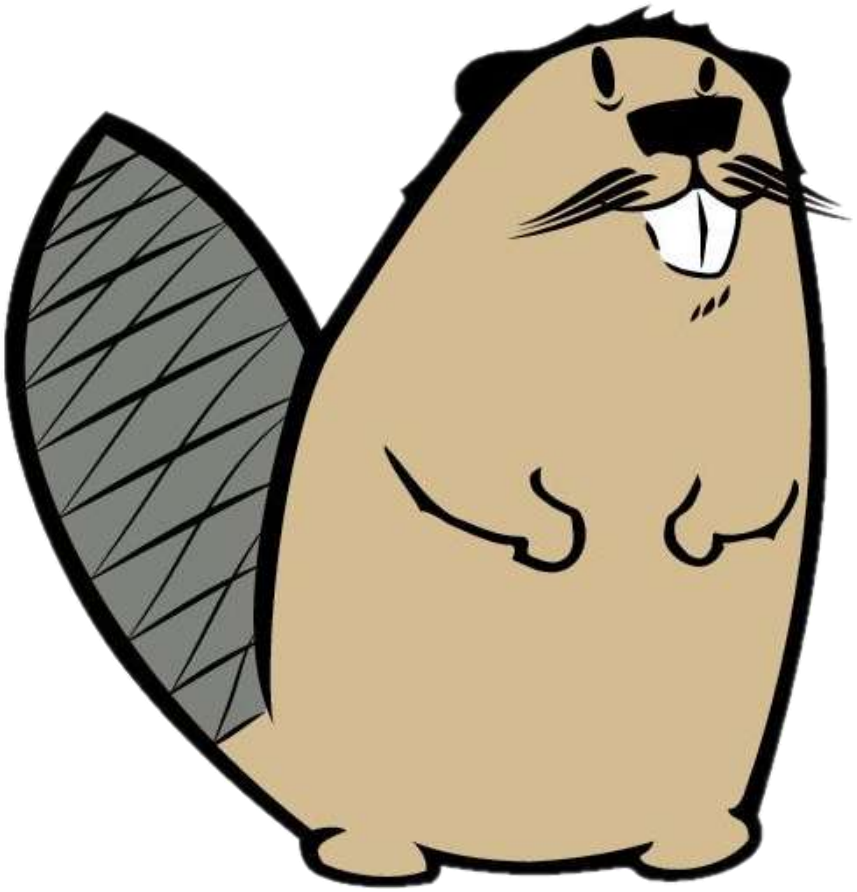
HÓDÍTSD MEG A BITEKET!

INFORMATIKAI GONDOLKODÁST TÁMOGATÓ, NEMZETKÖZI

BEBRAS KEZDEMÉNYEZÉS MAGYAR MEGVALÓSULÁSA



2020

MI IS AZ E-HÓD

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

A nemzetközi Bebras, melyhez 2019-ben már 50 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 2020-ban tizedik alkalommal, öt korcsoportban vehetnek részt a diákok 4-től 12. osztályig.

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

Az alábbi dokumentumban a 2020-as magyar verseny feladatai és megoldásai találhatóak.

További információkért látogasson el a HTTP://E-HOD.ELTE.HU/ 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 14:00 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 11 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 kapcsolattartónak kell összegyűjtenie és továbbítania a szervezők felé;
- A verseny célja: minél több pont összegyűjtése helyes válaszok megjelölésével, helytelen válaszok esetén 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 11, 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 162 pont érhető el.

TARTALOMJEGYZÉK

TARTALOMJEGYZÉK

FELADATOK

Kishód: 2020-IS-02, 2020-LV-02, 2020-PT-06, 2020-SK-04
2020-CH-04b, 2020-JP-04, 2020-LT-08, 2020-PK-05
2020-CH-21, 2020-JP-02, 2020-Sk-01, 2020-TR-03

Benjámin: 2020-CH-09b, 2020-CH-21, 2020-JP-02, 2020-SK-01, 2020-SK-04, 2020-TR-03
2020-AT-02, 2020-CA-02, 2020-CH-04b, 2020-JP-04, 2020-LT-08, 2020-PK-05
2016-DE-02, 2016-HU-06, 2020-DE-08b, 2020-HU-04, 2020-HU-07a, 2020-PK-06

Kadét: 2020-AT-02, 2020-CH-04b, 2020-CH-18, 2020-JP-04, 2020-LT-08, 2020-PK-05
2016-DE-02, 2016-HU-06, 2020-DE-08b, 2020-HU-04, 2020-HU-07a, 2020-PK-06
2020-DE-02, 2020-DE-05a, 2020-KR-01, 2020-MK-03, 2020-PH-02a, 2020-VN-01

Junior: 2016-DE-02, 2016-HU-06, 2020-DE-08b, 2020-HU-04, 2020-HU-07a, 2020-PK-06
2020-CH-04c, 2020-DE-02, 2020-DE-05a, 2020-KR-01, 2020-MK-03, 2020-PH-02a
2013-AT-08, 2017-DE-09, 2020-CZ-03, 2020-DE-04b, 2020-DE-06a, 2020-NZ-03

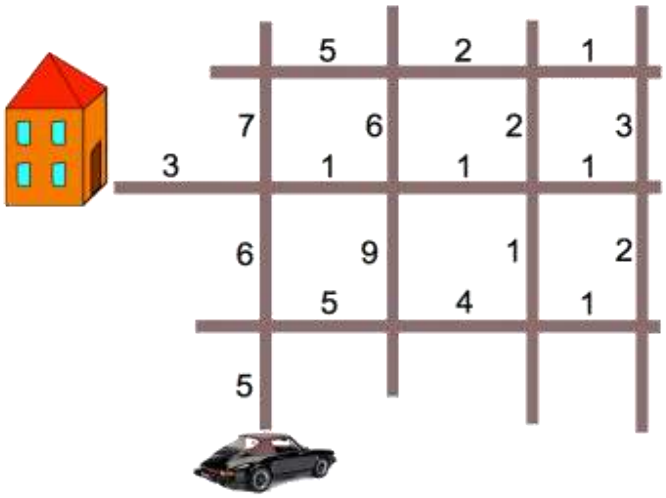
Senior: 2020-CH-04c, 2020-DE-02, 2020-DE-05a, 2020-MK-03, 2020-PH-02a, 2020-VN-01
2013-AT-08, 2017-DE-09, 2020-CZ-03, 2020-DE-04b, 2020-DE-06a, 2020-NZ-03
2016-CA-09, 2017-RU-05, 2020-AT-01, 2020-HU-01, 2020-IT-02, 2020-RU-02

SOSE FORDULJ BALRA! (2013-AT-08)

JUNIOR - NEHÉZ

SENIOR - KÖZEPES

Végtelen a szembeforgalom – gyakorlatilag lehetetlen ezeknél a kereszteződéseknél balra fordulni. Ha az autós gyorsan haza akar jutni, olyan útszakaszt kell választania, ahol sosem kell balra fordulnia. A képen percekben megadva láthatod azt az időtartamot, amire az autósnak egy-egy útszakasz megtételéhez szüksége van.



Minimum mennyi időre van szüksége az autósnak, ahhoz hogy hazaérjen, ha sosem fordulhat balra?

A) 35 percre

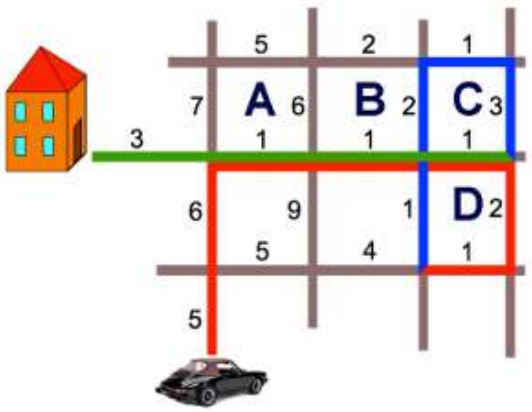
B) 33 percre

C) 32 percre

D) 30 percre

A HELYES VÁLASZ A D)

A leggyorsabb út a D blokk, majd a C blokk megkerülésével történik. Ehhez 30 percre van szükség:



5+6+1+1+1+2+1 a piros szakaszon

1+2+1+3 a kék szakaszon

és 1+1+1+3 a zöld szakaszon.

Minden más úton hosszabb a menetidő. Az A blokk megkerüléséhez 33 percre van szükség (A válasz). Az A és a B blokk megkerülése 32 percig tart (B válasz). Az A, a B és a C blokk megkerülésére 35 percre van szükség (C válasz).

MIÉRT INFORMATIKA?

Az informatikában gyakori probléma a minimális ráfordítást igénylő út keresése, ráadásul sokszor bizonyos keretfeltételeket is figyelembe kell venni. A feladatban például ilyen keretfeltétel az egyes útszakaszok menetideje, illetve az a kikötés, hogy nem tudunk balra fordulni. Gyakran túl nagy a szóba jöhető utak száma ahhoz, hogy mindegyiket egyesével megvizsgálják és megállapítsák, hogy az-e a minimális ráfordítást igénylő út. Ilyenkor meg kell próbálni a megvizsgálandó utak számát észszerűen szűkíteni. A feladatot néhány környékbeli háztömbre korlátoztuk. Ezzel kockára tesszük, hogy egy 30 percnél kevesebb időráfordítást igénylő utat találjunk, amely a mi kutatási horizontunkon (a képünkön) kívül vezet.

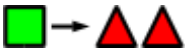
ALAKZATJÁTÉK (2016-CA-09)

SENIOR - NEHÉZ

Aliz geometriai alakzatokkal játszik. Egy-egy alakzatsort átvált alakzatok egy másik sorára.

Aliz minden játékkörben saját átváltási szabályokat rögzít. Minden átváltáskor a szabályt olyan sokszor alkalmazza, ahányszor csak lehetséges. Minden játékkört egy alakzattal kezd.

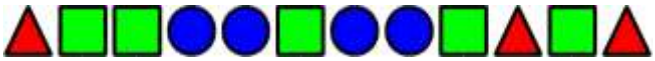
Az egyik korábbi körben Aliz letett egy négyzetet és a következő váltási szabályokat alkalmazta:



és

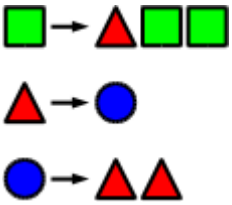
Három lépésben a jobb oldali alakzatsort hozta létre:

Egy másik körben Aliz a következő alakzatsort hozza létre:

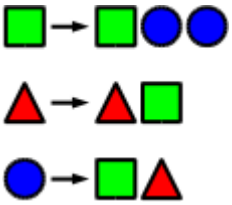


Az alábbi szabályrendszerek közül melyik lehet érvényben, ha a fenti kártyasorozat jött ki és tudjuk, hogy a kiindulópont a három geometriai alakzat egyike volt?

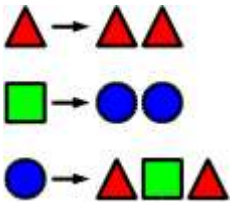
A)



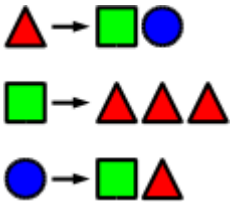
B)



C)



D)



A HELYES VÁLASZ A B)

B) Aliz háromszöggel kezdett és három lépésben jutott el az alakzatsorhoz:

A többi válaszlehetőség a következő átváltások megfontolásával kizárható:

A) Háromszöggel vagy körrel kezdve soha nem kaphat négyzetet. Négyzettel kezdve a következő történik:

Mivel minden egyes átváltás növeli a sorhosszt, nincs lehetőség a kívánt eredmény elérésére.

