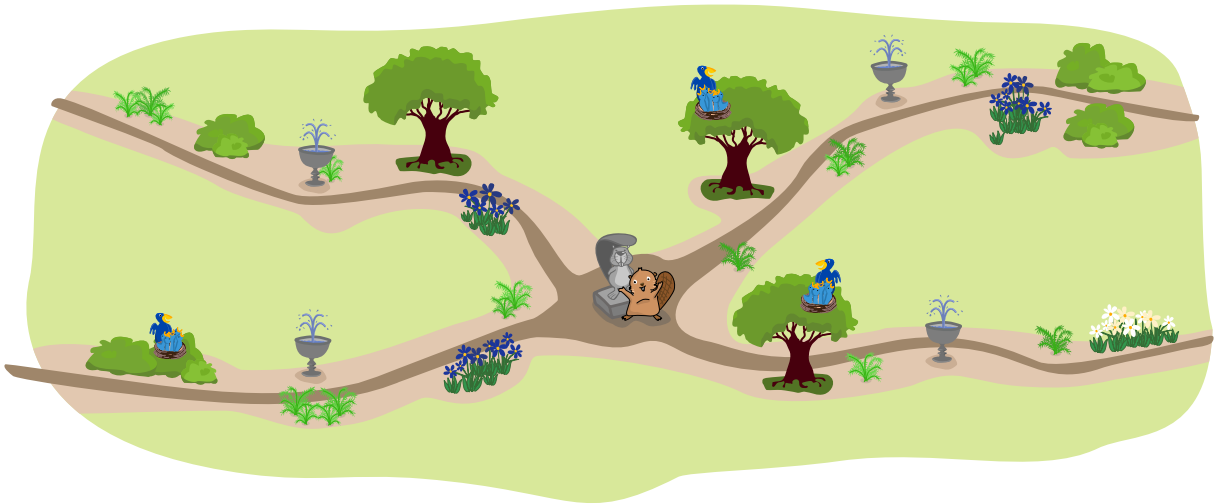
Piroska a parkban a hód szobornál  áll. Az odavezető úton a következőket látta:

először kék virágokat;

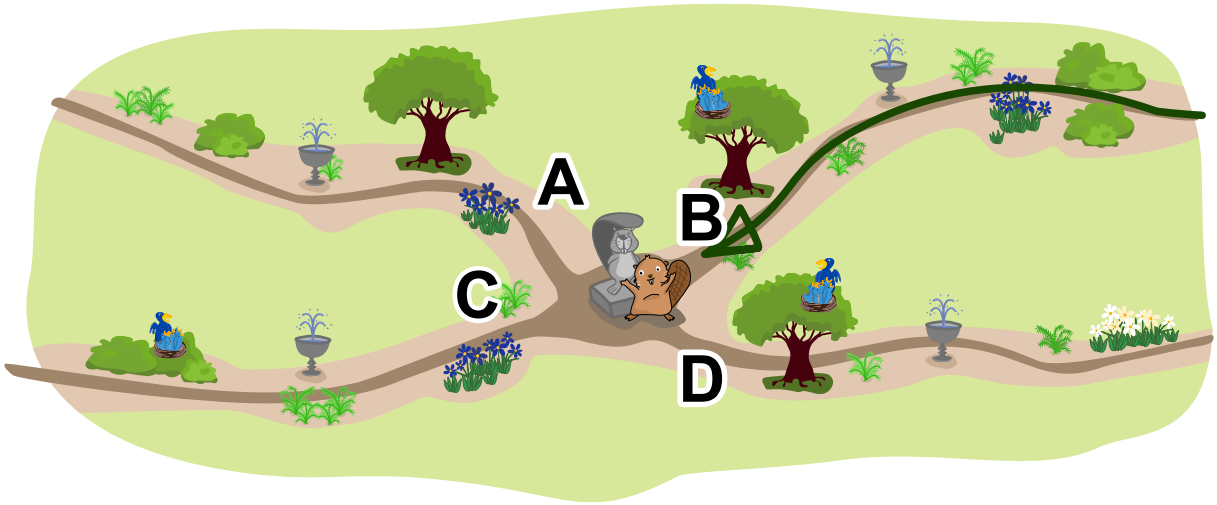
azután egy kis szökőkutat;

végül felfedezett egy madárfészket.

Melyik úton ment el a szoborig?



A helyes válasz a B-vel jelölt út:



Ahhoz, hogy a helyes megoldást megkapjuk, először megvizsgálhatjuk, melyik úton találhatóak meg mindezek, majd azt, hogy a szobortól visszafele melyik úton találhatóak fordított sorrendben.

Az A-val jelölt úton nincs madárfészek, a D-vel jelölten pedig nincsenek kék virágok. A C-vel jelölt úton viszont pont fordított sorrendben szerepelnek, mint ahogy Piroska láthatta azokat, amikor a szoborhoz ment.

A B-vel jelölt úton a megfelelő sorrendben láthatott mindent.

MIÉRT INFORMATIKA?

Ez a feladat egy úgynevezett LIFO (Last In First Out, utolsónak be, elsőnek ki) módszert használó *adatszerkezet*et, *vermet* szimulál. A Piroska által látott dolgokat egy veremben elhelyezett objektumoknak lehet elképzelni. Az első látott objektum (kék virágok) kerül a verembe elsőnek („legalulra”), majd a második objektum (szökőkút) és a harmadik objektum (fészek) következik. Az út visszakövetéséhez az egyes objektumokat egymás után, fordított sorrendben kell eltávolítani a veremből.

Az informatikában sokféleképpen tárolhatjuk az adatainkat, sokféle adatszerkezetet használhatunk. A feladattól, használt adatoktól függően érdemes kiválasztani a megfelelőt. A vermeket (te is találkozhattál már más hódfeladatokban vele) pl. számolási műveletek egyértelmű tárolásához, akár különböző funkciójú, feladatú adatok feldolgozásához használják.

WEBOLDAL

Verem (adatszerkezet) – Wikipédia

Kategória:Adatszerkezetek – Wikipédia

KULCSSZAVAK

Verem, Adatszerkezet, Adat

Forgó lóhere (2024-IE-01c)

KISHÓD – KÖNNYŰ

Sanyi leejtette a képen látható kedvenc lóheréjét a földre.

Emiatt az összekeveredett más lóherékkel.

Melyik Sanyi, képen látható lóheréje?

A)

B)

C)

D)

A helyes válasz a D)

A D a helyes válasz, mert a sárgult levélcsúcs és a szár barnuló területe ugyanott van. Valamint a levélen lévő bevágás és a megevett rész is egyezik.

Az A lóhere nem azonos, mert nincs rajta a sárgult levélcsúcs a felső levélen.

A B sem azonos, mert a száron nincs ott a barnuló terület.

A C sem azonos, mivel a megevett rész és a bevágás nem egyezik.

MIÉRT INFORMATIKA?

A mai világban a képeken lévő információkat gyakran számítógépes programok dolgozzák fel, mégpedig úgy, hogy mintákat keresnek. A minták azonosításával az információk szűrhetők, elemezhetők és jobban megérthetők. Sokszor, ha eltávolítjuk azt, ami nem felel meg a mintának, jobb eredményeket találhatunk.

A minták bármilyen típusú adatban előfordulhatnak, akár azok részeként tökéletes másolatban vagy tökéletlenül, hibákkal, esetleg valamilyen módon átalakítva (pl. megfordítva). Minták vizsgálhatóak az adathalmazok közötti különbségekben is. Ebben a hódfeladatban a minták a forgatás és a felfordítás miatt kissé el vannak rejtve, hogy szemléltessük, hogy az adatokat esetleg fel kell dolgozni ahhoz, hogy felfedjük a keresett mintát.

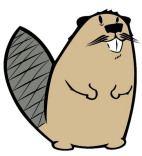
WEBOLDAL

Számítógép-programozás – Wikipédia

Feltételes utasítás – Wikipédia

KULCSSZAVAK

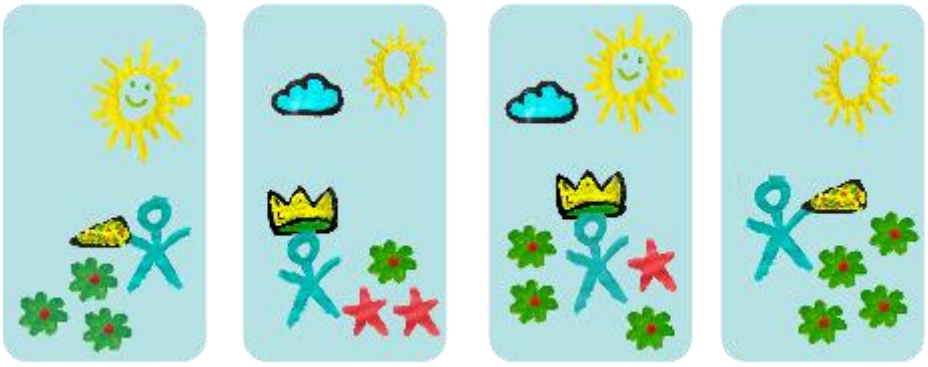
feltételes utasítás, információ



Kártyarendezés (2024-IN-04)

BENJÁMIN – NEHÉZ
KADÉT – KÖZEPES
JUNIOR – KÖNNYŰ

Matyi egy egyszerű, titkos szabályt követve kártyákat rajzol. Négy kártyát már el is készített, amelyek megfelelnek ennek a szabálynak:



Saci megfigyeli a kártyákat, és létrehoz egy másik kártyát, de Matyi azt mondja, hogy az **nem felel** meg a szabálynak.

Az alábbiak közül melyik lehet az a titkos szabály, amelyet Matyi követ a kártyák rajzolásánál?

A) Ha felhős az ég, nincsenek virágok.

B) Ha felhős az ég, a nap nem mosolyog.

C) Vagy egy piros csillagnak vagy egy szelet pizzának kell lennie a kártyán.

D) Ha a kártyán van egy szelet pizza, akkor nincs korona.

A helyes válasz a D)

A lehetőségeket mérlegelve megállapítható, hogy:

A : Nem, mert a 2. és 3. kártyán egy felhő és virágok is vannak.

B : Kizárható, mert ebből nem állapítható meg, hogy ez igaz-e vagy sem, mert nincs felhő a rajzolt kártyán.

C: Ez az állítás akár igaz is lehet, de esetünkben azt a szabályt keressük, amely miatt a fenti kártya nem felel meg.

D: Ennek a szabálynak nem felel meg Saci kártyája, de a Matyi kártyái igen, tehát ez lesz az, amelyik miatt Saci kártyája nem felel meg Matyi szabályainak.

MIÉRT INFORMATIKA

Ez a feladat az informatikában használt alapvető logikát szemlélteti: hogyan lehet döntéseket és megfigyeléseket reprezentálni és feldolgozni.

A számítógépek egyértelmű szabályokat követnek az eredmények meghatározásához és alapvető logikát használnak a feltételek feldolgozásához és a cselekvések meghatározásához, hasonlóan ahhoz, ahogyan a mindennapi életben megfigyeljük a mintákat és döntéseket hozunk.

WEBOLDALAK

Logika – Wikipédia

Boole-algebra (informatika) – Wikipédia

KULCSSZAVAK

Feltételes logika, Boole-logika

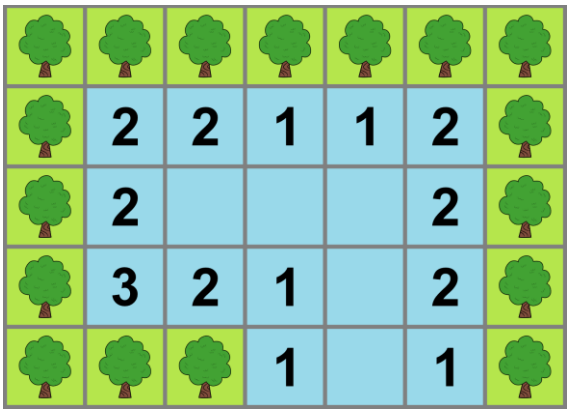
Élelem térkép (2024-IT-01b)

JUNIOR – NEHÉZ
SENIOR – KÖZEPES

Huba hód elrejtette az élelmiszerkészleteit a tavat körülvevő 17 fa közül 9 alá.

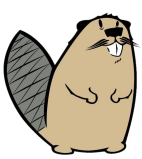
Készített egy térképet és minden part menti négyzetbe beírta, hogy a négyzet körüli fák közül hány alatt van elrejtett élelem.

Például a 3-as számot tartalmazó négyzet azt jelenti, hogy az 5 kiemelt fa közül pontosan 3 alatt van elrejtett élelem.



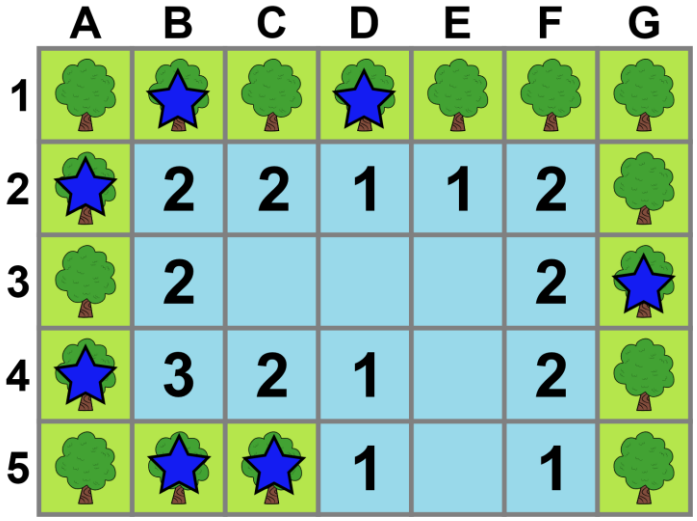
Ez nem tűnik jó ötletnek a barátnője, Bella szerint és igaza is van!

Valójában Huba csak 7 fát tud majd biztosan azonosítani a 9 közül, amelyek alatt elrejtette az ételt. Kattintással jelöld, hogy melyik ez a 7 fa!

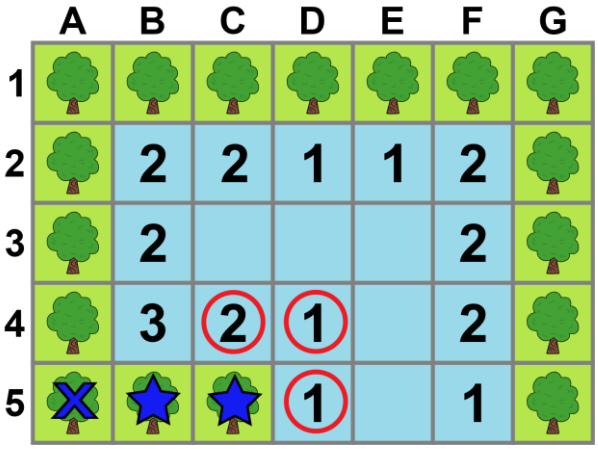


Megoldás

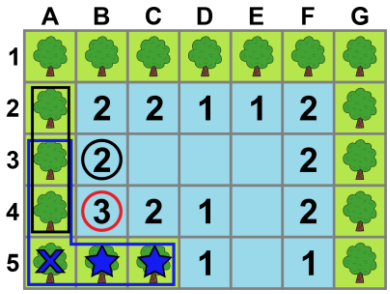
Az alábbi képen kiemelt hét fa az, amely alatt biztosan el van rejtve élelem:



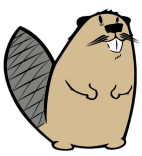
Alulról kiindulva két fát lehet azonosítani, amelyeken rejtett élelem van, és egy biztosan üres (X).

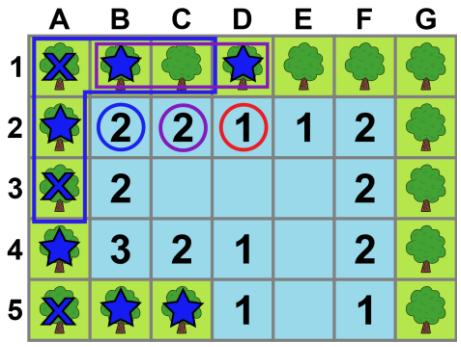


Hiszen az első fa biztosan rejt táplálékot, hogy a legalsó 1-esnek megfeleljen; a második fánál is kell legyen táplálék, hogy a legalsó 2-esnek megfeleljen. Ha a bal alsó sarokban lévő fa alatt lenne táplálék, akkor a 3-asnál már mindet megtaláltuk, azaz a mellette és a balra átlósan felette lévő fa alatt nem lenne táplálék. Emiatt a 3-as feletti 2-eshez nem tudnánk 2 olyan fát találni, ami alatt van táplálék.

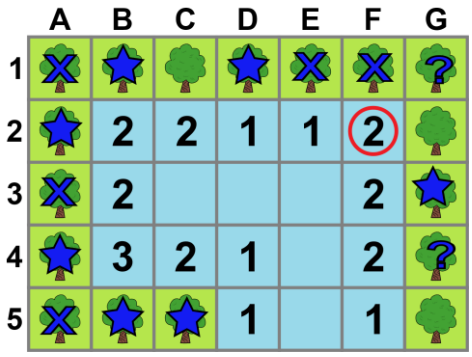


Ehhez hasonló módon könnyen bebizonyítható, hogy a tó bal oldali részét illetően a folytatás az alábbiak szerint alakulhat csak:

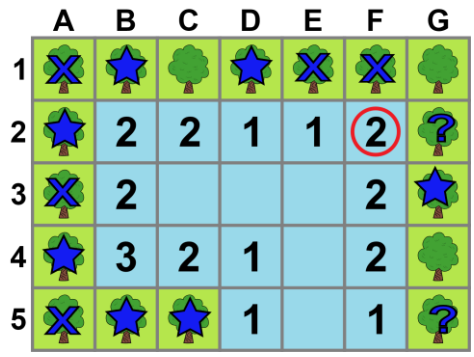




Ami a tó jobb oldalát illeti, két lehetőség van:



vagy



... így viszont már csak egy fát lehet beazonosítani biztosan.

Hogyan tudnánk jobban formalizálni a problémánkat? Miután kiválasztottuk az utolsó sorban azt a két négyzetet, amelyben biztosan van elrejtett étel, nevet adunk minden más négyzetnek, amely fát tartalmaz:

A kék dobozokban lévő számok korlátokat fejeznek ki; az aljától kezdve az óramutató járásával megegyező irányban az első három (1, 1, 2) már teljesült, a 3. korlát teljesítéséhez pontosan egy további elemet kell választani az {A, B, C} halmazból, és így tovább. Ezután a problémát a következőképpen lehet formalizálni:

- válassz az {A, B, C} halmazból pontosan egy elemet,
- válassz a {B, C, D} halmazból pontosan 2 elemet,
- válassz a {C, D, E, F, G} halmazból pontosan 2 elemet,
- válassz a {F, G, H} halmazból pontosan 2 elemet,
- válassz a {G, H, I} halmazból pontosan egy elemet,
- válassz a {H, I, J} halmazból pontosan egy elemet,
- válassz a {I, J, K, L, M} halmazból pontosan 2 elemet,
- válassz a {L, M, N} halmazból pontosan 2 elemet,
- válassz a {M, N, O} halmazból pontosan 2 elemet,

- válassz a {N, O} halmazból pontosan egy elemet.

Az összes lehetőség vizsgálata során (nem választhatod az A-t az első halmazból, mert akkor a másodikból B-t vagy C-t kellene választanod; ha az első halmazból B-t választod, akkor a másodikból D-t kell választanod, és így tovább) megállapítható, hogy két megoldás létezik:

{B, D, F, H, K, M, N}, {B, D, F, H, L, M, O}

Az elemek, amelyeket biztosan piros csillaggal lehet jelölni, a B, D, F, H, M (amelyek a két kapott halmaz metszetét alkotják), csak úgy mint a korábban kapott fák esetében.

MIÉRT INFORMATIKA?

Ezt a feladatot egy híres logikai rejtvény-videójáték ihlette, amely különböző neveken ismert (Aknakereső, Virágmező), ahol a szerencse is szerepet játszik. Sokszor előfordulhatnak ugyanis olyan helyzetek, amikor nincs elég információ a játék biztonságos folytatásához, így végül a szerencsére kell hagyatkozni. Pontosan ez a helyzet merül fel a feladatunkban: a tó jobb oldala nem oldható meg egyértelműen!

Vegyük észre, hogy általában, ha N fa van a tó körül, akkor ezek 2N lehetséges konfigurációnak felelnek meg, és ilyenkor általában sokkal könnyebb ellenőrizni, hogy egy adott megoldás összhangban van-e a probléma korlátaival, mint megtalálni azt!

Ezen a feladaton keresztül azt is láthatjuk, hogy egy probléma egészét milyen gyakran lehet fokozatosan külön részekben kezelni és megoldani. Az informatika szempontjából messze a legérdekesebb aspektus a probléma formalizálása egy megfelelő adatstruktúra révén (ahogy ezt a feladat utolsó részében tettük), ami azt sugallja, hogy általában ez a probléma számításilag nehéz, valamint több megoldása is lehetséges.

WEBOLDAL

P versus NP probléma – Wikipédia

Minesweeper Online

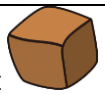
KULCSSZAVAK

Logika, Aknakereső, NP-teljesség

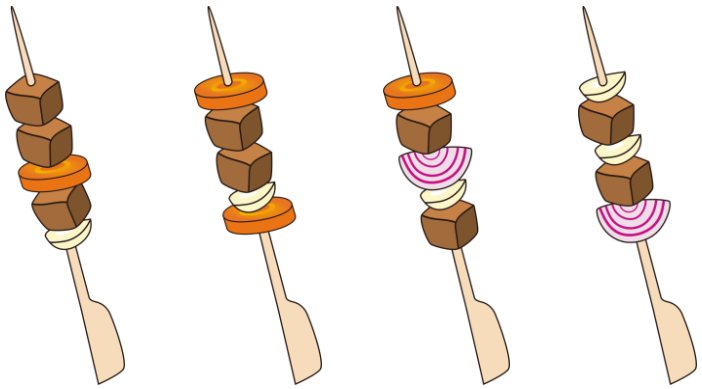
Grillparti (2024-KR-02)

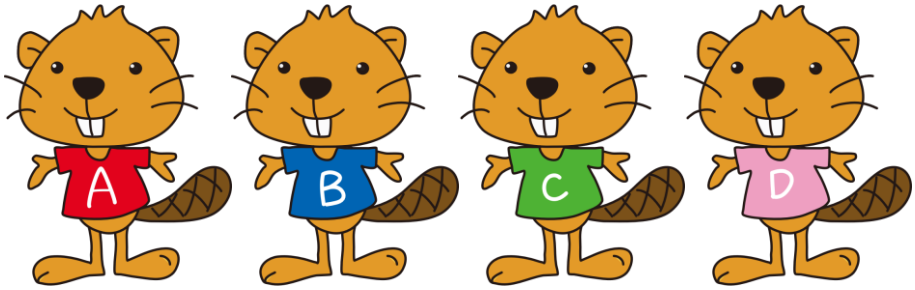
KISHÓD – KÖNNYŰ

A grillpartin résztvevő négy hód leírja, milyen nyársat szeretne:

			
Szeretnék répát 	Nem ehetek hagymát 	Annyi fokhagymát  szeretnék, amennyit csak lehet	Annyi húst szeretnék, amennyit csak lehet 

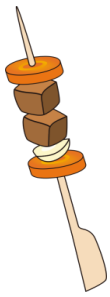
Oszd el a 4 nyársat a hódok között úgy, hogy minden hód a legelégedettebb legyen!





A megoldás:

A helyes válasz a következő:

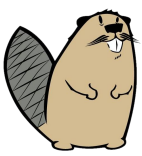
			
			

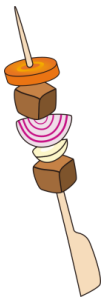
Az egyszerűség kedvéért, jelöljük számokkal a nyársakat.

			
1	2	3	4

- A-nak sárgarépát kell tennie a nyársára, így az 1, 2 és 3 lehetőségek közül választhat.
- B nem ehet hagymát, így az 1 és 2 lehetőségek közül választhat.
- C a lehető legtöbb fokhagymát szeretné enni, így a 4-es számú a legjobb választás. Azonban az 1, 2 és 3 lehetőségek is lehetségesek.
- D a lehető legtöbb húst szeretné enni, így az 1-es számú a legjobb választás. Azonban a 2, 3 és 4 lehetőségek is lehetségesek.

Ezek alapján a legjobb opció kiválasztásához egy táblázatot is létre tudunk hozni, ahol tudjuk jelölni, ha valamelyik hód semmiképpen nem kaphatja meg az adott nyársat(---), hogy az adott nyárs megfelel neki(0) és azt is, ha az a legjobb választás számára(+++):



				
A	+++	+++	+++	---
B	+++	+++	---	---
C	0	0	0	+++
D	+++	0	0	0

A táblázatból pedig következik, hogy:

- Csak a 4. nyárson van több mint egy gerezd fokhagyma. C kapja a 4. nyársat.
- Csak az 1. nyárson van több mint két darab hús. D kapja az 1. nyársat.
- A két megmaradt nyárs közül csak a 3.-on van hagyma; ez ne B-hez kerüljön, hanem A-hoz.
- B kapja a 2. nyársat.

MIÉRT INFORMATIKA?

A korlátok kielégítési problémája (CSP) az egyik kulcsfontosságú fogalom az informatikában. A CSP olyan probléma, amelyben egy értéket kell találni, amely kielégít bizonyos korlátokat (feltételeket) egy változóhalmazon.

Ezek a problémák különböző területeken fordulnak elő, mint például az ütemezés, tervezés, erőforrás-allokáció, elrendezés tervezés és számítógépes látás. Az informatika területén a CSP fontos megközelítés a problémák megoldására.

Az informatika magában foglalja az adatok elemzését, algoritmusok fejlesztését, valamint szoftverek és rendszerek tervezését, ahol a CSP felhasználható hatékony megoldások kidolgozására. Például a CSP használható adatbázis-rendszerekben, amikor korlátokat kell meghatározni az adatok konzisztenciájának fenntartása érdekében, és megoldásokat kell találni, amelyek kielégítik ezeket a korlátokat.

WEBOLDAL

Logikai kielégítési probléma – Wikipédia

KULCSSZAVAK

Logika, CSP

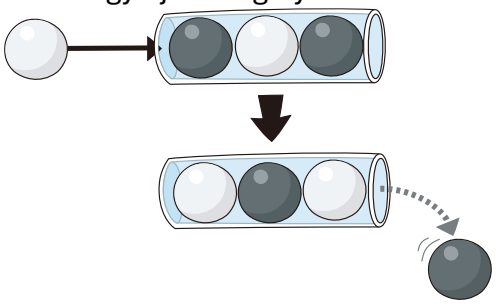
Átlátszó csövek (2024-KR-03a)

KISHÓD – KÖZEPES
BENJÁMIN – KÖNNYŰ

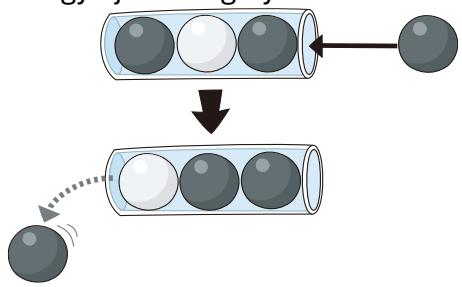
Egy átlátszó csőbe legfeljebb három golyó fér. A cső mindkét oldalán nyitott.