C) Háromszöggel kezdve Aliz nem kaphat sem négyzetet, sem kört. Négyzettel kezdve a következő történik:

A sorkezdeten álló dupla háromszöget már soha többé nem lehet háromszög-négyzet párossá átalakítani, pedig ez lenne szükséges a cél eléréséhez. Körrel kezdve is előjön a kezdő dupla háromszög feloldhatatlan problémája:

D) Aliznak semmiképpen sem adódik lehetősége két kört egymás mellé helyezni, pedig ez lenne a kívánatos cél.

MIÉRT INFORMATIKA?

Az alakzatjáték szimbólumai és szabályai együtt a szimbólumláncok egy átváltásrendszerét alkotják. Az elméleti informatikában nagy jelentőséggel bírnak az ilyen átváltási rendszerek. Az átváltásrendszereket nyelvtannak is szokás nevezni. A szimbólumsorozatok, amint itt az alakzatjátékban is, előállíthatók a szabályok alkalmazásával, és így egy nyelv szavai lesznek. Az elméleti informatikában egy nyelv egyben szavak tömege is, a nyelvtan pedig előállít egy ilyen formális nyelvet.

Minden nyelvtan teret ad a szóprobléma megfogalmazódásának: egy adott szó az adott nyelvtannal jellemezhető nyelvhez tartozik-e vagy sem? A szóprobléma megoldásának nehézségi fokát döntően befolyásolja, hogy a nyelvtan szabályai milyen formát öltöttek. A szabályok formájától függően a nyelveket különféle tulajdonságokkal rendelkező osztályokba sorolják. Például:

- A rendezett nyelvtanok olyan nyelveket hoznak létre, amelyek szabályai előírásokká fogalmazhatók és végső soron automatizálhatók. Az ilyen rendezett nyelvtanok, ill. kifejezések rendkívül hasznosak pl. keresési algoritmusok felállításánál.
- A kontextusfüggetlen nyelvtanok olyan nyelveket hoznak létre, amelyekben a szóprobléma hatékonyan megoldható. A programnyelvek nyelvtana éppen ezért rendszerint kontextusfüggetlen. Az összeállítók által elfogadott program érvényesül formális szempontból a nyelv szavainál.
- A korlátmentes nyelvtanok olyan nyelveket hoznak létre, amelyekben a szóprobléma csak részlegesen oldható meg. Egy adott szó korlátmentes nyelvtannal jellemezhető nyelvhez tartozását alkalmasan tervezett Turing-géppel bizonyíthatjuk. Ugyanaz a Turing-gép nem lesz alkalmas az adott nyelven kívüli szóra.

HÓDCHAT (2016-DE-02)

BENJAMIN - NEHÉZ

KADÉT - KÖZEPES

JUNIOR - KÖNNYŰ

A Hódchat egy ingyenes alkalmazás, mely reklámokból tartja fenn magát. A Napsütés Utazási Iroda igyekszik a célcsoportnak megfelelő reklámot választani. Minden üzenetekről automatikus értékelést készítenek, melyekben meghatározott szavakra keresnek és ezekhez pontszámokat rendelnek.

- Például valamely megszokott üdvözlés az ifjú hódok között (`Hello, Hi, Cső, Csá`) **-2** pontot ér.
- Az idősebbek higgadt megszólításai (`Kedves, Tisztelt`) ezzel szemben **2 pontot** jelentenek.
- Minden rövidítés (`lol, yolo, pill, tom, vki`) **-1 pont**.
- Minden 10 betűs vagy annál hosszabb szó **1 pont**.

A hódok üzeneteik összpontszáma alapján besorolódnak a célcsoportok egyikébe:

Pontszám	Célcsoport	Mutatott kép
Pozitív	Senior	Strandkép
Negatív	Fiatal	Szörfös kép
0	Nincs besorolás	A párizsi Eiffel-torony

Az alábbi üzenetek közül melyikhez fogja az értékelő a párizsi Eiffel-tornyot hozzárendelni?

A) Sziasztok Kedves Barkácstársak, felteszem a mai kérdésemet, használnátok-e horganyzott csavarokat egy víz alatti projektben? Ricsi
B) Hi, Nem tom, bocsi. Rozsdálnak a cuccok?
C) Cső, vki?
D) @Liza: <3 <3 <3

A HELYES VÁLASZ A D)

Az értékelésnél semmilyen támpontot nem találunk, így nem tudunk pontot adni.

Az A) üzenetben teljes megszólítás és hosszabb szavak is szerepelnek. Ezzel mindenképpen pozitív értékelést ér el. Így a strandkép jelenik majd meg. A B) és a C) üzenetben is rövidítések és „fiatalos" megszólítások szerepelnek, ezért több mínusz ponttal végeznek. Így a szörfös kép jelenik majd meg.

MIÉRT INFORMATIKA?

Egy szöveg meghatározott szabályok alapján történő kiértékelése könnyen elvégezhető számítógépes programokkal. Szövegrészletek keresése egyszerű példa a minta-egyezéses keresésre („pattern matching" angolul). Ez nem csak a szövegfeldolgozásnál fontos, de pl. a képfeldolgozás területén is számos számítógépes alkalmazás használja. Társadalmi fontosság: Az internetes felhasználói profilok manapság számtalan automatizált vizsgálatban kerülnek kiértékelésre, például azért, hogy a „megfelelő" a vásárlói ajánlatokkal kereshessék meg a felhasználókat. Ezért is egyre fontosabb, hogy az internethasználók tudatosak legyenek és a személyes adataikat figyelmesen kezeljék, adják ki. Mindannyian döntéshelyzetben vagyunk: amennyiben kiszolgáltatunk magunkról adatokat, azt fel tudják használni előrejelzésekhez, segíthetik a megelőzést több területen is, mint pl. az egészségügy, bűnüldözés, …. Ha vigyázunk az adatainkra, akkor viszont csökkentjük annak kockázatát, hogy visszaélhessenek vele, kárt okozhassanak nekünk vagy akár hozzátartozóinknak.

BONBONIER (2016-HU-06)

BENJAMIN - NEHÉZ

KADÉT - KÖZEPES

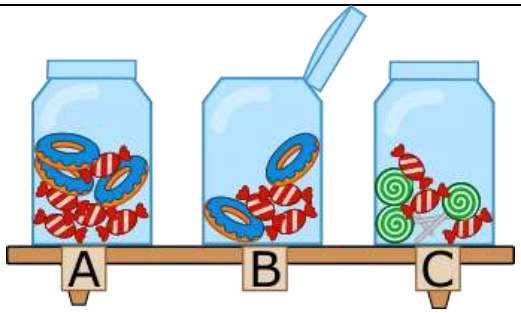
JUNIOR - KÖNNYŰ

Károlynak és Juditnak van 3-3 üvegedénye, melyekben édességeket tartanak. Minden üvegedény a következő tíz tulajdonságok közül többel is rendelkezhet:

- Vagy nyitva (1) vagy zárva (2) van.
- Vagy tartalmaz piros-fehér csíkos cukrot (3) vagy nem (4).
- Vagy tartalmaz kék cukorfánkot (5) vagy nem (6).
- Vagy tartalmaz zöld spirál-nyalókát (7) vagy nem (8).
- Vagy gömbölyű (9) vagy szögletes (10).

Károly „A" üvegedényének például a 2, 3, 5, 8 és 10-es tulajdonságai vannak.

Nézd meg jól! Károly üvegedényeinek vannak közös tulajdonságaik. Judit üvegedényeinek is vannak közös tulajdonságaik.

Károly üvegedényei	Judit üvegedényei
	

Egy üvegedénynek azonban Károly üvegedényeinek közös tulajdonságai és Judit üvegedényeinek közös tulajdonságai is megvannak. Melyik ez az edény?

A HELYES VÁLASZ: A C JELŰ EDÉNY

Károly üvegedényeinek (A, B és C) két közös tulajdonsága van:

- Az üvegedények szögletesek (10).
- Az üvegedények tartalmaznak piros-fehér csíkos cukrot (3).

Judit üvegedényeinek (D, E és F) ugyancsak két közös tulajdonságuk van:

- Az üvegedények tartalmaznak zöld spirál-nyalókát (7).
- Az üvegedények zárva vannak (2).

Csak a „C" jelű edénynek van meg mind a 2, 3, 7 és 10-es tulajdonsága: zárva van, szögletes, tartalmaz piros-fehér csíkos cukrot és zöld spirál-nyalókát. A 6-os tulajdonsággal is rendelkezik, az most számunkra lényegtelen.

MIÉRT INFORMATIKA?

Az informatika az adatok modellezésénél tulajdonságok alapján csoportosítja össze az objektumokat (elemeket). Ebben a feladatban tíz tulajdonsággal és két hármas csoporttal volt dolgunk. Olyan objektumot keresünk, melynek a tulajdonságai mindkét csoportra jellemzőek. A relációs adatbázisoknál ezt úgy is nevezik: „két halmaz metszetét képezzük".

Nagy, átláthatatlan adathalmazokból leszűrhetjük a kívánt tulajdonságok alapján keresett objektumokat („részhalmazát képezhetjük").

Például egy webáruházban célzottan kereshetünk meghatározott kijelző mérettel rendelkező okostelefonra, de tovább szűkíthetjük a keresésünk az összes tárolt tulajdonság alapján is.

Egy adatbázis felépítésénél kiemelt figyelmet kell arra fordítani, milyen tulajdonságok kerülnek elő az adatmodellben. Ha fontos tulajdonságokat kifelejtünk, az összes későbbi keresés kevésbé lesz hatékony. Ha felesleges tulajdonságokat is modellezünk, a későbbi karbantartás, tárolás költségesebb lesz anélkül, hogy a hasznosságot növeltük volna.

BRINGÁRA FEL! (2017-DE-09)

JUNIOR - NEHÉZ

SENIOR - KÖZEPES

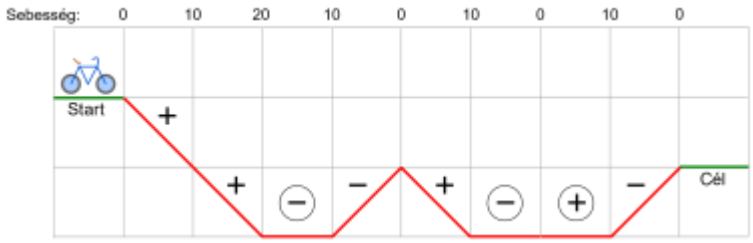
A bringázás egy óriási szórakozás. Speciális kerékpárokkal különböző pályákat lehet bejárni. Mindegyik pálya szakaszok sorozatából áll. Egy szakasz iránya „hegyre fel", „hegyről le" vagy pedig „vízszintesen" lehet.

Egy pálya teljesítése az alábbi szabályok szerint történik:

- A rajtnál 0 km/h a kerékpáros sebessége.
- Lefelé menetben 10 km/h-val nő a kerékpár sebessége.
- Hegyre felfelé tartó szakaszon a kerékpár sebessége 10 km/h-val csökken.
- Vízszintes szakaszon Te magad döntheted el, hogy a sebesség nőjön vagy csökkenjen 10 km/h-val.

A célban újra 0 km/h legyen a kerékpáros sebessége. Menet közben lehet 0 km/h a sebesség, azonban azonnal növelni kell, különben a kerékpáros elesik és nem tudja befejezni a pályát.