Ha egy teli csőbe még egy golyót teszünk be, a másik oldalon egy golyó kiesik:

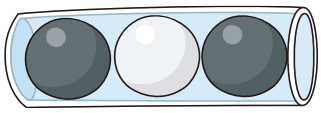
balról egy új fehér golyót teszünk be



Jobbról egy új fekete golyót teszünk be

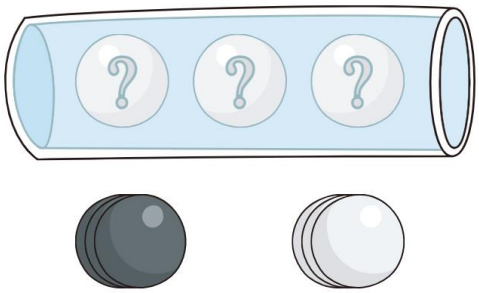


A csőben most három golyó van:



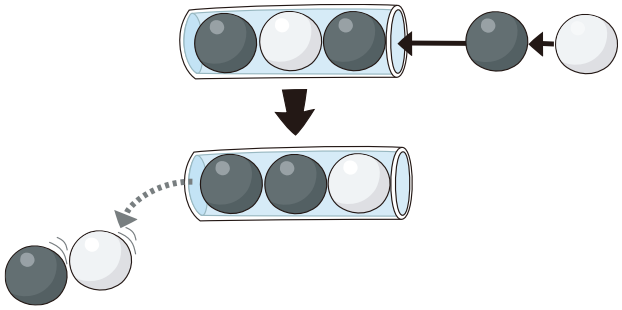
További golyókat teszünk be egymás után a csőbe az alábbi sorrendben (és irányból)

Milyen golyók lesznek végül a csőben?

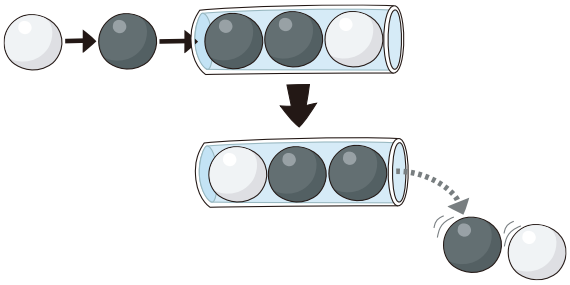


Megoldás/ A helyes válasz:

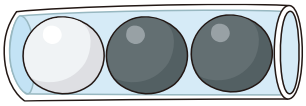
Két golyót teszünk a csőbe jobbról, először egy fekete, majd egy fehér golyót. Ennek eredményeként balra két golyó esik ki, először a fekete, majd a fehér.



Ezután balról két golyót teszünk a csőbe, először egy fekete, majd egy fehér golyót. Ezért ezúttal jobbra két golyó esik ki, először a fehér, majd a fekete.



Tehát a végén a csőben egy fehér és két fekete golyó lesz:

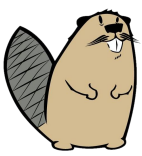


Ebben az esetben gyorsabban is megadhatjuk a választ, mivel a csőbe először jobbról két golyót, majd balról két golyót teszünk be. Emiatt a csőben az eredetileg jobbra lévő (fekete) golyó ismét visszakerül a helyére, tőle balra (jobbról balra) pedig a csőbe utoljára betolt két golyó kerül (fekete, fehér – ebben a sorrendben).

MIÉRT INFORMATIKA?

Ennek a feladatnak a csöve a sorok egy speciális formája, amelyet az informatikában Kétvégű sornak (deque, a double-ended queue rövidítése) neveznek. A hagyományos sorral ellentétben a deque mindkét végéhez hozzáadhatunk egy elemet, és a deque mindkét végéből eltávolíthatunk egy elemet.

A deque egy adatszerkezet, mely sokféleképpen használható, például a verem (utolsónak hozzáadott vehető el/ki elsőként) és az egyszerű listák (elsőnek hozzáadott vehető el/ki elsőnek) általános helyettesítőjeként. Gondolhatsz rá úgy is, mint egy tolatási pályára vonatoknál: elöl és hátul is lehet vagonokat hozzáadni vagy eltávolítani, de középről nem lehet egy vagont eltávolítani. Egy egyvégű vágánynál ezzel szemben a vagonokat csak az egyik oldalon lehetne fel- és lekapcsolni.



Ahogyan a feladathoz tartozó cső is csak legfeljebb három golyót tartalmazhat, úgy egy sornak is gyakorlatilag korlátozott a kapacitása. Ha egy már teljesen kihasznált kétvégű sorhoz hozzáadunk egy elemet, akkor a másik végén helyet kell csinálni. Ez úgy történik, hogy eltávolítunk egy elemet, és elveszítjük az ehhez tartozó információt. A tárhely véges erőforrás, és a tárolható adatok mennyisége függ tőle. Ennek ellenére a kétoldalú sorokat gyakran definiálják hipotetikusan korlátlannak, különösen akkor, ha inkább elméleti kérdésekkel, mint gyakorlati alkalmazásokkal foglalkoznak.

WEBOLDAL

Sor (adatszerkezet) – Wikipedia

Double-ended queue - Wikipedia

KULCSSZAVAK

Kétvégű lista, Kétvégű sor, Adatszerkezet

Barátok (2024-LT-05)

JUNIOR – NEHÉZ
SENIOR – KÖZEPES

Hét hód Alex, Bess, Cora, Dave, Eric, Fred és Gigi egy iskolába járnak. Néhányan ismerik egymást, néhányan pedig nem.

Az alábbi ábrán két hód között egy pipa van, ha ismerik egymást.

	Alex	Bess	Cora	Dave	Eric	Fred	Gigi
Alex		✓	✓				
Bess	✓						✓
Cora	✓				✓		
Dave						✓	
Eric			✓				
Fred				✓			
Gigi		✓					

Mindegyik hód kapott egy üzenetet valakitől az iskolán kívülről, és megosztották azt az összes hóddal, akit az iskolájukban ismernek. Amikor egy hód új üzenetet kap valakitől az iskolán belülről, azonnal megosztja azt az összes hóddal, akit az iskolájában ismer (kivéve azzal, akitől kapta).

Ki vagy kik kapták összesen a legkevesebb üzenetet?

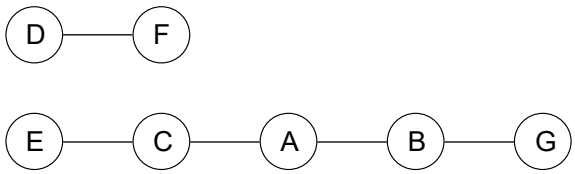
Alex	Bess	Cora	Dave	Eric	Fred	Gigi

Megoldás/ A helyes válasz Dave és Fred.

Alex	Bess	Cora	Dave	Eric	Fred	Gigi

Az iskolában szereplő hódok közötti kapcsolatokat ábrázolhatjuk egy ábra segítségével. Ezen az ábrán minden hódot egy-egy kör ábrázol, és ha a két hód ismeri egymást, akkor a körüket egy vonal összekötjük.

Dave csak Fredet ismeri, és Fred csak Dave-et ismeri, ezért összekötjük a köreiket, és nem kapcsoljuk össze őket más körökkel. Eric csak Corát ismeri, ezért összekötjük a körüket. Cora ismeri Alexet is, tehát összekötjük a körüket. Alex ismeri Bess-t is, ezért összekötjük a körüket. Végül Bess ismeri Gigit is, így összekötjük a körüket.



Erről az ábráról leolvashatjuk, hogy Dave és Fred csak két üzenetet kap (amit egymásnak küldenek), amíg a többiek ötöt (pl. Cora elküldi Eric-nek és Alex-nek, Alex továbbküldi Bess-nek, aki továbbítja Giginek. Azaz Cora üzenetét mindenki megkapja ebből a csoportból. Ugyanígy Eric, Alex, Bess és Gigi üzenetét is mindenki megkapja.)

MIÉRT INFORMATIKA?

A gráfok az informatikában fontos adatszerkezetnek számítanak. Sokféleképpen ábrázolhatjuk ezeket, például mint ebben a hódfeladatban szomszédsági mátrixként.

A mátrix elemei azt jelzik, hogy csúcspárok szomszédosak-e vagy sem a gráfban. A feladatban szereplő gráf irányítatlan (azaz minden éle kétirányú), emiatt ez a mátrix szimmetrikus.

Az, hogy hogyan ábrázoljuk a gráfokat, azaz hogyan és milyen információt tárolunk az elemekről (csomópontokról és kapcsolatokról) támogatni, gyorsítani tudja majd a feldolgozást. Amennyiben tudjuk, hogy irányítatlan gráfunk van szomszédsági mátrixban, ugyan több helyen tárolunk (és a felét feleslegesen), de a feldolgozás során elég a mátrix „felével” törődnünk.

WEBOLDAL

Szomszédsági mátrix – Wikipédia

Gráf – Wikipédia

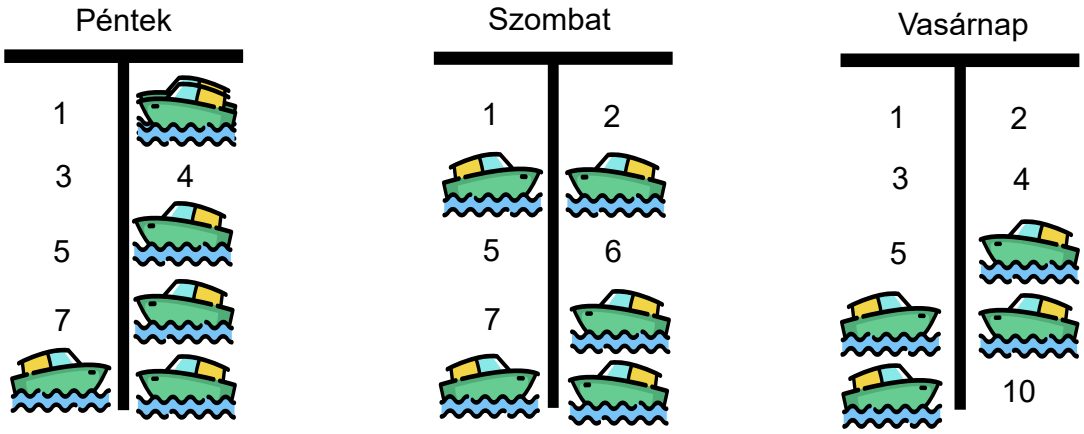
KULCSSZAVAK

Gráf, Szomszédsági mátrix, Adatszerkezet

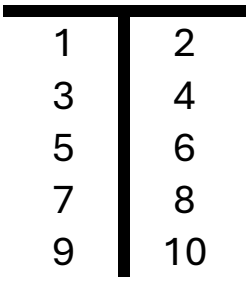
Hajóparkoló (2024-MT-02)

KISHÓD – NEHÉZ
BENJAMIN – KÖZEPES
KADÉT – KÖNNYŰ

Hód Tomi segít leparkolni a csónakokat egy speciális vízi parkolóban. Az alábbi képen látható, hogy melyik napon hol parkolnak hajók. Most olyan hajók számára kell üres parkolóhelyeket találnia, amelyek két *egymást követő napon* szeretnének maradni ugyanazon a parkolóhelyen.



Válaszd ki azokat a parkolóhelyeket, amelyek legalább két egymást követő napon üresek lesznek!



A zölddel jelölt parkolók lesznek szabadok két egymást követő napon:

1	2
3	4
5	6
7	8
9	10

MIÉRT INFORMATIKA?

Minden adatot nullák és egyesek sorozataként lehet elképzelni. Minden egyes nullát vagy egyest bitnek, az ezekből álló sorozatot pedig bináris kódnak vagy bináris számnak nevezzük.

Ebben a feladatban egy üres helyet 0-ként, egy foglalt helyet pedig 1-ként modellezhetünk; így minden egyes parkolóhely egy bitnek felel meg. A bitek sorozatát úgy kaphatjuk meg, hogy egy nap parkolóhelyeit nézzük. Például a péntek megfelel a 0100010111, a szombat a 0011000111, a vasárnap pedig a 000001110 bit sorozatnak.

Ez a feladat azt kéri, hogy vegyük a sorozatot két egymást követő napra, olyan biteket (parkolóhelyeket) keresve, amelyeknél ugyanazon a helyen mindkettő 0. Ennek megtalálására használhatjuk a NEM VAGY (NOR) nevű logikai operátort. Konkrétan:

- 0100010111 NOR 0011000111 egyenlő 1000101000
- 0011000111 NOR 000001110 egyenlő 1100100000

Azaz akkor kapunk 1-et, ha egyik érték sem 1. Minden más esetben pedig 0-t kapunk.

Az informatikában sok helyen használunk logikai állításokat és műveleteket. Például az e-mail programok spamszűrői logikai operátorokkal dolgoznak: a program csak akkor helyez át egy e-mail-t a spam mappába, ha az e-mail ismeretlen feladótól érkezik ÉS a tárgysorban bizonyos szavak szerepelnek. Tehát a program csak akkor mozgatja a levelet, ha mindkét részleges állítás IGAZ.

WEBOLDALAK

Logikai kapu – Wikipédia,

Logikai függvények – Wikipédia,

Bool algebra – Wikipédia

KULCSSZAVAK

Logikai kapu, Bool algebra, NOR

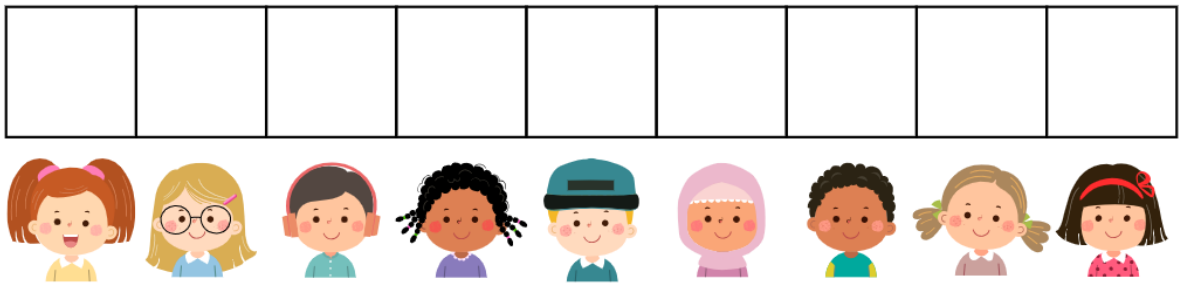
Online találkozó (2024-MY-03)

KISHÓD – NEHÉZ
BENJAMIN – KÖZEPES
KADÉT – KÖNNYŰ

Ava új iskolába jár, de szeretné viszontlátni a régi iskolai barátait. Barátai egymás mellett ülnek, de különböző számítógépeket használnak.

Így látja Ava a barátait a képernyőn:

Húzd Ava barátainak a képét sorba, ahogy egymás mellett ülnek?

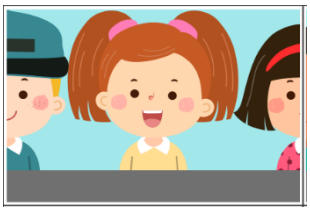


Megoldás/ A helyes válasz:

Abból, amit a képernyőkön látunk, összerakhatjuk, ki ki mellett ül. A képernyőn jobbra lát-

hatjuk, hogy  mellett az egyik oldalon nem ül senki, tehát ő

biztosan a szélén ül a sornak. A mellette ülő  -t kell



megkeresnünk (hiszen ő látszik még a képen), és innen tudjuk folytatni: mindig megkeressük a következő „belógó” arcot valamelyik másik képernyőrész közepén.

MIÉRT INFORMATIKA?

Ebben a feladatban olyan képek vannak, amelyeken nem csak egy gyereket, hanem a mellette lévő gyerekek egy részét is láthatod. Ezek mintegy „referenciaként” szolgálnak a szomszédok számára.

Az informatikusok sokszor használnak hasonló adatszerkezeteket.

Ezt „kétszeresen láncolt listának” nevezik. Egy kétszeresen láncolt listában a lista minden egyes eleme tartalmazza az elem értékét, egy hivatkozást az előző elemre és egy hivatkozást a következő elemre. Ebben a feladatban minden egyes képernyő (elem) a gyereket (a lista elemét) és a gyerektől jobbra és balra lévő gyermekekre való hivatkozásokat (mutatókat) tartalmazza, mutatja.

Ez a kettős hivatkozás segít az informatikusoknak abban, hogy ne csak a következő, hanem az előző elemet is elérjék. Ez lehetővé teszi számukra, hogy gyorsabban megtaláljanak és szerkesszenek egy elemet. Természetesen kicsit több memóriára van szükség ahhoz, hogy minden elemhez két mutatót is tároljunk.

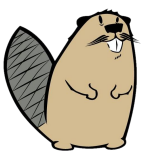
WEBOLDAL

Láncolt lista – Wikipédia

Adatszerkezet – Wikipédia

KULCSSZAVAK

Kétszeresen láncolt lista, Láncolt lista, Adatszerkezet



Szülinapi ajándék (2024-NL-02)

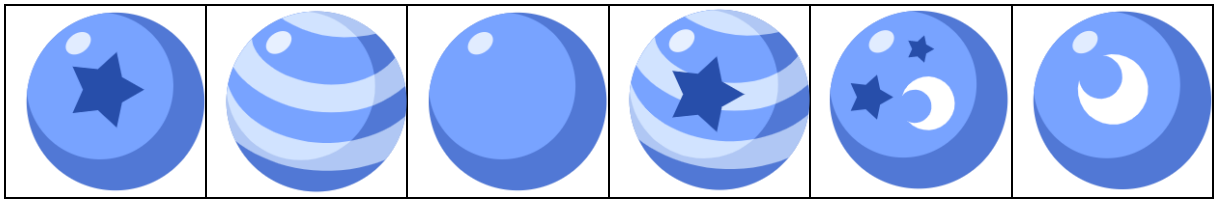
KISHÓD – KÖZEPES
BENJAMIN – KÖNNYŰ

Fiona egy labdát szeretne a születésnapjára. Szeretné, ha a labda

- nem lenne csíkos;
- lenne rajta csillag;
- nem lenne rajta hold.

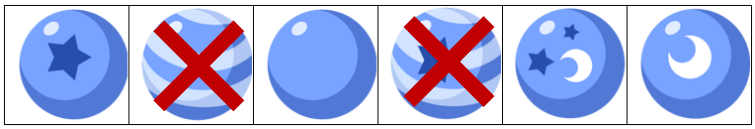
Melyik labdát vegyük meg Fionának?

Kattints a labdára, ami Fiona kívánságainak megfelel.

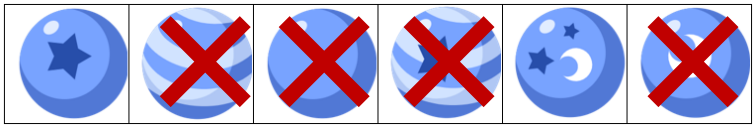


A helyes válasz: 

Fiona olyan labdát szeretne, ami nem csíkos. Emiatt a második és a negyedik labda nem megfelelő.



Olyat szeretne, amin van csillag. Emiatt a harmadik és a hatodik (utolsó) labda sem megfelelő.



Az utolsó kérése, hogy ne legyen rajta hold. Ez a maradék két labda közül csak az elsőre igaz.

MIÉRT INFORMATIKA?

A feladatban szereplő labdáknak több tulajdonsága is van. Nem csak a mintázatuk, de különböző lehetne a méretük, a keménységük, a színük is. Ezeket a tulajdonságokat eltárolva rá tudunk keresni a labdákra 1-1 tulajdonság alapján.

Az objektumok tulajdonságainak tárolása – pl. hogy milyen formában, milyen módon – meghatározza, hogyan tudunk majd ezekre rákeresni, csak azokat kiszűrni, amelyek egy (vagy több) tulajdonságukkal megfelelnek a feltételeinknek.

Ebben a hódfeladatban nem is egy tulajdonsággal rendelkező objektumot (labdát) akartunk megtalálni, hanem olyat, ami egyszerre három feltételünknek is megfelel. A megoldás során egyenként megkerestük azokat, amik megfelelnek, és azok között néztük meg a második, majd az azzal is szűkítettek közül a harmadik feltételünknek megfelelőeket.

Megtehettük volna azt is, hogy „összefűzzük” ezeket a feltételeket és minden labdán egyesével végig nézzük, hogy teljesül-e egyszerre mindhárom feltétel.

Ez két különböző stratégia, algoritmus. Az, hogy melyiket érdemes használni, függhet az objektumok számától, az objektumokra eltárolt tulajdonságok számától vagy a feltételeink összetettségétől is.

WEBOLDAL

Algoritmus – Wikipédia

Informatika 5. évfolyam | Sulinet Tudásbázis

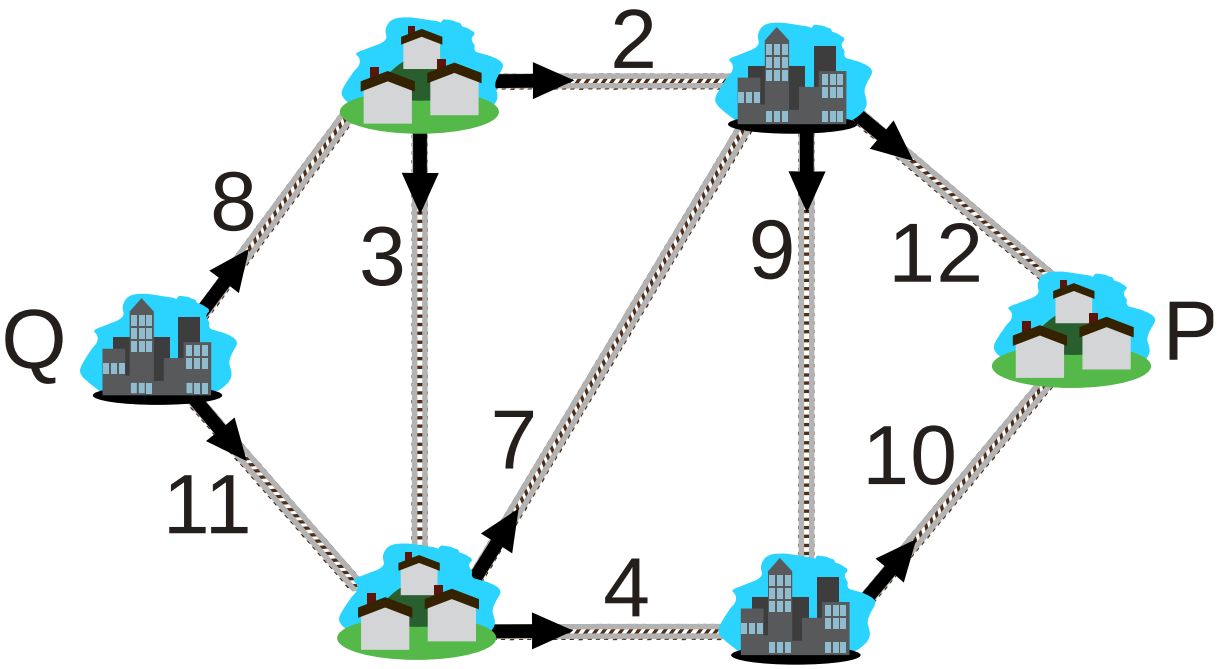
KULCSSZAVAK

Algoritmus, Logika

Vasúthálózat (2024-PK-03)

JUNIOR – NEHÉZ
SENIOR – KÖZEPES

Bebravia földjén a szomszédos településeket vasúti pályák hálózata köti össze. Minden pályán naponta csak korlátozott számú vonat közlekedhet, ahogy azt alábbi ábra mutatja.



Q város felajánlotta, hogy nyersanyagot küld P városnak. Ebben az irányban a vonatoknak mindig a nyilakat kell követniük.

Legfeljebb hány vonat indulhat Q városból és érkezhet P városba naponta?

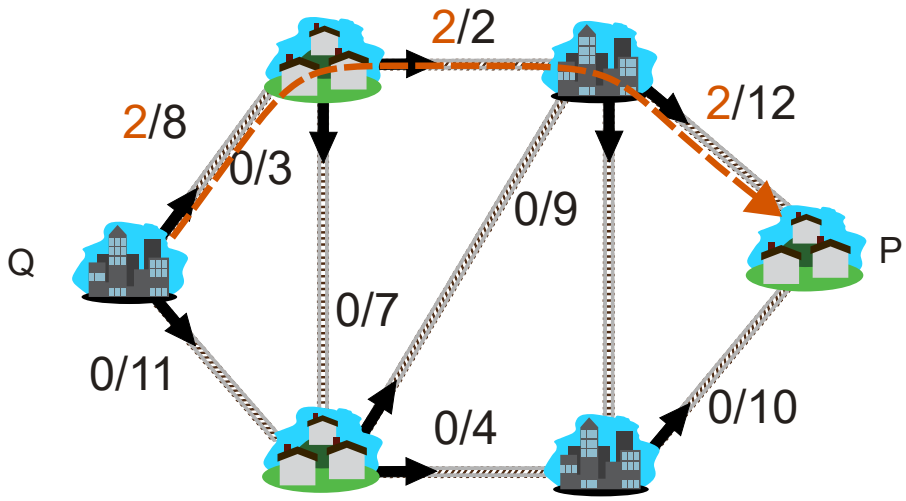
A) 13
B) 15
C) 19
D) 22

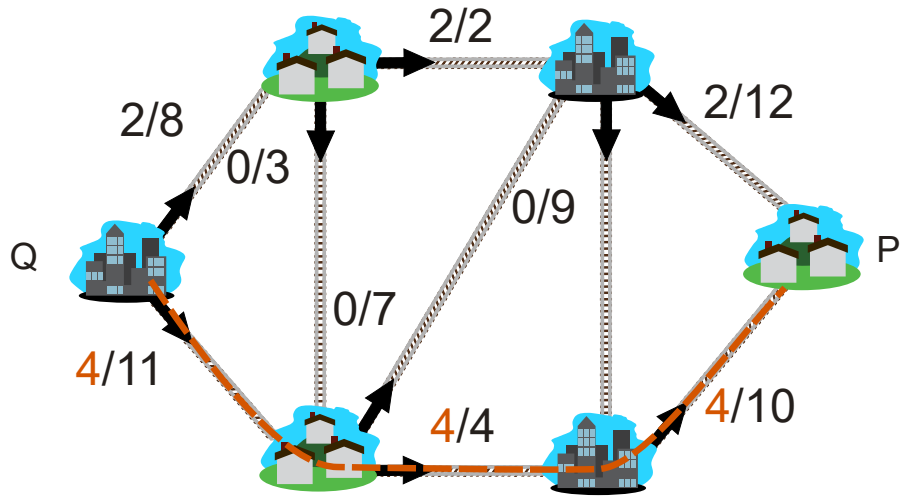
A helyes válasz az A)

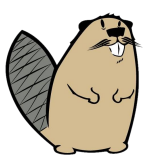
Az első megfigyelés amit tehetünk az az, hogy Q város nem küldhet P városba több vonatot, mint 8+11=19 vonat, ami a Q városból indítható vonatok száma.