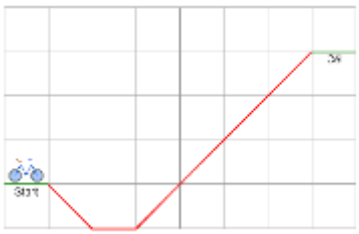
Itt látható egy pálya vázlata, mely ezek alapján a szabályok alapján megtehető. A célba éréshez a vízszintes



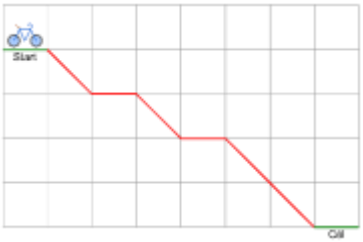
szakaszokon úgy kell változtatni a sebességet, ahogy a körökben meg van adva.

Az alábbiakban további pályák vázlatát láthatod. Csak egy fejezhető be a megadott szabályok szerint. Melyik az?

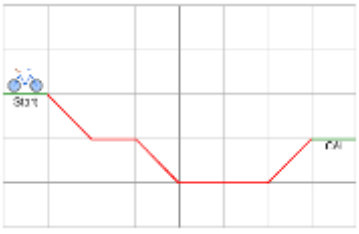
A)



B)



C)



D)

A HELYES VÁLASZ A C)

Egyedül csak a „C" pályán lehet végigmenni.

A „C" pálya teljesítéséhez a vízszintes szakaszokon az alábbi sebességváltoztatásokat kell végrehajtani: + – –, azaz gyorsítás, lassítás majd újra lassítás. Célba éréskor így lesz 0 km/h a kerékpár sebessége. Akkor is 0 km/h, ha az alábbi változások történnek: – + – vagy – – +.

A többi pálya nem teljesíthető:

- „A" pálya: Ha minden lehetséges helyen nő a kerékpár sebessége, akkor is csak az emelkedő feléig jut a kerékpáros.
- „B" pálya: Célba éréskor a kerékpár sebessége nagyobb, mint 10 km/h, akkor is, ha a vízszintes szakaszokon lassít a kerékpáros.
- „D" pálya: Ahhoz, hogy a célban 0 km/h legyen a sebessége, a kerékpárosnak az utolsó szakaszon lassítania kellene, de hegyről lefelé a sebesség nő.

MIÉRT INFORMATIKA?

A zárójelek fontos szerepet játszanak a műveletek világában. A matematikai kifejezésekben, mint például [n(n-1)] /2 vagy (a+b)(a-b) meghatározzák a zárójelek, hogy milyen sorrendben kell kiértékelni az egyes részkifejezéseket. A zárójelek mindig párosával fordulnak elő: egy nyitó és egy záró zárójel. Egy zárójeles kifejezés akkor helyes, ha minden nyitó zárójelet pontosan egy csukó zárójelhez, minden csukót pedig egy nyitó zárójelhez tudunk rendelni.

Nem csak a matematikában, hanem az informatikában is vannak zárójeles műveletek, melyeket a számítógépeknek fel kell dolgozniuk. Jó példa erre a HTML nyelv, amely a weboldalak leíró nyelve. HTML kódban egy bekezdést és tag-ek határolnak. Minden HTML tag-ben van egy < és >. A zárójelpárok nagyon népszerűek az informatikában, mert a számítógép könnyen fel tudja dolgozni őket a verem (egyszerű és hatékony adatszerkezet) segítségével.

Az egyes pályaszakaszok olyanok, mint a zárójelek. Egy „hegyről le" szakasz megfeleltethető a nyitó, a „hegyre fel" pedig a csukó zárójelnek. A vízszintes szakasz a joker terület, ami lehet nyitó vagy csukó zárójel. Egy pálya akkor teljesíthető, ha a pályaszakaszok zárójelre cserélése után helyes zárójeles kifejezés jön létre. A "C" pálya az alábbi módon szemléltethető: (?(??). Ez a pálya megtehető, mert a ? helyére tudunk olyan zárójelet írni, hogy helyes zárójeles kifejezés szülessen.

Ezek a következőek lehetnek: ((())) vagy ()(()) vagy ()()(). Az elméleti informatikában azokat a nyelveket, amelyek csak helyes zárójeles kifejezésekből állnak, „Dyck-nyelvnek" nevezzük.

TITKOSÍRÁS MEGFEJTÉSE (2017-RU-05)

SENIOR - NEHÉZ

Egy különleges szövegkódoló rendszer minden betűt a 0 és 9 közötti számjegyekből készített kóddá alakít. A sajátossága az, hogy semelyik kód nem kezdődhet egy másik betű kódjával.

Egy példa: Az X betűt 12-ként kódolja. Ekkor az Y-t kódolhatja 2-ként. Így sem a 12 nem kezdődik 2-vel, sem a 2 12-vel. Most a Z-t 11-ként kódolhatja, mivel sem 12 sem 2 nem kezdődik 11-gyel és a 11 sem kezdődik 2-vel vagy 12-vel. A 21 már nem engedélyezett a Z kódolására, hiszen 2-vel kezdődne, ami viszont már az Y-hoz tartozó kód.

A BEBRAS szót a rendszer így kódolta:

12112233321

Melyik számsor jelölheti az A betűt?

A) 22

B) 23

C) 33

D) 233

A HELYES VÁLASZ A C)

A teljes BEBRAS szó kódfelosztása csak a következő lehet: **1 – 21 – 1 – 22 – 33 – 321**

A számsor elején kezdjük a szétbontást. Amennyiben a **B**-t **12**-ként kódoljuk, az E betű szükségképpen az **1**-es kódot kapja (ekkor utána ismét **12**, mint **B** betű jön). Ez viszont ellentmond a kódkezdő szabálynak, vagyis a **B** betűt kódoló **12** az **E** betűt kódoló **1**-essel kezdődne.

Nem lehet, hogy a **B** betűt több, mint 2 számként kódoljuk (**121**, **1211**, **12112**, stb.), mert még egyszer meg kellene ismétlődnie. Ezáltal a **B** betűt kizárólag az **1**-es kóddal láthatjuk el. Az **E** betű után ismét **B** következik, ezért az **E**-t csak **2**-re vagy **21**-re kódolhatjuk (vagy **211223332**-re). Nem lehet **2**, mert akkor a kódolt szöveg **BEBB**-bel kezdődne. Nem lehet **211223332** sem, hiszen akkor az egész szó csak **BEB** lenne. Következésképp az **E** betűt a 2**1**-es számként kódolhatjuk csak. Így tudjuk, hogy **1 21 1** a **BEB** kódolása.

Még a hátralevő **2233321** kódrészt kell feldarabolnunk. Tekintsük a szó végén álló **S** betűt. Nem lehet a kódja **1**, sem **21**, mivel ezek a számsorok már foglaltak a **B** és **E** betűk kódolásánál. Következésképp ezek lehetnek a lehetséges kódok az **S**-hez: **321**, **3321**, **33321**, **233321** és **2233321**. **S**-t nem kódolhatjuk **2233321**-ként, hiszen akkor a szó mindössze **BEBS** lenne. Ugyanilyen okból nem lehet **233321** sem, mert akkor is hiányozna még az **R** vagy az **A** betű (ha csak egyegy számjeggyel kódolnánk is). Ha **S**-t **3321**-ként kódoljuk, az **RA** párost **223**-ként kell. Azonban az **R** nem lehet **2**, és az **A** sem lehet csak **3**, hiszen más kódszavak már kezdődnek ezekkel a számokkal. Ezek alapján az **S** csakis **321**-ként kódolható. Az **RA** párosra marad a középső számsor: **2233**. A már ismert okok miatt **R**-t **22**-ként, **A**-t **33**-ként kódoljuk.

MIÉRT INFORMATIKA?

A feladatban szereplő kódolási technikát előtag-kódnak (prefix kód) nevezzük. Egy előtag egy karaktersorozat, mellyel egy másik karaktersorozat kezdődik. Egy előtag-kód esetében a kód előtagja nem lehet egy másik kód. Vagyis nem kezdődhet egy kód egy másik kóddal. Az előtag kódoláskor a kódszavak különböző hosszúságúak.

Az előtag-kód előnye, hogy nincs szükség elválasztó jelekre a kódszavak között. Mindig felismerhető, hogy melyik helyen kezdődik a következő kódszó. Ha a gyakran előforduló betűket rövid kódszavakkal jelöljük, akkor egy szöveget hatékonyan kódolhatunk, sőt, ha szöveghalomról van szó, helytakarékosan tárolhatjuk azt. A Huffman-kódolás egy módszer hatékony előtagkódolás keresésére. Szélesen elterjedt és olyan adatformátumok használják, mint pl. a JPEG és az MP3.

WEBOLDALAK

DIGITÁLIS FÁK (2020-AT-01)

SENIOR - NEHÉZ

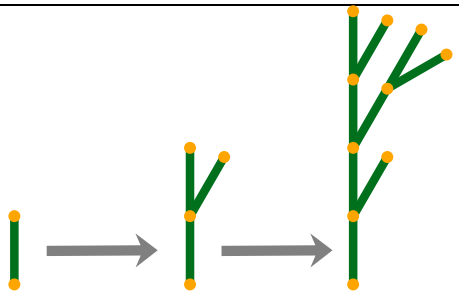
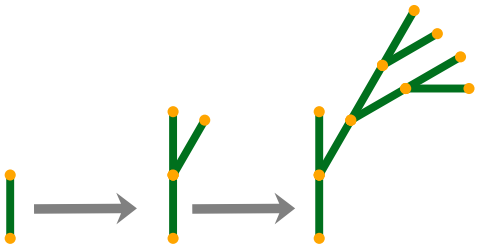
Egy digitális fa kezdetben egy fadarabból  áll:

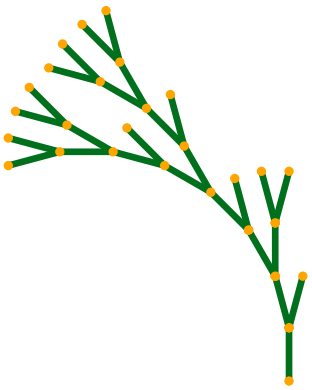
Ezután fokozatosan, megadott szabályokat követve nő.

A növekedési szabályok határozzák meg, hogyan helyettesíthető be egy fadarab egy már létrehozott összetett fa-szerkezettel. A szabály egyidejűleg minden növekedési lépésben minden fadarabra alkalmazásra kerül.

A nyílhegyek mutatják meg, hogy a fa-szerkezetek hol és melyik irányba kerülnek összekapcsolásra.

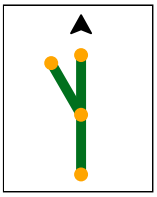
A képen látható 2 példa megmutatja az első két növekedési lépést két különböző növekedési szabály esetében.

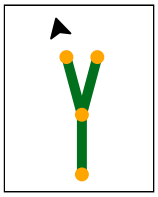
Növekedési szabály	*Első két növekedési lépés*
	
	

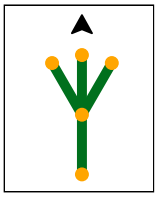


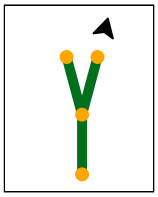
Ez egy négy lépésben nőtt digitális fa:

Milyen növekedési szabály alapján nőhetett?