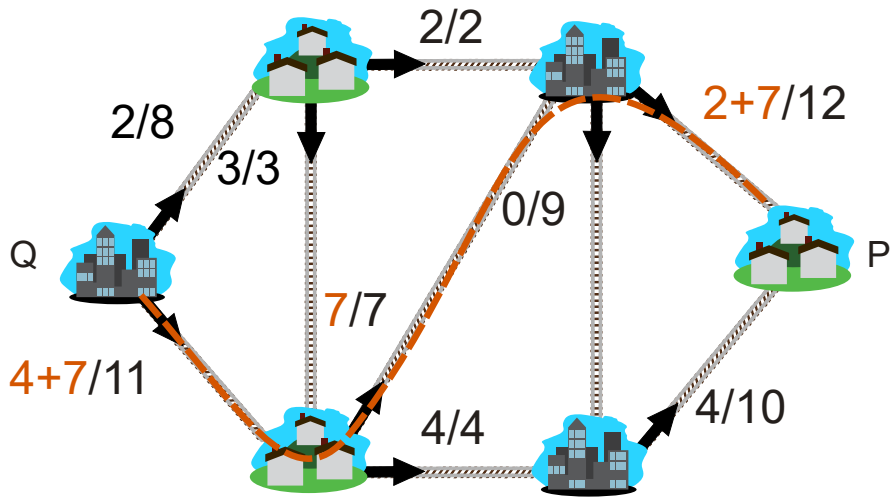
A probléma megoldásának egyik módja, hogy megkeressük a lehető legrövidebb (ebben az esetben 3 hosszúságú) útvonalat Q városból P városba, és meghatározzuk a rajta elférő vonatok legnagyobb számát.

Ezután megismételjük ezt a lépést a hálózat többi részén az egyes vágányokon rendelkezésre álló vonatok fennmaradó számával. Az alábbi ábrák három, így talált útvonalat mutatnak, amelyek együttes kapacitása 2+4+7=13 vonat. Más módon is lehet 13 vonatot Q városból P városba juttatni, többek között hosszabb útvonalakat használva.









Annak ellenőrzéséhez, hogy a 13 a maximum, figyeljük meg a hálózat közepén lévő három szakaszt 2, 7 és 4 kapacitással: minden útvonal ezek egyikén megy keresztül, így nem tudunk több vonatot célba juttatni, mint 2 + 7 + 4 = 13.

MIÉRT INFORMATIKA?

Ebben a hódfeladatban a maximális áramlási problémára mutattunk példát. Az ilyen típusú problémák olyanok, mint a csöveken átfolyó vízzel kapcsolatos kirakós játék: van egy különböző kapacitású csövekből álló hálózatunk, és azt szeretnénk kitalálni, hogy mennyi víz folyhat el egy forrásból egy mosogatóba.

A maximális áramlási probléma különböző algoritmusokkal oldható meg: Edmonds-Karp algoritmus, Dinit'z algoritmus és még sok más algoritmus. A hálózat közepén lévő elkerülhetetlen szakaszok ebben a feladatban úgynevezett „minimális vágást" alkotnak.

A fenti algoritmusok mindegyike rendelkezik előnyökkel és hátrányokkal a futási idő hatékonysága, a könnyű megvalósíthatóság és a különböző típusú hálózatokra való alkalmazhatóság szempontjából. Az algoritmus kiválasztása olyan tényezőktől függ, mint a hálózat mérete, a kapacitások jellege és a kívánt optimalizálási szint. A maximális áramlási algoritmusokat többek között a forgalom optimalizálásában, az elosztórendszerekben (pl. víz és villamos energia), az erőforrás-elosztásban és a számítógépes hálózatokban használják.

WEBOLDAL

Maximális folyam – minimális vágás – Wikipédia

Dinic's algorithm - Wikipedia

Maximális folyam, Ford-Fulkerson-algoritmus | mateking

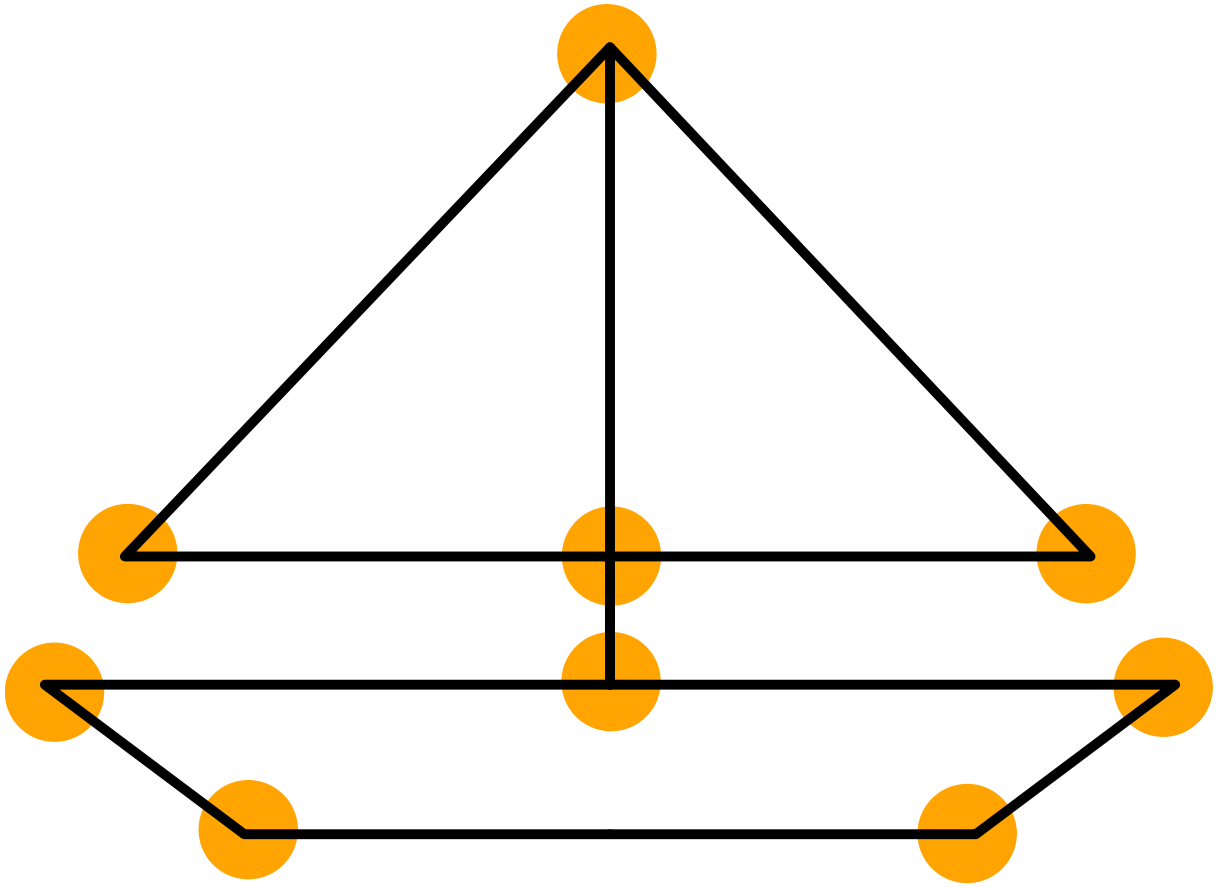
KULCSSZAVAK

gráf, maximális áramlás, minimális vágás, Ford-Fulkerson tétel, Dinit'z algoritmus

Vitorláshajó (2024-PL-03)

KISHÓD – KÖZEPES
BENJAMIN – KÖNNYŰ

Kata át akarja rajzolni a vitorlásról készített rajz vonalait.



Két szabályt fogalmazott meg:

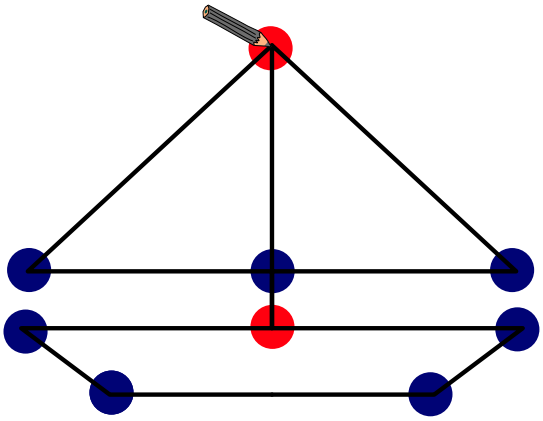
- Úgy akarja a meglévő vonalakat újra átrajzolni, hogy közben nem emeli fel a ceruzát.
- Minden vonalon pontosan csak egyszer akar átmenni, azaz többször nem rajzolhat át egyetlen vonalat sem.

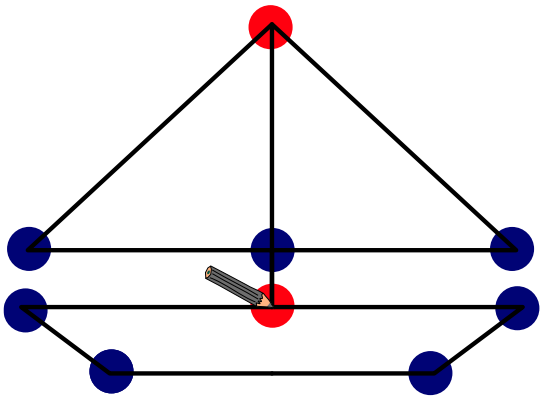
Katának sikerülhet az átrajzolás, ha azt a megfelelő kiindulóponton kezdi el.

Hol kezdje az átrajzolást? Kattints a kiindulási pontra!

A megoldás:

Próbálgatások útján kiderül, hogy az alábbi pontok alkalmasak kiindulópontnak.



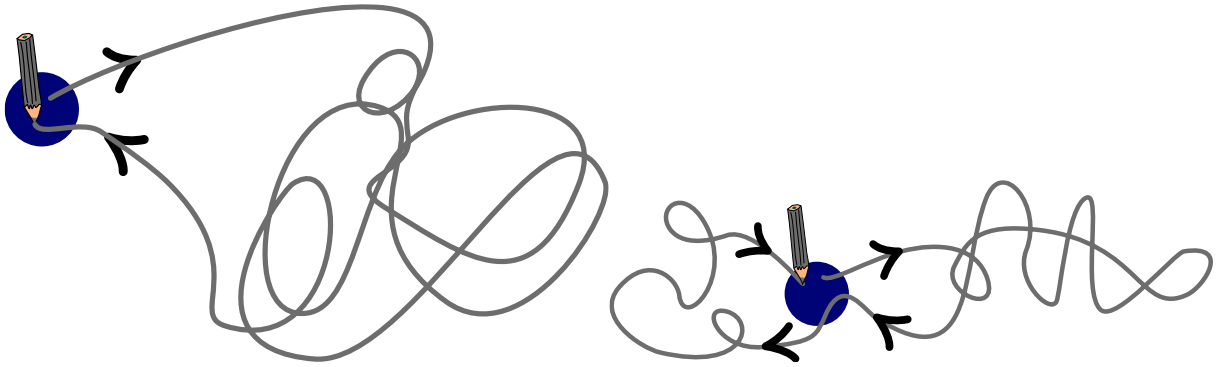


A próbálgatásnál van azonban gyorsabb megoldási mód is: kétféle pont között lehet különbséget tenni:

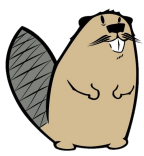
- A kék pontokból páros számú vonal indul ki.
- A piros pontokból páratlan számú vonal indul ki.

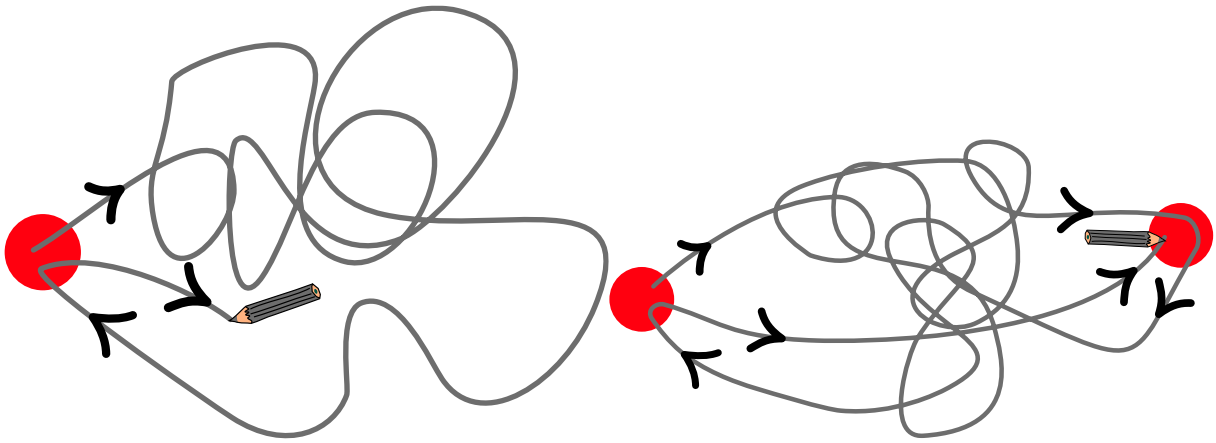
Kata vagy egy kék, vagy egy piros pontnál kezdhet. Mindkét lehetőséget kipróbáljuk:

Kata egy kék pontnál kezdi, és meghúzza az első vonalat. Ahhoz, hogy mindkét, ebből a pontból induló vonalat újra rajzolja, vissza kell térnie ehhez a ponthoz. Minden olyan pontnál, ahol páros számú vonal indul ki, a végén ugyanahhoz a ponthoz kell visszaérkeznie. Hogy ez lehetséges-e, azt nem tudja megmondani, ezért próbálgatással kell megtalálnia a megoldást.



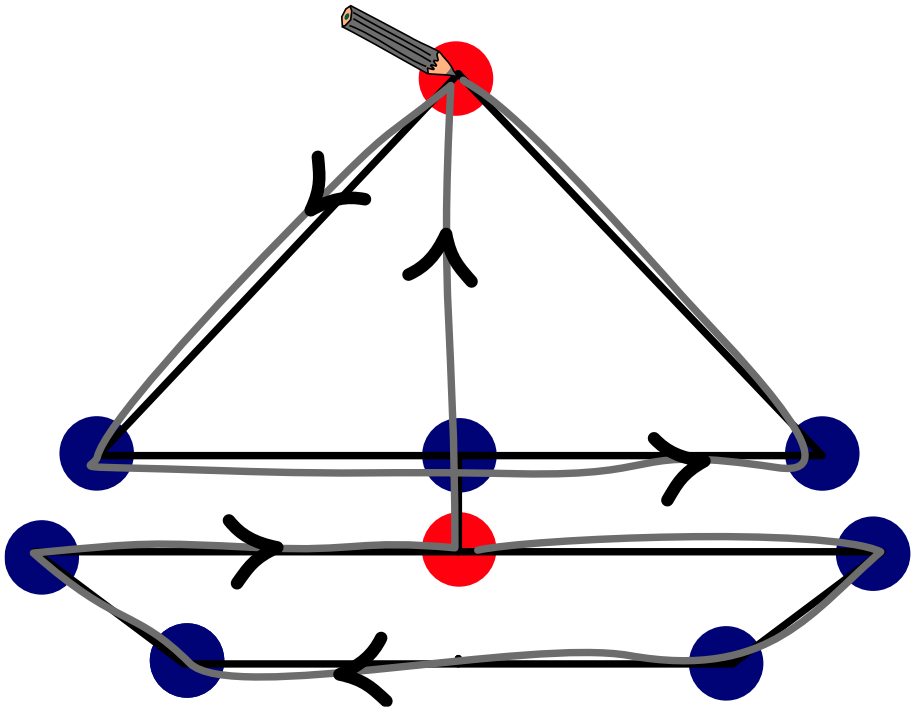
Ha Kata egy piros pontnál kezdi. Valamikor visszatér ehhez a ponthoz, és az utolsó, még nem átrajzolt vonalon keresztül el tudja hagyni, és el is kell hagynia, hiszen minden vonalat át szeretne rajzolni. Most egy másik pontot kell találnia, ahol befejezi a rajzát (egy végpontot). Itt a második piros pont kínálkozik, mivel ennél is van még egy nem átrajzolt vonal. Tehát van kapcsolat a két piros pont között: Katának az egyik piros pontnál kell kezdenie és a másik piros pontnál kell befejeznie.





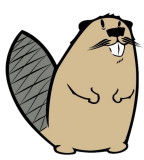
Az elmélet most megerősíti, hogy a problémának valóban van megoldása. Kata ezt természetesen nem tudhatja. Felfedezheti azonban, hogy a megoldásnak egy piros pontnál kell kezdődnie, és a másik piros pontnál kell befejeződnie.

Próbálgatás útján például megrajzolhatja a hajótestet, majd az árbocon keresztül az egész vitorlát és így az egész hajót átrajzolhatja.



MIÉRT INFORMATIKA?

A vitorlás rajza pontokból és vonalakból áll, amelyek összekötnek néhány pontot. Az informatikában az olyan struktúrákat, amelyekben bizonyos elemek összekapcsolódnak, gráfnak nevezzük. Egy gráf csomópontokból (pontokból) és élekből (vonalakból) áll. A vitorlás rajza tehát egy gráf ábrázolása: a pontok a csomópontok, a vonalak pedig az élek, amelyek két csomópontot kötnek össze.



Ebben a hódfeladatban Kata olyan utat szeretne találni a vitorlás-gráfban, amely minden csomóponton áthalad, és amelyben minden él pontosan egyszer fordul elő. A csomópontokat tetszőleges számú alkalommal látogathatja meg. Ahhoz, hogy a probléma megoldható legyen, nem lehetnek olyan csomópontok, amelyeket egyáltalán nem lehet elérni.

Egy gráf, amely megfelel ennek a feltételnek, összefüggő. A "Mikulás háza" egy ismert rejtvény, amelyben gyerekek és felnőttek egy ilyen utat keresnek egy összefüggő gráfban.

A svájci matematikus, Leonhard Euler (1707-1783) jelentős mértékben hozzájárult az ilyen típusú problémák megértéséhez és megoldásához. Az ő kutatása nyomán az informatika az ilyen utakat, mint amilyet Kata is keres a vitorlás-gráfban, Euler-útnak nevezi. Egy ismert algoritmus, amely megállapítja, hogy van-e Euler-út egy összefüggő gráfban, azt vizsgálja, hogy van-e a gráfban olyan csomópont, amelyhez páratlan számú él kapcsolódik.

- Ha pontosan két ilyen csomópont van, akkor a probléma megoldható, és az Euler-út az egyik ilyen csomópontnál kezdődik, és a másiknál végződik.
- Ha több, mint két ilyen csomópont van, akkor a gráfban nincs Euler-út.
- Ha egy gráfban csak páros számú él kapcsolódik minden csomóponthoz, akkor bármelyik csomópontból kiindulva felépíthető egy Euler-út. Ezek az Euler-utak körök, mert ugyanabban a csomópontban kezdődnek és végződnek. Az informatikában ezt Euler-körnek nevezzük.

Az Euler-út és az Euler-kör számos alkalmazással bír a mindennapi életben. Ha valaki egy kiszállítási, begyűjtési (pl. szemetes) útvonalat tervez, megpróbálhat egy Euler-utat vagy Euler-kört konstruálni. Ha talál egyet, akkor minden úton sorban végig mehet anélkül, hogy bármelyiket kétszer használná.

WEBOLDAL

Euler-kör – Wikipédia (wikipedia.org)

Leonhard Euler – Wikipédia (wikipedia.org)

Haus vom Nikolaus – Wikipedia

KULCSSZAVAK

gráf, Euler-kör, Euler-út

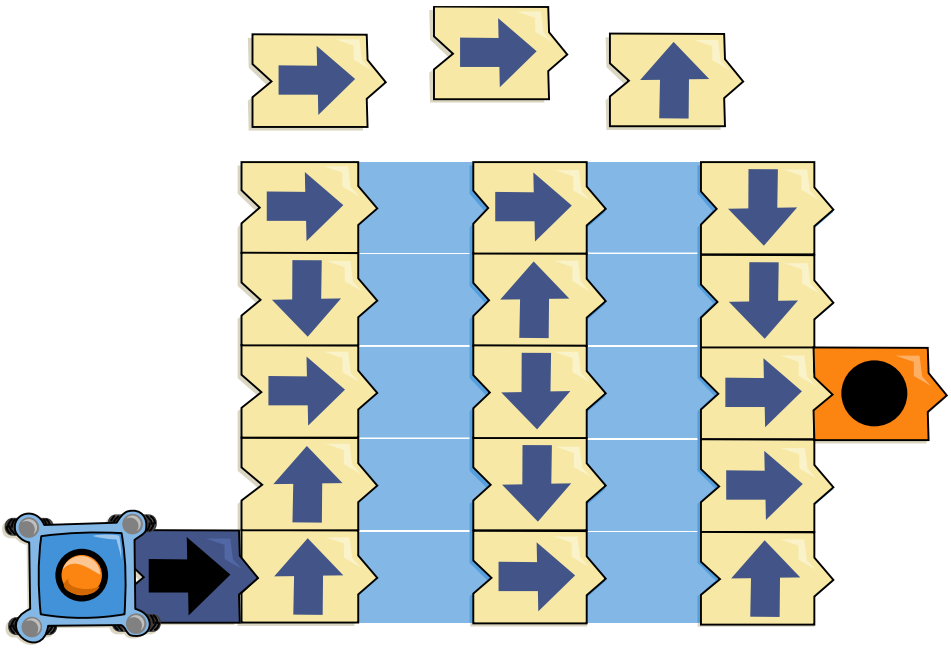
Robotút (2024-PL-04)

KISHÓD – NEHÉZ
BENJAMIN – KÖZEPES
KADÉT – KÖNNYŰ

Bence robotja  kártyákon, adott szabályok szerint mozog:

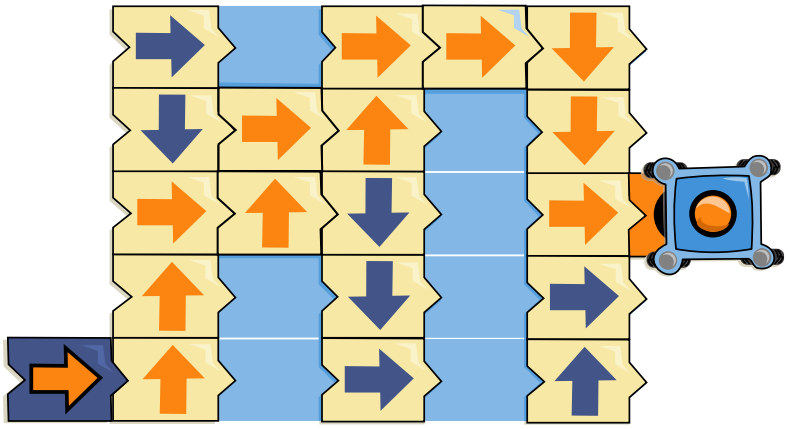
- Az indító kártyán indul el.
- Mindig abba az irányba mozog tovább, amit az a kártya mutat, amelyen áll.
- Ha a robot lefut a kártyákról, akkor megáll.

Néhány kártya az alábbi képen látható módon előre le van helyezve. Bencének három új kártyát kell lehelyeznie úgy, hogy mindhárom extra kártyát az üres területekre helyezze, és a robotja elérje a körrel jelölt célkártyát  .

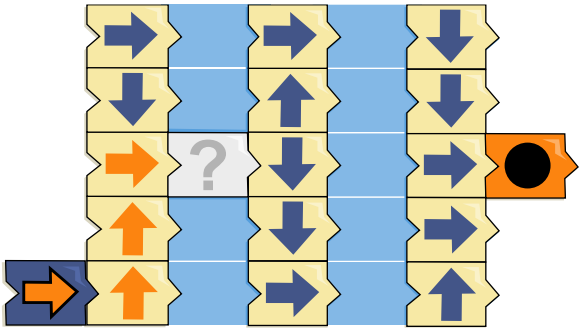


Húzd a három új kártyát a megfelelő helyre.

Megoldás/ A helyes válasz:

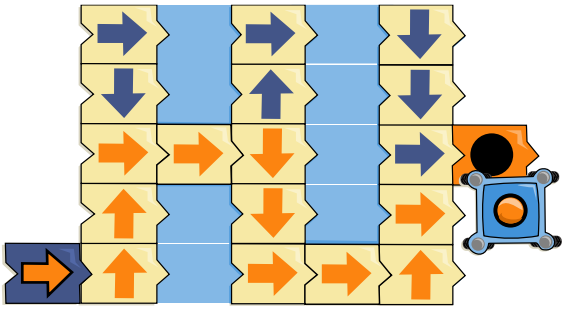


A feladatban leírtak alapján a robot először jobbra megy, ahol elmegy felfele kettő kártyán, majd megint egyet jobbra. Ekkor a „?”-lel megjelölt helyre kerül:

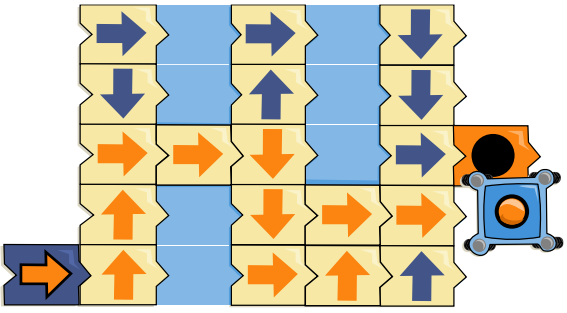


A rendelkezésre álló kártyák (fel  és jobbra ) egyikét ide kell tehát lehelyezni.

Amennyiben a robot jobbra megy tovább (a  kártyát választottuk), kettőt le és egyet ismét jobbra fog menni az előre lerakott kártyákon. Itt megint le kell raknunk egy kártyát (fel  vagy jobbra ). A robot mindkét esetben a cél () alatt lefutna a kártyákról:



vagy



Tehát a felfele mutató nyilas kártyát () kell letennünk a „?” helyére. Ezután már csak a két jobbra mutató nyíl () marad. Ebből az egyiket rögtön le is kell tenni, hiszen a felfele haladás után nem lenne kártya a robot alatt. Az út ezután mutatja magát és az utolsó jobbra mutató nyíl () helyét.

A megoldás visszafele gondolkodva – a céltól haladva visszafele – is megtalálható hasonló nyomkövetéssel.

MIÉRT INFORMATIKA?

Ezt a hódfeladatot általában a válasz magyarázatában leírtak szerint oldjuk meg: kipróbálunk egy utat, és ha az nem vezet a célhoz, visszamegyünk, és megpróbálunk egy másik utat. Az informatikában ezt mélységi keresésnek (bejárásnak) nevezik, mert először egy utat próbálnak ki teljesen, és csak utána próbálkoznak a következővel.