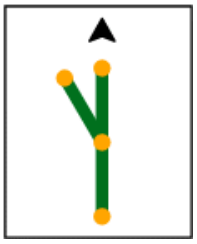
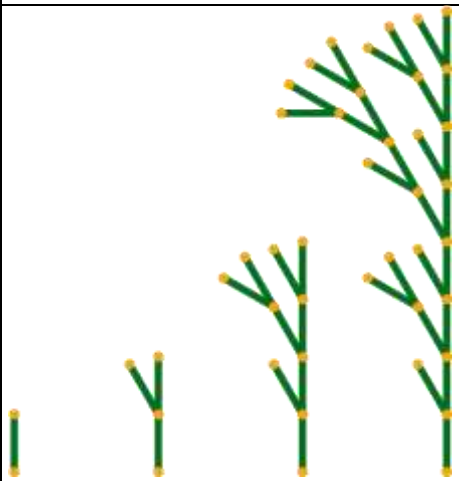
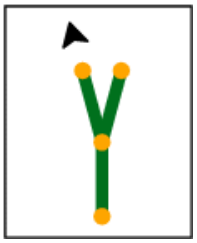
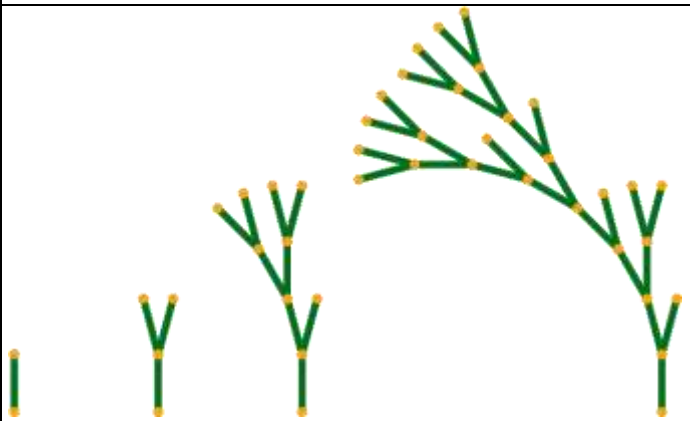
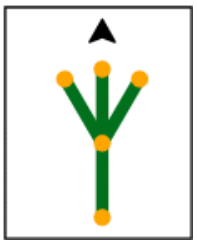
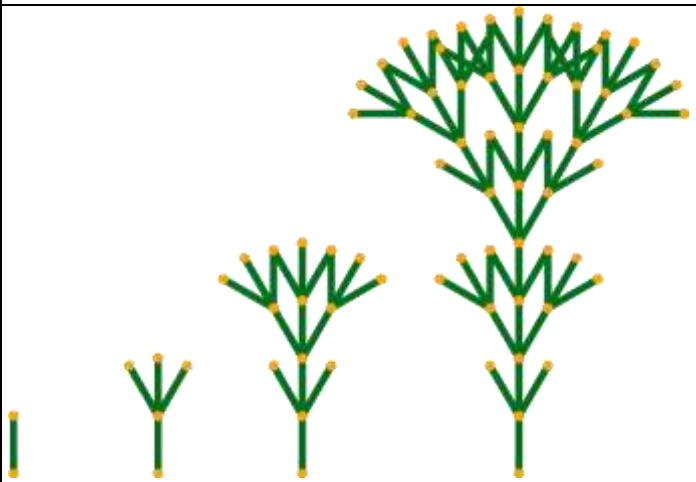
A) 

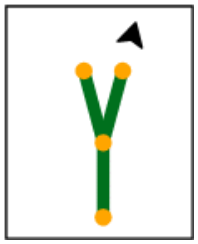
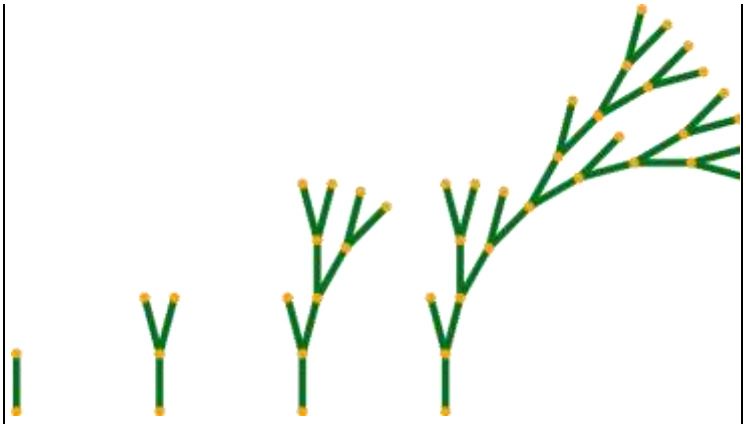
B) 

C) 

D) 

A HELYES VÁLASZ A B)

Növekedési szabály	A 4 növekedési lépés	Leírás
		A fa következő része a felfele mutatóághoz egyenesen kerül hozzáfűzésre. Emiatt egy egyenesen álló fa keletkezik, csak balra kiálló ágakkal.
		A fa többi része a baloldaliághoz, ferdén kerül hozzáfűzésre. Emiatt a fa balra nő.
		A fa többi része középen, egyenesen felfele kerül hozzáfűzésre. Emiatt a bal- és jobboldali ágak egy kiegyensúlyozott, szimmetrikus szerkezetet alkotnak.

		A fa többi része a jobbra dőlő ághoz kapcsolódik. Ezért a keletkező fa-szerkezet jobbra dől.

.

MIÉRT INFORMATIKA?

A fraktálokat többek között a számítógépes grafika területén is használják, például tájak modellezéséhez, speciális effektek megvalósításához.

De találkozhatunk velük a biológiában a növények növekedésének, a baktériumok növekedési mintázatának stb. modellezési megvalósításaiban.

Az Aristid Lindenmayer által kitalált Lindenmeyer-rendszer, más néven L-rendszer, egy formális nyelv a növényi sejtek viselkedésének leírására és a növény fejlődésének növekedési folyamatainak modellezésére. Az L-rendszerek önmagukhoz hasonló fraktálok előállítására is használhatók.

Amit megmutattunk, csak egy nagyon leegyszerűsített példa az L-rendszerre. Csak egy szabály és csak egy szimbólum van a feladatban, de a valós feladatok összetettebbek.

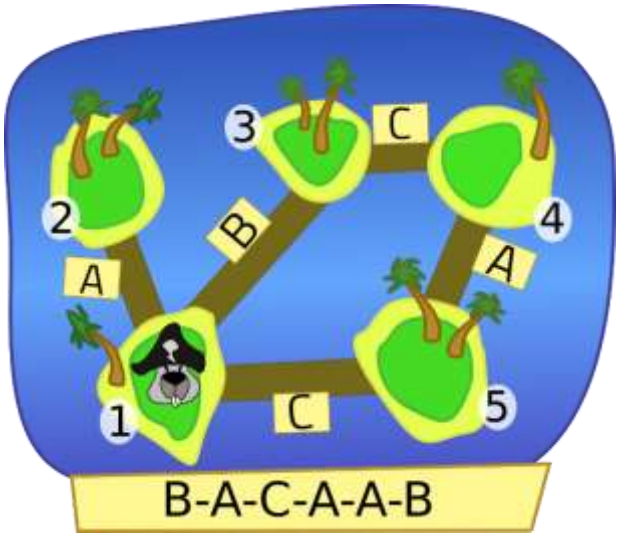
A fa felépítésének folyamata ismétlődő, az iterációt használja, mint a példában látható. Az iterációk a számítógépes programok kulcsfontosságú elemei, segítik őket a potenciálisan korlátlan adatokkal való munkában. A folyamat minden egyes komponenst egy másik fával helyettesít, amelyet legjobban rekurzív folyamatként lehet leírni.

KINCSES SZIGET (2020-AT-02)

BENJAMIN - KÖZEPES

KADÉT - KÖNNYŰ

Kalóz Kázmér a térképen 1-essel jelölt szigeten lakik. A kincseit egy másik szigeten ásta el. Az ehhez vezető utat az alábbi titkos kóddal jegyezte le: `B-A-C-A-A-B`. Ez a kód megadja neki, hogy az 1-es szigetről indulva mely hidakon kell átmennie ahhoz, hogy eljusson a kincseihez.

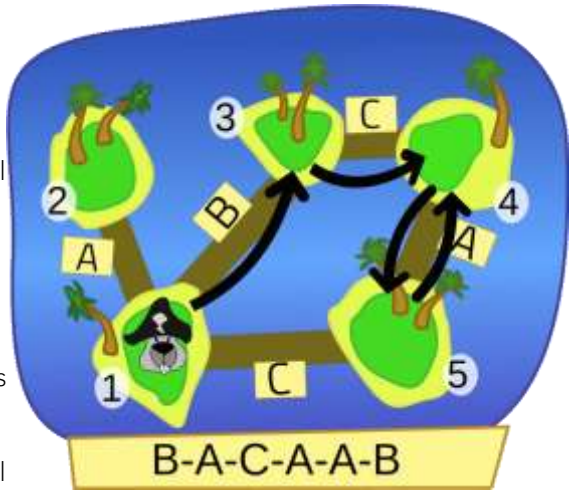


A kód időnként tartalmazza olyan hidak betűjét, melyek nem vezetnek arra a szigetre, ahol Kázmér éppen áll. Ezeket a betűket Kázmér kihagyja és a következő betűvel jelölt hídon megy tovább.

Például ha a kód `A-B-A`, Kázmér az 1-es szigetről átmegy a 2-esre az `A` hídon. Mivel nincs `B` betűvel jelölt híd a 2-es szigetről, így marad a szigeten és megnézi a következő betűt. Ez egy `A`, ezért visszamegy az 1-es szigetre.

A megadott **`B-A-C-A-A-B`** kóddal melyik szigeten rejtette el a kincseit?

A HELYES VÁLASZ: A 4-ES SZIGETEN



Az alábbi ábra fekete nyilai mutatják az utat:

Kalóz Kázmér az 1-es szigetről indul.

`B` … A „B” hídon átmegy az 1-es szigetről a 3-asra.

A … Mivel a 3-as szigethez nem csatlakozik „A” betűvel jelölt híd, így ott marad és veszi a következő betűt.

`C` … A „C” hídon átmegy a 3-as szigetről a 4-esre.

`A` … Az „A” hídon átmegy a 4-esről az 5-ös szigetre.

`A` … Ismét az „A” híd jön. Ezen visszamegy az 5-ös szigetről a 4-esre.

`B` … Mivel a 4-es szigethez nem csatlakozik „B” betűvel jelölt híd, így ott marad.

MIÉRT INFORMATIKA?

Ebben a hód-feladatban egy térképen jelölt utat kell bejárni.

A számítógépeknek is sokszor kell térképekkel dolgozniuk: például az összes GPS-eszköz térképeket tárol és ezeken kell két pont közötti útvonalat kiszámítania. A térképeket nem képekként, hanem gráfoknak nevezett szerkezetként (struktúraként) tárolják – ezek leírják a helyeket és a helyek közötti kapcsolatokat, és így könnyebben végrehajtható az útvonalak kiszámítása.

A számítógépen tárolt térkép csak a kezdet. Az informatikusoknak ezután olyan programokat kell írniuk, amelyek eligazodnak a sokféle térképen. Vannak algoritmusok a leggyorsabb, a legrövidebb útvonal és – mint ebben az egyszerűbb esetben – csak az útvonal követésére.

A feladat kódja furcsának tűnhet. Miért építsen be egy hidat, amely valójában nem áll rendelkezésre? A programok írása és futtatása során fontos, hogy legyen mód a hibás adatok kezelésére, hogy a program ne egyszerűen összeomoljon és leálljon. Meglehetősen biztosak lehetünk abban, hogy egy programunknak bármikor kellhet hibás adatokkal megküzdenie, és erre fel kell készülnünk.

HEGYMÁSZÓ (2020-CA-02)

BENJAMIN -KÖZEPES

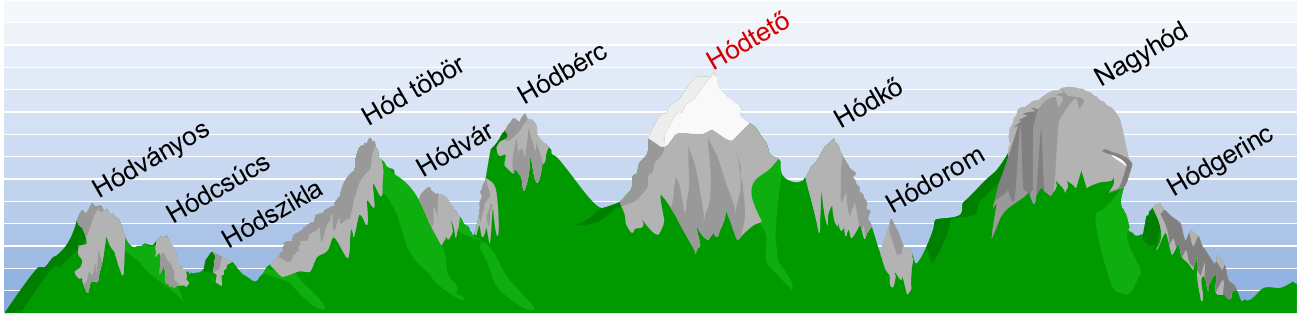
Gabi szeret a Hódhegységben kirándulni.

Amint felért egy csúcsra, megnézi a két szomszédos csúcsot.

- Ha csak egy magasabb szomszédos csúcs van, felmászik erre a csúcsra.
- Ha mindkét szomszédos csúcs magasabb, akkor felmászik a kettő közül a magasabbra.

Ezt addig ismételgeti, amíg el nem éri a csúcsot, amelynek nincs magasabb szomszédja.

A felsorolt csúcsok közül melyikre másszon fel először Gabi, hogy eljusson a legmagasabb csúcsra (Hódtetőre)?



A) Hódványos

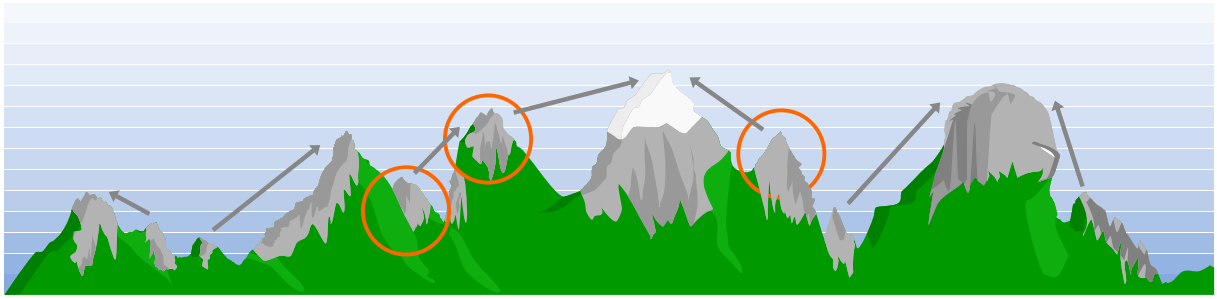
B) Hódorom

C) Hódvár

D) Hód töbör

A HELYES VÁLASZ A C): HÓDVÁR A FELSOROLT CSÚCSOK KÖZÜL A MEGFELELŐ

Az ábrán a nyilak mutatják, hogy Gabi melyik csúcsról melyikre mászna át/fel. A három körrel megjelölt csúcsról juthat fel Gabi a legmagasabbra, ha a szabályokat követi.



Azaz Hódbérc és Hódkő is megfelelő lett volna.

MIÉRT INFORMATIKA?

Gabi stratégiája a hód feladatban az, hogy mindig csak a szomszédos csúcsokat nézi meg – azaz a következő lépéséről dönt anélkül, hogy előre látná az azutáni lépéseket. Az ilyen stratégiát hívják „mohó"-nak (Mohó algoritmusnak, vagy angolul Greedy Algorithms).

Az ilyen mohó algoritmusok meglehetősen jól működnek, de nem mindig találják meg a legjobb megoldást (jelen esetben a legmagasabb csúcsot). Megmaradnak az úgynevezett helyi (lokális) maximumnál (szélsőértéknél). Akkor érdemes tehát ezeket használni, ha biztosak vagyunk benne, hogy nincs ilyen lokális maximum, vagy megelégszünk a közel optimális megoldással, nincs szükségünk az optimálisra.

Hogy elkerüljük, hogy ottragadjunk a helyi maximumon (szélsőértéken), újra is indulhatunk, vagy bonyolultabb sémákat alkalmazhatunk. Hegymászó-algoritmusnak nevezzük azt az eljárást, hogy egy kezdeti – véletlenszerű – megoldásból indulunk ki, majd iteratívan megkísérlünk egy mind jobb megoldást találni minden lépésben, mindig egy elemet megváltoztatva az eredményhalmazon, ameddig nem találunk jobbat.

A hegymászó-algoritmus gyakran jobb eredményt nyújthat, mint más algoritmusok, ha a keresés elvégzésére álló idő korlátozott, és kis számú lépés elegendő a jó megoldáshoz (az optimális megoldás vagy egy megközelítése). Sokszor használja a mesterséges intelligencia is.

FA SUDOKU (2020-CH-04B)

KISHÓD - NEHÉZ

BENJAMIN - KÖZEPES

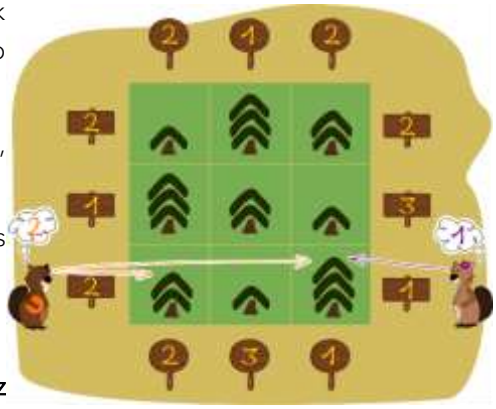
KADÉT - KÖNNYŰ

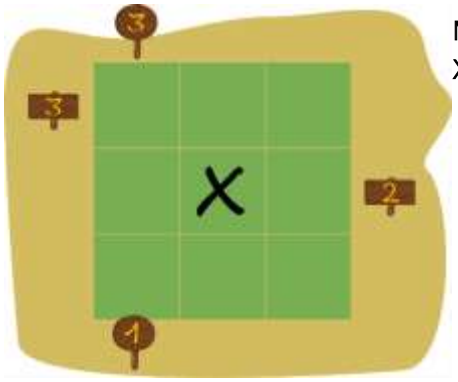
A hódok fákat ültetnek. A fák három különböző magasságúak (1  , 2  , 3 ) és minden sorba minden magasságú fából csak egyet ültetnek.

Amikor a hódok az egyes sorokon benéznek, a magasabb fák mögött álló alacsonyabbak nem látszódnak, azokat a magasabb fák eltakarják.

A sorok mindkét végén egy-egy tábla áll, amin egy szám jelzi, hogy onnan hány fa látható.

A hódok három sorban 3-3 fát ültetnek úgy, hogy a függőleges sorokra (oszlopokra) is igazak legyenek a szabályaik.



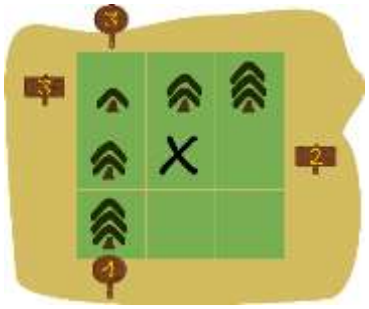


Milyen magas fa kerüljön az X-szel jelölt helyre?

A HELYES VÁLASZ A 3 MAGAS:

Kezdjük el az ültetést úgy, hogy az egyértelműen elhelyezhető fákat beültetjük.

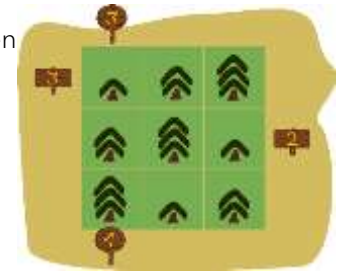
Az első sor és első oszlop előtt álló táblákon 3-as szerepel, ezzel meghatározták az adott sor és oszlop



fáinak sorrendjét (csak 1, 2, 3 lehet):

A második sor végén álló tábla 2-t mutat és az adott sorból az 1 és a 3 magas fa hiányzik. Ha 1, 3 lenne a sorrendjük, akkor a táblán 1-nek kellene lennie, mivel csak a 3 magas fa látszik arról az oldalról. A 3, 1 sorrend pont megfelelő: 2 fa látszódik a sor jobb végéről. Tehát a 3 magas fa kerül az X helyére.

Természetesen a további mezőket hasonlóan ki tudjuk tölteni, hiszen minden oszlopból pont egy fa hiányzik, ami meghatározza az alsó sorunkat.



MIÉRT INFORMATIKA?

Ez a hód feladat három gyakori informatikával kapcsolatos feladatra, célra összpontosít.

Az első egy olyan megoldás megtalálása, amely megfelel a megadott korlátozásoknak, vagy szükség szerint kijavít egy helytelen megoldást.

A második, hogy részleges információkból rekonstruálhatjuk az objektumokat az ábrázolásuk segítségével. Ez összefügg azzal, amikor korlátozott rendelkezésre álló információkból (csak a szabályszerűségének ismeretében) állítjuk elő az objektumokat. Az ilyen eljárások tömörítésre is használhatóak.

A harmadik a hibajavítás témaköre. A táblákhoz hasonlóan úgynevezett önellenőrző kódokat használhatunk az adataink tárolásakor (üzeneteink küldésekor). Az adatok bevitelénél vagy az információk továbbításakor fellépő hibák automatikusan felismerhetők vagy akár kijavíthatók az ilyen ellenőrzőkódok segítségével.

FA SUDOKU (2020-CH-04C)

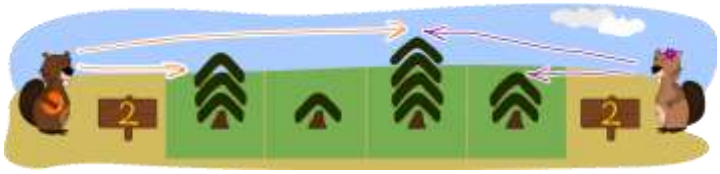
JUNIOR - KÖZEPES

SENIOR - KÖNNYŰ

A hódok16 fát ültetnek (négy 4 magasat  , négy 3 magasat  , négy 2 magasat  és négy 1 magasat) egy 4 x 4 fának alkalmas területre. Eközben az alábbi szabályokra figyelnek:

- Minden sorban (vízszintesen) minden magasságú fából pontosan egy ültethető.
- Minden oszlopban (függőlegesen) minden magasságú fából pontosan egy ültethető.

Amikor a hódok az egyes sorokon (és oszlopokon) benéznek, a magasabb fák mögött álló alacsonyabb fák nem látszanak, azokat a magasabb fák eltakarják.

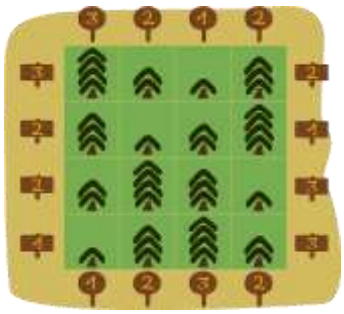


A terület szélén, minden sor és oszlop mindkét végén egy-egy tábla áll, amin egy szám jelzi, onnan hány fa látható.