Mivel gyakran egy vagy több lépést kell visszamennünk, hogy egy másik megoldást próbáljunk ki, úgynevezett visszalépést (backtracking) teszünk. A probléma megoldása azonban minden esetben előrefelé történik.

Egyes problémákat azonban jobb visszafelé megoldani úgy, hogy megnézzük, hogyan jutunk el a kiindulástól a célhoz. Tulajdonképpen az ilyen típusú problémákat – amit ez a feladat is mutat – visszafelé nem nagyon lehet megoldani, mert míg előre mindig tudjuk, hogy melyik a következő kártya, addig hátrafelé több kártyáról is érkezhettünk (a nyíl előrefelé mutat, hátrafelé pedig az összes szomszédos kártyát meg kell nézni, hogy a kártyánk irányába mutatnak-e).

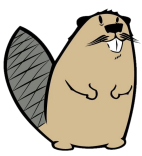
WEBOLDAL

Mélységi keresés – Wikipédia

Visszalépéses keresés – Wikipédia

KULCSSZAVAK

Mélységi keresés, Visszalépéses keresés



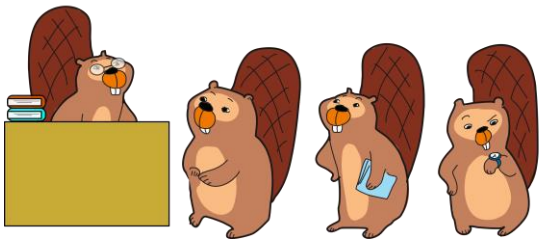
Könyvtár (2024-SI-01)

BENJAMIN – NEHÉZ
KADÉT – KÖZEPES
JUNIOR – KÖNNYŰ
SENIOR – KÖNNYŰ

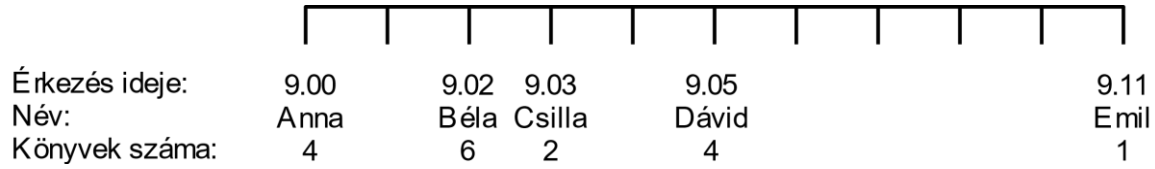
Faváros hódjai szorgalmas olvasók. Ezért van az, hogy általában rengeteg hód áll sorban a könyvtár kölcsönzőpultjánál. A szabály itt is érvényes:

Az a hód, akinek a legkevesebb könyve van, kerül következőnek sorra.

A könyvtárosnak mindig pontosan egy percre van szüksége egy könyv visszavételéhez. Miután feljegyezte a visszahozott könyveket és elrakta őket, a soron következő hód a várakozók közül kerül sorra. Ez mindig az a sorban álló hód lesz, akinek a legkevesebb könyve van, függetlenül attól, mikor érkezett.



Egy reggel 5 hód jön, és vissza akarja adni a könyveit. Az ábrán látható az érkezés időpontja és az egyes hódok könyveinek száma.



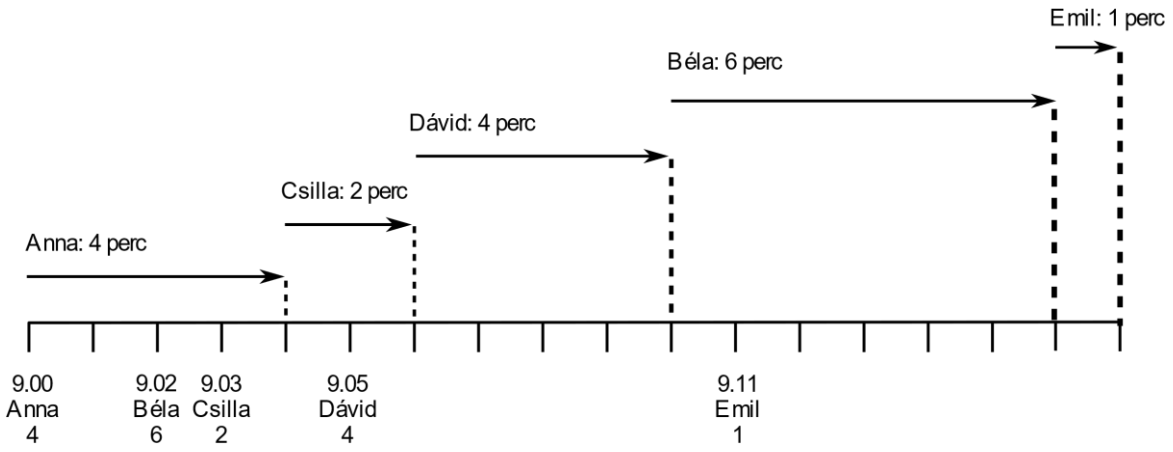
Anna érkezik elsőnek és azonnal visszaadhatja a 4 könyvet, ami nála volt.

Milyen sorrendben adják vissza a könyveket a hódok?

A) Anna, Béla, Csilla, Dávid, Emil
B) Anna, Csilla, Béla, Dávid, Emil
C) Anna, Csilla, Dávid, Béla, Emil
D) Anna, Emil, Csilla, Dávid, Béla

A helyes válasz a C: Anna, Csilla, Dávid, Béla és Emil.

A következő ábra azt mutatja, hogy a könyvek hogyan kerülnek vissza:



- Először Anna kerül sorra. A könyvtárosnak 4 percbe telik, amíg feljegyzi a 4 könyvet. Csak ezután foglalkozhat a következő könyvekkel.

- Közben Béla és Csilla megérkeztek és várnak. Csillának csak 2 könyve van. Így ő következik a sorban. 2 perc múlva az ő könyveit is feljegyzik.

- Időközben megérkezett Dávid is és csatlakozik Bélához. Dávidnak 4 könyve van, Bélának pedig 6. Dávidnak van éppen a legkevesebb könyve, ezért ő következik a sorban. Szegény Bélának még mindig várnia kell. 4 perc múlva a könyvtáros végzett Dávid könyveivel, és végre Béla is sorra kerülhet. Mert ő az egyetlen a sorban.

- Közben Emil is megérkezett. Ő érkezik utolsóként, de utána nincs senki, így ő kerülhet sorra.

MIÉRT INFORMATIKA?

Ez a hódfeladat bemutatja, hogyan működik az informatikai rendszerekben a számítógép operációs rendszerének ütemezője. A számítógép működése közben számos feladatot kell feldolgozni. Minden egyes feladathoz elindul egy folyamat, amely megoldja a feladatot. Az ütemező vezérli a folyamatok sorrendjét. Meghatározza, hogy egy folyamatot mikor és mennyi ideig hajtson végre a számítógép központi feldolgozó egysége (CPU). A példában szereplő feldolgozással ellentétben előfordulhat, hogy megszakít egy feladatot, végrehajt egy másikat és csak ezután fejezi be az előzőt.

Különböző stratégiák vannak, amelyek szerint az ütemezők működnek:

- Prioritási ütemezés - A legmagasabb prioritású folyamatot hajtja végre legközelebb (teljes egészében).

- First Come First Serve (FCFS) - A sorba elsőként bejutott folyamatot hajtják végre először (teljesen). A legtöbb sorban állás a mindennapi életben ezen elv szerint működik („aki először jön, azt szolgálják ki először”).

- Round-robin - a várakozó folyamatok csak sorban, egy bizonyos ideig kerülnek végrehajtásra. Ha egy folyamat még nem fejeződött be, megszakad, és vissza kell térnie a sor végére. Ezután a következő folyamat kerül sorra.

- Shortest Job First (SJF) - a legrövidebb várható munkaidővel rendelkező folyamatot hajtják végre először (teljes egészében). Ebben a feladatban a Shortest Job First stratégiát alkalmaztuk, mert a legkevesebb könyvvel rendelkező (és a leggyorsabban feldolgozható) hódot szolgáltuk ki legközelebb.

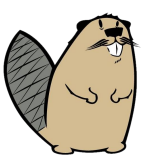
WEBOLDAL

Sor (adatszerkezet) – Wikipédia

Ütemező programtervezési minta – Wikipédia

KULCSSZAVAK

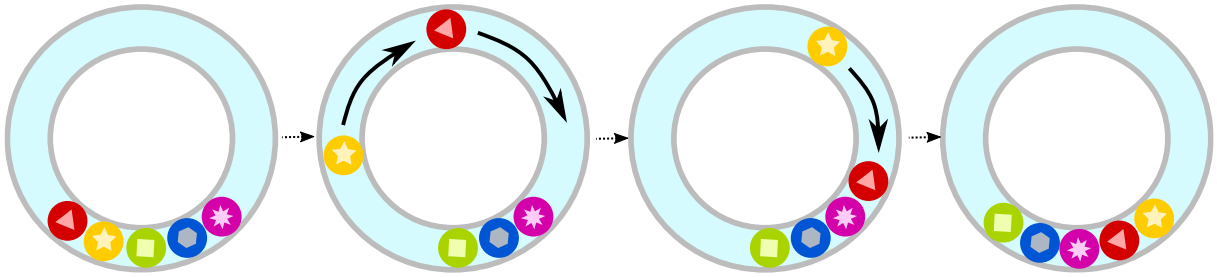
Prioritási sor, Ütemező



Csörgő (2024-SK-01b)

KISHÓD – KÖZEPES
BENJÁMIN – KÖNNYŰ

Olivérnek van egy átlátszó csörgője színes golyókkal. Amikor megrázza, néhány golyó átkerül a másik oldalra, ahogy a képen is látható.



Olivér még egyszer megrázza a csörgőt.

Vajon most hogy néz ki a csörgő?

Húzd a hiányzó golyókat a fehér helyekre, ahogyan Olivér csörgőjében kell lenniük.

A helyes válasz:

A golyók nem változtatják meg a sorrendjüket attól, hogy megrázzuk a csörgőt. A piros golyó egyik oldalán a sárga golyó van, ami mellett a másik oldalon a zöld. A zöldet a kék követi, majd a lila következik. Végül ismét a piroshoz jutottunk.

A megoldás ennek a sorrendnek a visszaállítása.

MIÉRT INFORMATIKA?

Az informatikusok különböző módon reprezentálják és tárolják az adatokat. Minden adatreprezentációnak vannak előnyei, amelyek az adott adat tulajdonságaitól függenek.

Ebben a feladatban egy úgynevezett körkörösen összekapcsolt listát használunk, amelyet „csörgőnek” nevezünk, és a tárolt elemek a golyók. Ebben a körkörösen összekapcsolt listában minden elemnek van egy úgynevezett mutatója a következő elemre. Továbbá az utolsó elem mutatója visszamutat az első elemre, és így zárja be a kört.

Ez azt jelenti, hogy ebben a feladatban minden egyes golyó esetében tudjuk, hogy melyik golyó következik. Az utolsó golyó után pedig az első. Amikor a golyók sorrendjét ellenőrizzük, bármelyik golyótól indulhatunk, és az összeset (a teljes listát) a helyes sorrendben kell végigjárnunk.

Két előnye van annak, ha körkörösen összekapcsolt listát használunk az elemek tárolására: Az első, hogy a körkörösen összekapcsolt listában nincs végpont. Mivel a lista visszafordul az elejére, az elemek soha nem fogynak ki az ellenőrizhető vagy felhasználható adatokból. A második előny a listában való navigálás hatékonysága. Könnyen lehet haladni a listán anélkül, hogy a végpontra vonatkozó speciális szabályokat kellene követni. Egyszerűen csak haladunk tovább egy irányba.

WEBOLDAL

Láncolt lista – Wikipédia

Adatszerkezet – Wikipédia

KULCSSZAVAK

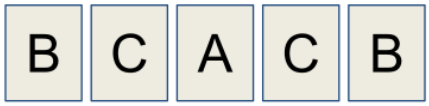
Láncolt lista, Lista, Körkörös lista, Adatszerkezet

Legnagyobb értékű sorozat (2024-SK-04)

JUNIOR – NEHÉZ
SENIOR – KÖZEPES

Rakd a betűket egy sorba, és szerezz pontokat: Ha 2 azonos betű közvetlenül egymás után van, akkor 2 pontot kapsz, ha 3 azonos betű van közvetlenül egymás után, akkor 3 pontot kapsz és így tovább.

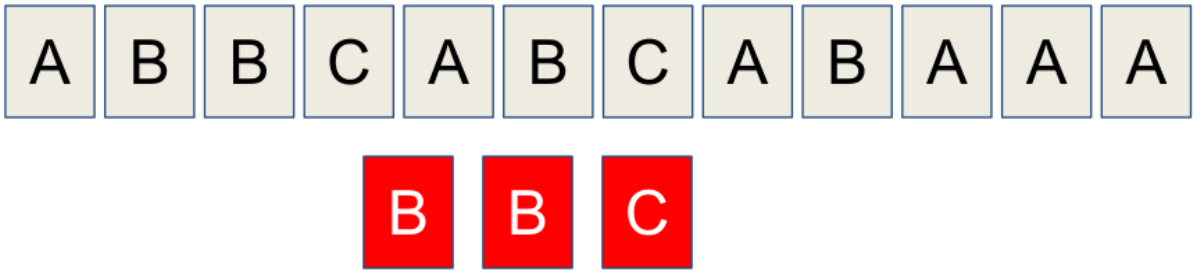
Egy példa: Erre a sorra 0 pontot kapsz, mert nincsenek azonos betűk közvetlenül egymás után:



Ha az első C betűt A betűre, a második C betűt pedig B betűre cseréljük. 4 pontot kapunk: 2 pontot a két egymás utáni A betűért és 2 pontot a két egymás utáni B betűért a sorban.