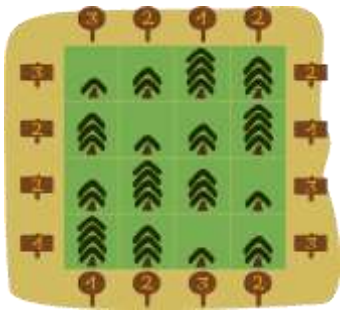
Kubo lerajzolta a területet egy darab papírra. Felírta a táblán látható számokat és berajzolta a fákat is.

Melyik lehet Kubo rajza, ha minden szabály teljesült az ültetéskor?

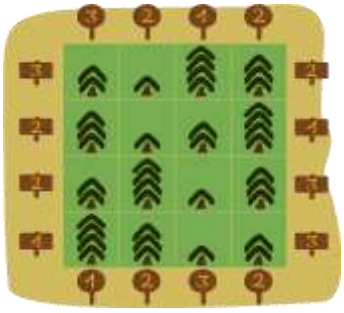
A)



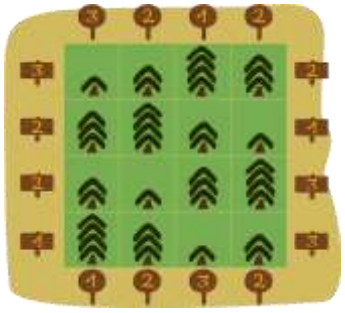
C)



B)



D)



A HELYES VÁLASZ A B)

Az „A" válasz esetében az első sor balról első fája 4 magas, így sem a sor, sem az oszlop elején álló táblán nem jó szám szerepel (1-nek kellene lennie 3 helyett). Ugyanígy nem teljesül a negyedik sor első és harmadik, valamint az első sor harmadik fája miatt az ezen sorokra vonatkozó tábla.

A „C" válasz esetében az első sor elején és a második sor végén álló tábla nem a valóságot mutatja. Ráadásul az első oszlopban két 4 magas, a harmadik oszlopban meg két 1 magas fa található, ami ellentmond az ültetési szabályoknak.

A „D" válasz esetében a harmadik sor táblái nem fedik a valóságot, hiszen balról nem 2, hanem 3 és jobbról nem 3, hanem 1 fa látható.

MIÉRT INFORMATIKA?

Ez a hód feladat három gyakori informatikával kapcsolatos feladatra, célra összpontosít.

Az első egy olyan megoldás megtalálása, amely megfelel a megadott korlátozásoknak, vagy szükség szerint kijavít egy helytelen megoldást.

A második, hogy részleges információkból rekonstruálhatjuk az objektumokat az ábrázolásuk segítségével. Ez összefügg azzal, amikor korlátozott rendelkezésre álló információkból (csak a szabályszerűségének ismeretében) állítjuk elő az objektumokat. Az ilyen eljárások tömörítésre is használhatóak.

A harmadik a hibajavítás témaköre. A táblákhoz hasonlóan úgynevezett önellenőrző kódokat használhatunk az adataink tárolásakor (üzeneteink küldésekor). Az adatok bevitelénél vagy az információk továbbításakor fellépő hibák automatikusan felismerhetők vagy akár kijavíthatók az ilyen ellenőrzőkódok segítségével.

ÉVEK A HÓDVÁRAKON (2020-CH-09B)

BENJAMIN -KÖNNYŰ

Minden hódvár bejárata fölött egy táblán szerepel az építésének az éve. A hódok minden számjegy helyett egy ábrát állítanak össze. Ehhez egy táblázatot használnak:

Például ha az 5-ös számot szeretnék leírni, akkor a  ábrát állítják össze.

Így néz ki Jópofusz várának bejárata:

Melyik évben építették Jópofusz várát?

A) 1923

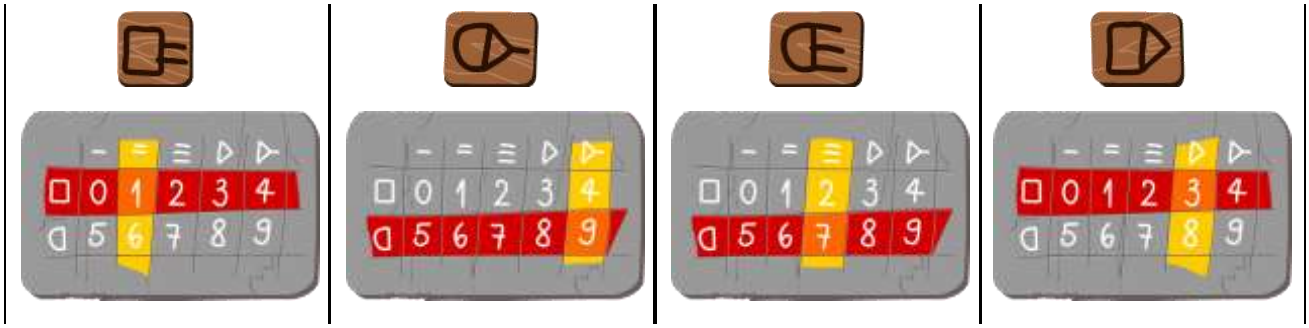
B) 1973

C) 1993

D) 1574

A HELYES VÁLASZ A B): 1973-BAN ÉPÍTETTÉK JÓPOFUSZ VÁRÁT.

A megoldáshoz minden ábrára meg kell keresnünk a táblázatban, hogy melyik sor és oszlop szimbólumából állították össze. A sor és az oszlop kereszteződéséből pedig leolvashatjuk a számjegyet.



A négy számjegy a megadott sorrendben 1973-at határozza meg.

MIÉRT INFORMATIKA?

Az információk titokban tartása vagy az adatok védelme többezer éves feladat. Számtalan titkos nyelvet fejlesztettek ki és használtak erre a célra. Ma az informatika egyik legfontosabb kérdése az adatbiztonság. Az adatok jogosulatlan hozzáférése elleni védelemnek egyik módja az, hogy titkosítjuk. Titkosításkor a sima szöveg titkosított szöveggé konvertálódik. A sima szöveg rejtjelezésből történő rekonstruálását megfejtésnek (dekódolásnak) nevezzük. Az információk titkosításának tudományát kriptográfiának hívják.

Az ókori kultúrák többnyire rejtjeleket használtak, amelyeket a betűk más szimbólumokkal történő kódolásával hoztak létre. A feladatban használt rejtjelet eredetileg a hódversenyre fejlesztettük ki, de ősi palesztinai koncepción alapul. A biztonsági szabály akkoriban az volt, hogy csak olyan titkosítási eljárásokat használtak, amelyeket könnyen meg lehetett tanulni fejből. A titkosítási algoritmus leírását túl nagy kockázatnak tartották. Az itt használt táblázat könnyen megjegyezhető. Ezen az elven alapul a szabadkőművesek híres titkosírása is.

Ahelyett, hogy csak a számok számára állítanánk össze új karaktereket, saját új titkosítót is kitalálhatunk a szövegeinkhez. Ehhez beírjuk az összes betűt egy táblázatba, és új szimbólumokat találunk ki az oszlopokhoz és sorokhoz, és így új karaktereket kapunk az összes betűhöz.

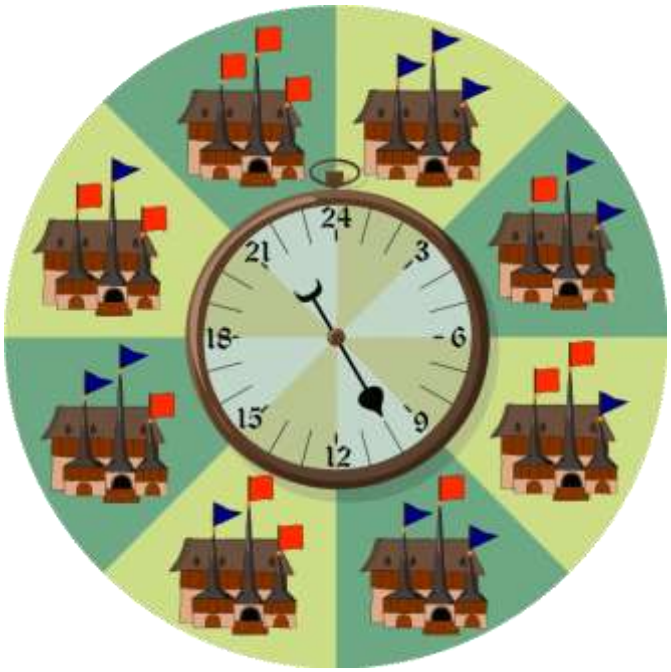
A feladat titkosítási módja az úgynevezett monoalfabetikus titkosítás. Ezzel a módszerrel minden betűhöz pontosan egy új karakter kerül kiválasztásra. Az ilyen szövegeket megfejtő kriptoanalitikusok speciális technikákat, például gyakoriság(frekvencia)elemzéseket vagy n-gram modellt használnak, hogy a karaktereket a helyesen megfejtett betűkkel összeillesszék. 1843-ban megjelent „Az aranybogár” című novellájában Edgar Allen Poe megmutatta, hogy ez minden monoalfabetikus titkosítással írt szövegnél lehetséges.

KÉNYELMES HÓDOK (2020-CH-18)

KADÉT - KÖNNYŰ

Egy kisvárosban nagyon kényelmes hódok laknak. A napokat 8 időszeletre osztják: 3 óránként egyre.

A városháza tornyain 3 zászló mutatja az időt. Kétféle zászlót használnak a hódok: piros négyzet alakút és kék háromszög alakút. (lásd a képen)



De a zászlók kezelését nem találják elég kényelmesnek. Olyan megoldást szeretnének, hogy minden váltáskor csak egy zászlót kelljen megváltoztatniuk.

Melyik két zászlóállást cseréljék fel, hogy ez teljesüljön?

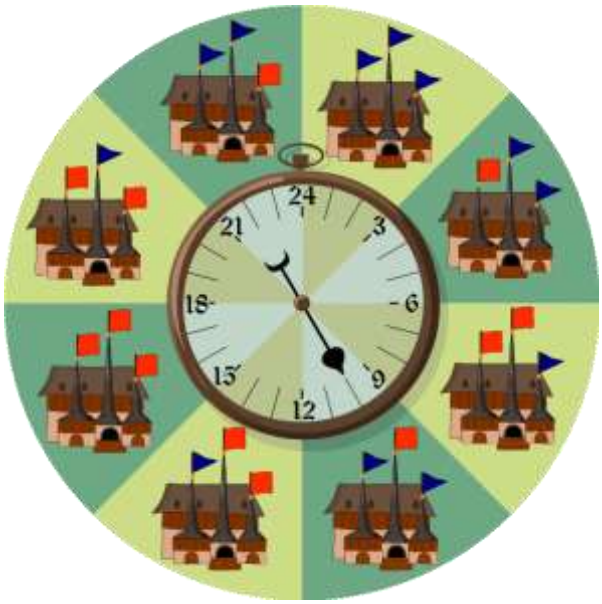
A)  -t és  -t

B)  -t és  -t

C)  -t és -  t

D)  -t és  -t

A HELYES VÁLASZ AZ A)



Természetesen más megoldások is előállíthatóak egy  cserével. Például, ha  -t cseréljük ki -vel

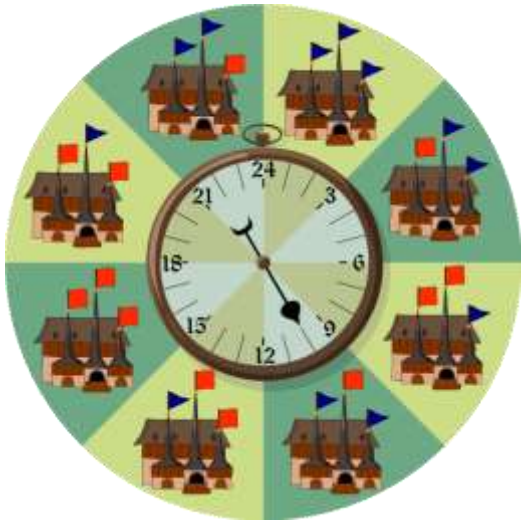
Az eredeti órán csak egy váltás esetében nem teljesült a feltételünk, amikor a csupa piros zászlókról váltunk a csupa kék zászlóra. Azaz biztosan ezek közül kel választanunk az egyik kicserélendő zászlóállást.

A zászlóállásokat leírhatjuk háromjegyű bináris számokkal: ahol az 1-es a kék, a 0-s a piros zászlót jelenti.

000	001	010	011	100	101	110	111
							

Tehát a zászlóállásaink a következő számokat jelenthetik: 000, 001, 010, 011, 100, 101 110, 111. Ezeket a számokat szeretnénk tehát úgy rendezni, hogy a szomszédos számok – valamint az első és az utolsó a sorban – csak egy számjegyben térjenek el.

Ezt megtehetjük próbálkozással is. Egy lehetséges megoldás: 111, 011, 001, 101, 100, 000, 010, 110. Melyhez itt a megfelelő óra:



De a következő módszer (stratégia) alkalmazásával biztosan találunk megoldást:

Vegyük először azokat a számokat, melyek két 0-val kezdődnek. Ezek a 000 és a 001. Két lehetséges sorrendjük lehet. Válasszuk a 000, 001-et.

Most ezt a két számot írjuk mögéjük fordított sorrendben (azaz 001, 000), de a második számjegyet változtassuk 0-ról 1-re (azaz 011, 010). Tehát megkapjuk a 000, 001, 011, 010 sorozatot, ami teljesíti a feltételt.

Most írjuk a kapott négy számot fordított sorrendben a sorozatunk végére úgy, hogy a harmadik számjegyüket 0-ról 1-re változtatjuk. Ezzel 000, 001, 011, 010, 110, 111, 101, 100 sorozatot kapunk, ami ismét teljesíti a feltételt. Megkaptuk a megoldásunkat.

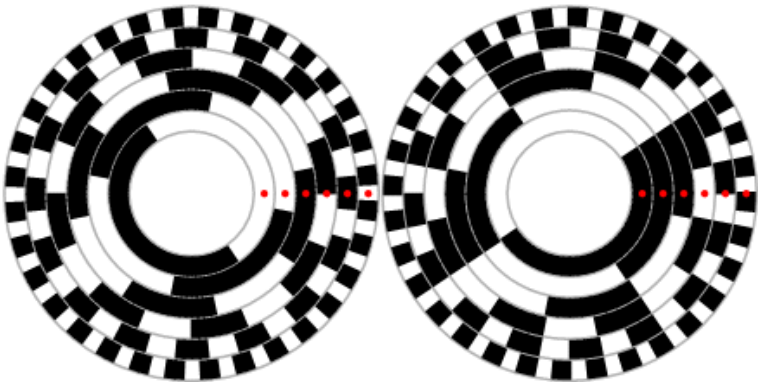
Ez a módszer – a meglévő számsor tükrözése és a következő számjegy megnövelése – a végtelenségig folytatható tetszőleges számú zászlók (számjegyek) elrendezéséhez.

Gondoljuk meg, miért is működik ez a módszer és miért tartalmaz majd minden szóba jöhető számot.

MIÉRT INFORMATIKA?

A bináris számok ilyen elrendezését Gray-kódnak hívják, és sokféle felhasználása van. Az a tény, hogy a szomszédos számok között csak egy bit változik, elősegítheti például az energiatakarékosságot. Több bit megváltoztatása mindenképpen több energiát igényel, és a bináris számok normális, növekvő felsorolásával sok bit nagyon gyakran változik egyszerre.

A Gray-kód legismertebb alkalmazása a lemezjátszó szögeinek mérése. A Gray-kódot a lemezre rajzoljuk, amint az a bal oldali képen látható, fehér jelenti a 0-t, fekete pedig 1-et. A piros pontok olyan érzékelők, amelyek egy vonalhoz kapcsolódnak, és megkülönböztethetik a fekete és a fehér színt. Az érzékelők egy bináris számot (egy kódszót) olvasnak le, amely a lemez aktuális forgási szögét kódolja.



A bal oldali képen láthatjuk, hogy a negyedik érzékelő pontosan a fekete és a fehér határán van. Tehát az érzékelő értéke 001010 vagy 001110. Mindkét lehetőség elfogadható, mert a valós szög pontosan középen van. Ha nem Gray-kódot használunk, akkor a dolgok sokkal rosszabbul néznek ki. Nézzük meg a normál bináris kódot a jobb képen. Itt a 111010 és 111001 kódszavak követik egymást. Ha az érzékelők pontosan kettő között vannak, akkor az utolsó két érzékelő nem tud a fekete és a fehér között dönteni, így az 111011 szám is leolvasható, ami sokkal távolabb van. A legrosszabb esetben az érzékelők a teljesen fehér kódszó 000000 és a teljesen fekete kódszó 111111 határán lennének. Ekkor minden érzékelő tetszés szerint választhat 0 és 1 között, ami a szögmérést teljesen használhatatlanná teszi.

MÚZEUMBEJÁRÁS (2020-CH-21)

KISHÓD - KÖZEPES

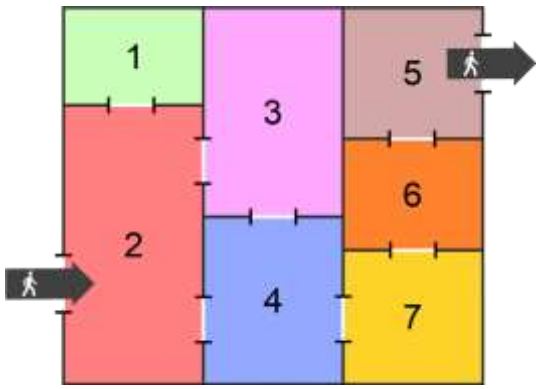
BENJAMIN - KÖNNYŰ

Egy új múzeum épül, ahol a tervezők szeretnék, ha a termek úgy helyezkednének el, hogy a látogatók minden terembe bemenjenek, de mindegyikbe csak egyszer.

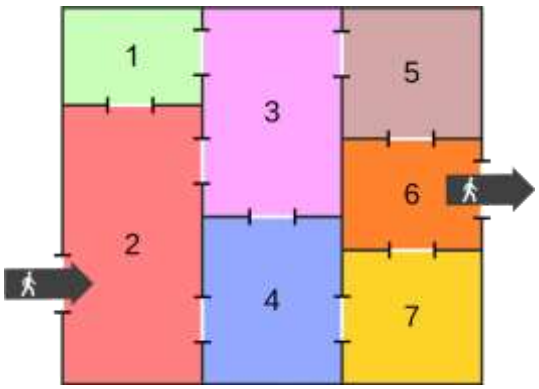
Négy tervrajz készül. Mindegyiken hét terem szerepel 1-től 7ig számozva. A nyilak mutatják, hol lépnek be a látogatók a múzeumba, és hol hagyják el azt.

Melyik alaprajz teszi lehetővé a látogatóknak, hogy minden terembe bemenjenek, de mindegyikbe csak egyszer?

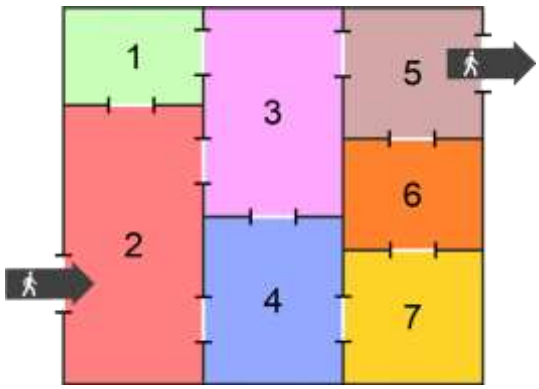
A)



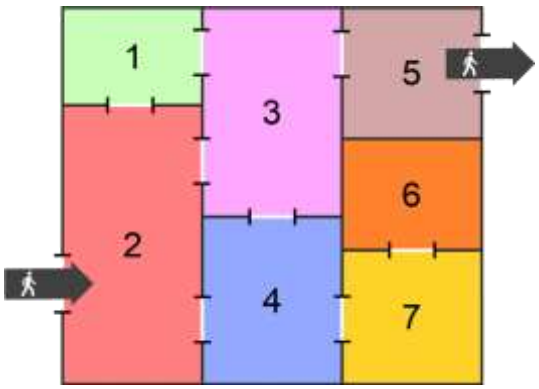
B)



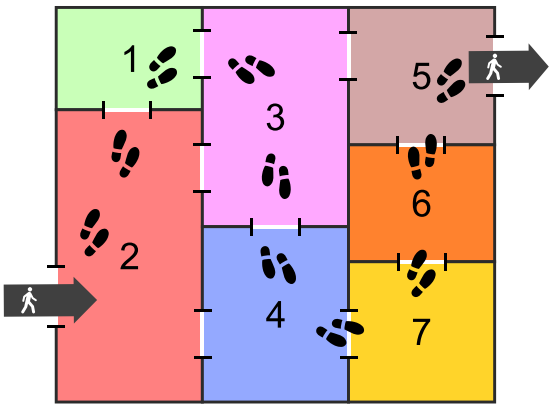
C)



D)



A HELYES VÁLASZ A C)



Csak a „C" alaprajz teszi lehetővé a látogatóknak az összes terem egyszeri meglátogatását. A termek sorrendje 2, 1, 3, 4, 7, 6, 5.

Általánosan az számít, hogyha a teremnek egy bejárata van, akkor az nem teszi lehetővé a látogatást a megadott szabályok szerint. Mivel amint egy látogató belép egy terembe, és onnan tovább menne, vissza kell mennie az előző terembe, ahol előtte már volt, ezzel pedig megszegi a szabályt, hogy minden termet csak egyszer látogathat meg.

Az „A" alaprajzon az 1-es teremnek csak egy bejárata van.

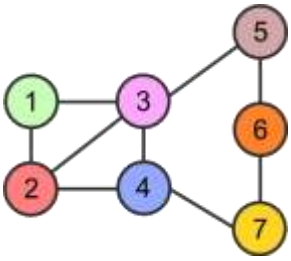
A „D" alaprajzon a 6-os teremnek csak egy bejárata van.

A „B" alaprajzon az utolsó terem a 6-os, ami az 5-ösön, illetve a 7-esen keresztül megközelíthető. Ha a látogató az 5-ös terem felől érkezik, beléphet a 7-es terembe, de csak a 6-os termen keresztül, ahhoz azonban hogy a kijáratot elérje, vissza kell mennie a 6-os terembe, vagy meg kell kerülnie, de mindenképpen érint olyan termet ahol már egyszer volt.

MIÉRT INFORMATIKA?

A legtöbb ember próbálgatással oldja meg a problémát, nem pedig valamilyen absztrakt ábrával. Ezzel bizonyos mértékben az általános visszalépéses stratégiát alkalmazzák. Felismerik, hogy a sikertelen kísérletekből tanulni tudnak, ebben az esetben vissza tudnak menni és egy újabb lehetőséget kipróbálni. Ugyanakkor szembesülnek az előre meg nem határozottság fontos fogalmával, mert számos lehetőség közül lehet választani.