Kezdjünk egy új sort 12 betűvel. Ebben bármelyik három betűt kicserélheted a B, B és C betűkre.

A betűket úgy cseréld ki, hogy a lehető legtöbb pontot kapd érte.



A cseréhez húzd a betűket a kicserélendő betűre.

Megoldás/ A helyes válasz:

Erre a sorra 2 + 2 + 4 + 3 = 11 pontot kapsz; mivel balról jobbra haladva 2 db B, 2 db C, 4 db B és 3 db A van egymás után. Több pontot nem kaphatsz:

- A sorozat 12 betűből áll, ami azt jelenti, hogy maximum 12 pontot szerezhetsz.
- A bal szélső A azonban nem tartozhat egy csoportba, mert nincs A, így a jobb oldali B-t nem lehet A-val helyettesíteni.
- Összesen 3 A betű van elkülönítve. Mivel nincs A-nk, hogy összehozzuk ezeket, ki kell cserélnünk mindet ahhoz, hogy eltűnjenek és teljes lehessen a sorunk. A három A-t nem tudjuk úgy kicserélni a 2db B és egy C betűvel, hogy legalább egy C ne maradjon egyedül

Tehát a maximális 12 pont nem rakható össze.

Van-e más olyan csere, amely 11 pontot adhat? Ahhoz, hogy minél több pontot kapj, a B, B és C betűket kell olyan sorba rendezni, ahol új csoport jön létre, és egyetlen meglévő csoport sem szűnik meg vagy csökken le a hossza:

- Ha egy betű egy meglévő csoport szélére kerül, akkor hozzáadunk 1 pontot.
- Ha azonban egy betű egy olyan mellé kerül, hogy új csoportot hoz létre (korábban nem volt mellette ugyanolyan betű), vagy egy másik betűt is hozzákapcsol egy meglévő csoporthoz, akkor 2 pontot adunk hozzá.
- Ha egy betűt két egyedül álló betű közé helyezünk, akkor egy új, három betűből álló csoport jön létre, ami 3 pontot ér.

Kezdetben a sor 5 pontot ér. A B, B és C betűvel nem lehet 3 pontot szerezni. 11 pont megszerzéséhez a B, B és C betűket úgy kell elhelyezni, hogy minden alkalommal egy új, két betűből álló csoport jöjjön létre. Ehhez csak két egyedül álló B és két egyedül álló C áll rendelkezésre. Csak egy lehetőség van arra, hogy három új két betűből álló csoportot alakítsunk ki B-vel, B-vel és C-vel anélkül, hogy csökkentenénk a meglévő csoportok méretét: ez pedig a megoldásunk.

MIÉRT INFORMATIKA?

Ez a példa megmutatja, hogyan járnak el többnyire az informatikusok, amikor optimális megoldást keresnek.

A feladat lényege, hogy a B, B és C betűk olyan elhelyezését keressük, amely a legtöbb pontot hozza. Egy első megközelítés lehet egyszerűen az összes olyan lehetőség generálása, amelyben a sorozat három betűjét B, B és C betűvel helyettesítjük. Egy tizenkét betűből álló sorozat esetében 12 * 11 * 10 = 1320 lehetőség van, amelyek mindegyikére ki

kell számítani a pontszámot, és össze kell hasonlítani a legjobb megoldás meghatározásához.

A megoldások száma azonban jelentősen csökkenthető, ha figyelembe vesszük, hogy a B, B és C csak akkor kap pontot, ha ugyanazon betűk mellett állnak. Ha azt is figyelembe vesszük, hogy a meglévő csoportokat nem szabad csökkenteni, a megoldások száma tovább csökken. Ha azt is figyelembe vesszük, hogy az egyes betűk összekapcsolása több pontot hoz, mint a meglévő csoportok bővítése, akkor csak egy lehetséges elhelyezés marad.

A fent említett 1320 lehetőség nem sok egy számítógép számára, azonban az informatikában mindig felmerül a kérdés, hogy a feladat megoldásához szükséges „erőfeszítés” hogyan változik a probléma méretének növekedésével. Hosszabb sorok és több váltakozó betű esetén ennek megfelelően a megoldás hosszabb időt jelentene és több erőforrást igényelne. Érdemes alaposan elemezni a problémát, és hatékony módszert kidolgozni.

Az informatika szempontjából ez egy optimalizálási probléma. Az informatika területén az optimalizálásnak döntő szerepe van. Arról szól, hogy valamit bizonyos korlátok között a lehető leghatékonyabbá vagy legműködőbbé tegyünk, a lehetséges megoldások közül a legjobbat találjuk meg. Amikor például egy fordítóprogramban a kód optimalizálásáról beszélünk, akkor a kód módosításának folyamatára utalunk, hogy az hatékonyabbá és kisebbé váljon anélkül, hogy a kimenetei megváltoznának. Ez azért fontos, mert így a programok gyorsabban futnak és kevesebb erőforrást használnak. Hasonlóképpen az adattömörítés is az optimalizálás egy másik példája. Ennek során az információt az eredeti reprezentációnál kevesebb bit felhasználásával kódolják, hogy tárhelyet takarítsanak meg, vagy csökkentsék a továbbítandó adatok mennyiségét.

WEBOLDAL

Programoptimalizálás – Wikipédia

Matematikai optimalizálás – Wikipédia

KULCSSZAVAK

Optimalizálás

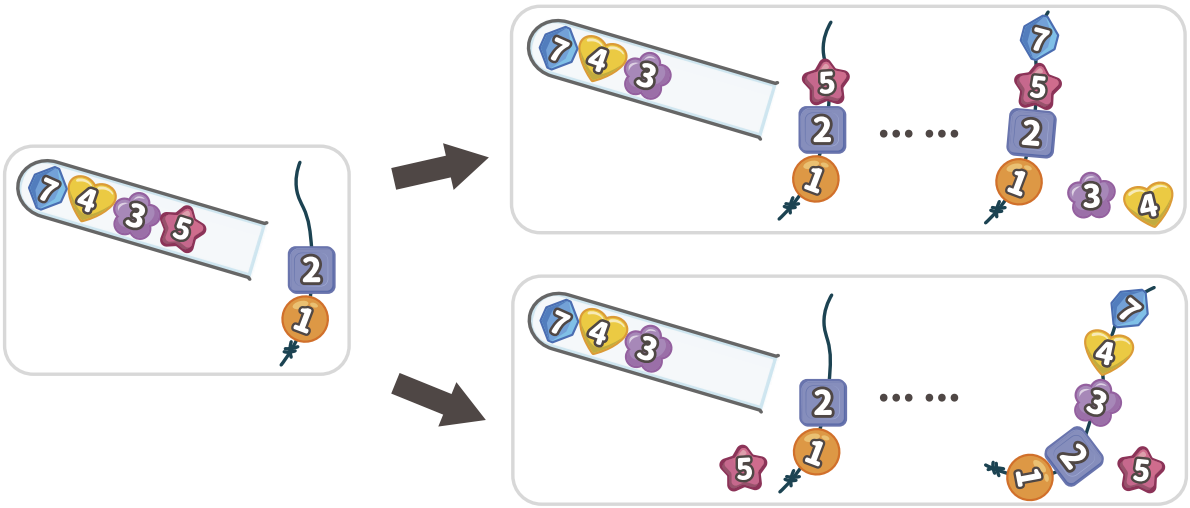
Karkötő (2024-TW-03)

KADÉT – NEHÉZ
JUNIOR – KÖZEPES
SENIOR – KÖNNYŰ

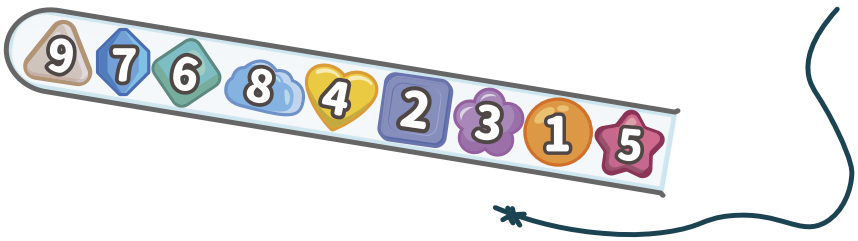
Sanyi karkötőt készít. Egy üvegcsőből számokkal ellátott gyöngyöket vesz elő. Néhányat felfűz közülük, néhányat pedig félretesz és nem használja fel őket. Csak akkor fűz fel egy gyöngyöt, ha a zsinór üres vagy a gyöngyön nagyobb szám van, mint a zsinór utolsó gyöngyén.

Ebben a példában a zsinór utolsó gyöngyén a 2-es van. Ezután Sanyi a csőből az 5-ös gyöngyöt veszi ki. Fel is fűzheti vagy félre is teheti, és nem használja fel.

Ha felfűzi az 5-ös gyöngyöt, akkor már csak egy négy gyöngyből (1257) álló karkötőt készíthet. Ha nem használja az 5-öst, akkor felfűzheti a 3-as és 4-es gyöngyöt és így hosszabb, öt gyöngyből (12347) álló karkötőt készíthet.



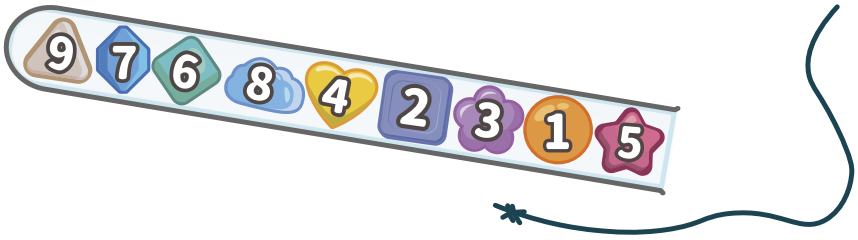
Sanyi új karkötőt kezd a képen látható gyöngyökből:



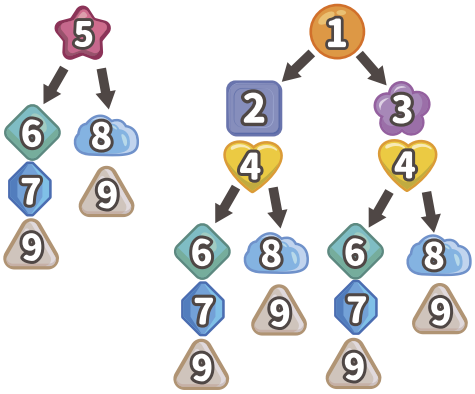
Hány gyöngyből állhat a leghosszabb létrehozható karkötő?

Megoldás/ A helyes válasz a 6.

Összeállíthatjuk az összes lehetséges karkötőt, ami a csőben lévő gyöngyökből felfűzhető, és megnézhetjük, melyikben van a legtöbb gyöngy. De ez időigényes lenne. Nézzük meg közelebbről a számokat:



Az első gyöngy az 5-ös számú. Ha fel van fűzve, csak a 6, 7, 8 és 9 számmal rendelkező gyöngyök fűzhetőek már fel, tehát a lehetséges sorozatok 5679 és 589 lennének. Ha az 5-ös gyöngyöt félretesszük, és a következő, 1-es gyöngyöt fűzzük fel, akkor több és hosszabb lehetséges sorozatunk lehet. Ezt le is rajzolhatjuk egy úgynevezett döntési faként:



Ha az 1-es gyöngyöt is félretesszük, és először egy másik gyöngyöt fűzünk fel, a lehetséges gyöngysorozatokat már tartalmazza a fenti sorozatunk egyike. Ha például a 2-es gyönggyel kezdünk, akkor 24679 vagy 2489-es karkötőket (sorozatokat) hozhatjuk létre, amelyeket a 124679, illetve 12489 hosszabb sorozatok (karkötők) tartalmaznak.

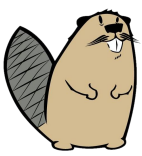
A feladatunk már csak megnézni, melyik sorozat a leghosszabb. Ez pedig a 6 gyöngyből álló 124679 vagy 134679.

MIÉRT INFORMATIKA?

Ebben a hódfeladatban minden egyes üvegcső számozott gyöngyök sorozatát tartalmazza, amit számsorozatként is felfoghatunk. Ahogy Scott a gyöngyöket felfűzi, minden egyes karkötője a „csősorozat” növekvő részsorozata; a növekvő azt jelenti, hogy a részsorozatban minden szám (az első szám kivételével) nagyobb, mint az előtte lévő szám. Ahhoz, hogy minél több gyöngyből álló karkötőt készítsen, Scottnak meg kell határoznia a leghosszabb növekvő részsorozatot.

Hosszú számsorozatok esetén sok időbe telne az összes lehetséges növekvő részsorozatot megalkotni (akár a megoldásban mutatott lehetséges összeállításokat egy döntési fában ábrázolni), hogy meghatározzuk a leghosszabbat. Ha a cső például 20 gyöngyöt tartalmaz, a számítási erőfeszítés egymillió lépés nagyságrendű lenne.

Szerencsére az informatikusok olyan algoritmusokat fejlesztettek ki, amelyek gyorsabban képesek meghatározni a leghosszabb növekvő részfolyamatot. Ehhez egy dinamikus programozásnak nevezett technikát használnak. A módszer lényege, hogy a kiindulási problémát részproblémákra bontjuk és a részproblémák megoldásaival fejezzük ki a



megoldást. Ezekkel a gyors algoritmusokkal a 20 gyöngyöt tartalmazó cső esetében a számítási erőfeszítés kevesebb mint száz lépés.

WEBOLDAL

Döntési fa – Wikipédia

Richard Bellman – Wikipédia

KULCSSZAVAK

Döntési fa, Dinamikus programozás