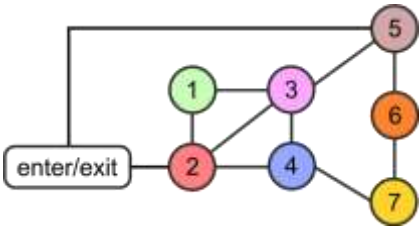
Általánosságban elmondható, hogy a probléma megfelel a Hamilton-út keresésének. Egy absztrakt ábrában a termek a gráf csomópontjainak, míg az ajtók a termek között a csomópontokat összekötő éleknek felelnek meg.



Ebben az ábrában a következő tulajdonságokkal rendelkező utat keressük:

1. Az út a 2-nél indul (bejárat/enter).
2. az útnak az 5-ösnél van vége (kijárat/exit).
3. minden élen csak egyszer mehetünk át

Ha a külső teret egy csomópontként jelöljük és a termek csomópontjaival összekapcsoljuk, akkor egy Hamiltoni kört (körutat) keresünk, ami minden élt pontosan egyszer érint.

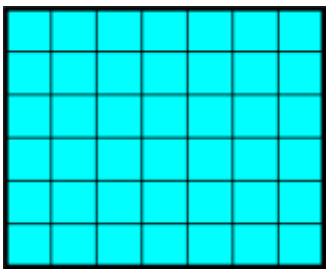


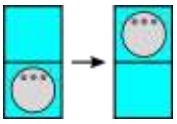
NE TÖRJ ÖSSZE! (2020-CZ-03)

JUNIOR - NEHÉZ

SENIOR - KÖZEPES

Egy robotporszívó  , egy 6x7-es négyzet alakú csempékkel kirakott, fallal körülvett szobában mozog. A robot mindig egy négyzet közepén helyezkedik el és valamelyik fal felé néz (ez a haladási iránya). A következő parancsokkal lehet mozgatni.



- `LÉPJ`: a következő csempére lép (mindig az aktuális haladási irányának megfelelően) 
- `BALRA`: 90 fokot fordul az óra járásával ellentétesen (bal felé), miközben ugyanazon a csempén  marad
- `JOBBRA`: 90 fokot fordul az óra járásával megegyezően (jobb felé), miközben ugyanazon a csempén marad 

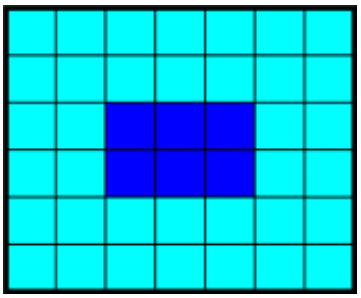
Tegyük fel, hogy a robot a következő parancsokat hajtja végre egymás után:

```
LÉPJ BAL LÉPJ JOBB LÉPJ
```

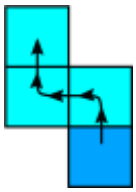
Hány különböző csempéről indulhat el a robot a parancsok egymás utáni végrehajtásához úgy, hogy a kezdeti iránytól függetlenül ne ütközzön falba?

A HELYES VÁLASZ: 6 CSEMPÉRŐL INDULHAT EL.

A jobb oldali ábra mutatja, melyek ezek a csempék (sötétkék):



A program végrehajtása alatt a robot két csempét haladt előre és egy csempét balra.



A feladat teljesítéséhez gondoskodni kell arról, hogy ez a mozgás, a kezdő csempétől mind a négy irányba megtörténhessen.

Abban az esetben, ha a robot bármelyik fal melletti csempéről vagy bármelyik fal melletti csempével szomszédos csempéről indul, (legalább) az egyik irányban (a legközelebbi fal felé) a robot falba ütközik a program végrehajtása közben.

A jelölt „központi" csempék biztonságosak, mert mindegyik faltól megfelelő távolságra helyezkednek el.

MIÉRT INFORMATIKA?

Egy program megalkotásakor a programozóknak gondolniuk kell a végrehajtás közbeni lehetséges eseményekre. Figyelembe kell venniük az összeomláshoz vezető helyzeteket, vagy a programot végrehajtó gép (számítógép, robot stb.) érvénytelen viselkedését. Vagyis meg kell határozniuk és ellenőrizniük azokat a kezdeti állapotokat, melyek esetében a program működni fog. Mondhatni, képesnek kell lenniük a program tesztelésére.

A mi robotunk hasonlóan viselkedik, mint a programozás híres „öreg" hőse, melyet az egyik legrégebbi, gyerekeknek fejlesztett programozási nyelven alkottak meg. A hőst Karelnek hívják, és több, mint 40 éve jelent meg az első számítógépeken. A neve a cseh íróra, Karel Čapekre emlékeztet, aki száz évvel ezelőtt, az 1920-ban írt R. U. R. – Rossum Univerzális Robotjai című drámájában elsőként használta a robot kifejezést azokra a személyekre, akik intelligens-mechanikus munkát végeztek.

Futurisztikus drámájában egy gyárat ír le, melyben a robotokat azért készítették, hogy az embereknek és az emberek helyett dolgozzanak, így hozva el a földi paradicsomot. Semmi sem tart azonban örökké és még a robotok is elkezdhetnek magukra gondolni.

HŐTÉRKÉP (2020-DE-02)

KADÉT - NEHÉZ

JUNIOR - KÖZEPES

SENIOR - KÖNNYŰ

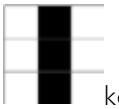
Egy gép a következő képeket, mint I, T, O, C és L betűket ismeri fel.

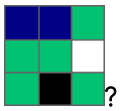
Ehhez mind az öt képhez egy „Különbségkártyát” állít elő.

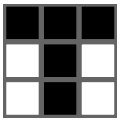
A különbségkártya az adott kép minden képpontjához egy színt rendel. A szín azt mutatja meg, hogy a többi kép esetében hánynál szerepel az adott helyen ugyanaz a képpont.

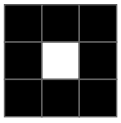
Minél világosabb egy szín a különbségkártyán, annál fontosabb az ezzel jelölt képpont a különbségek megadásánál.

Szín	Ennyi képnél szerepel ugyanazon a helyen ugyanaz a képpont
	Egy sem (0)
	1
	2
	3
	Mind (4)

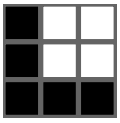
Például a  képnek ez lesz a különbségkártyája: 

Melyik kép különbségkártyája a következő:  **?**

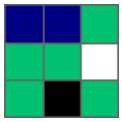
A) 

B) 

C) 

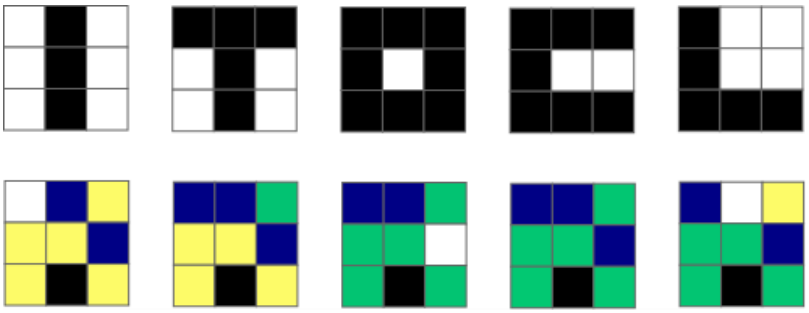
D) 

A HELYES VÁLASZ A B)

A  különbségkártya a második sor harmadik oszlopában szereplő képpontban fehér. Tehát a hozzá tartozó kép ebben a képpontjában minden más képtől különbözik.

A B válaszban szereplő kép  ezen képpontja fekete, míg az összes többinek fehér. Tehát ebben a képpontjában csak ez a kép különbözik a többitől.

Természetesen el tudjuk készíteni az összes kép különbségkártyáját:



MIÉRT INFORMATIKA?

A hőtérkép (heat map) egy grafikai megjelenítés, ahol az egyes értékeket 2 dimenzióban színekkel jelölik. Az intenzitást (erősséget) a színek árnyalataival, illetve különböző színekkel jelölik.

Biztosan találkoztatok már vele az időjárás előrejelzéseknél, illetve jelentéseknél. Itt arra is használják, hogy csak bizonyos ponton végzett mérésekkel a többi pontban is meghatározzák, megjelenítsék az értékeket.

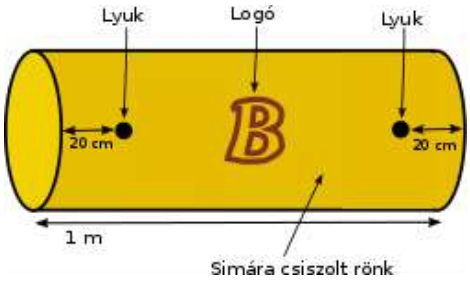
Hőtérképeket használnak az orvostudományban, az építőiparban, de a weboldalak látogatottságának elemzésére is.

FAFELDOLGOZÓ (2020-DE-04B)

JUNIOR - NEHÉZ

SENIOR - KÖZEPES

Új rönkszállítmány érkezett. Minden rönk 1 méternél hosszabb, és a képen látható hintaülések készülnek belőlük.



A következő négy robot párhuzamosan dolgozik:

Vágó: Egyméteresre vágja a rönköt.

Fúró: Pontosan 20 centiméter távolságra a rönk jobb és bal szélétől fúr egy-egy lyukat.

Nyomtató: Rányomtatja a cég logóját a megtisztított rönkdarab közepére.

Háncsoló: Eltávolítja a fakérget és simára csiszolja a rönköt.

Minden rönk csak egyszer kerülhet egy robothoz és egyszerre csak egy robot dolgozhat rajta.

Egy robot akkor kezd el dolgozni, amikor az irányítóprogram egy start-jelzést küld a számára és akkor áll le, amikor minden rönköt feldolgozott. Semelyik robot nem dolgozhat rönk nélkül (nem működhet üresen). Az irányítóprogram a parancsait egymás után küldi ki.

Készítettünk négy irányítóprogramot. **Az egyik esetében egyik robot sem dolgozik rönk nélkül és mindegyik csak a lehető legrövidebb ideig dolgozik. Melyik ez a program?**

A)

Vágó indítása
Háncsoló indítása
Várakozás amíg a dolgozó robotok befejezik a munkát
Fúró indítása
Nyomtató indítása

C)

Háncsoló indítása
Nyomtató indítása
Várakozás amíg a dolgozó robotok befejezik a munkát
Vágó indítása
Fúró indítása

B)

Vágó indítása
Nyomtató indítása
Várakozás amíg a dolgozó robotok befejezik a munkát
Háncsoló indítása
Fúró indítása

D)

Vágó indítása
Nyomtató indítása
Háncsoló indítása
Várakozás amíg a dolgozó robotok befejezik a munkát
Fúró indítása

A HELYES VÁLASZ AZ A)

A várakozás előtti és utáni robotok munkavégzése párhuzamosan történhet, mivel felcserélhető a sorrendjük. A fúró és a nyomtató robotnak méretre vágott rönkökre van szükségük és a nyomtató robotnak fontos, hogy a rönk már le is legyen tisztítva. Ezért jó megoldás, ha a háncsoló és a vágó párhuzamosan dolgozik először, majd a fúró és a nyomtató kapja meg a rönköket.

A „B" és a „D" válasz esetében a nyomtató a még kérges fára nyomtatna. A „C" válasz esetében a nyomtatónak minimum egy háncsolt fát meg kellene várnia és nem tudná hol a rönk közepe (illetve rossz helyre nyomtatna).

MIÉRT INFORMATIKA?

Ez a hód feladat a párhuzamos programozás két alapvető technikáját mutatja meg: a folyamat szinkronizálást és a mapping-et.

1) A mapping olyan alternatív fajtája az ismétlésnek, amikor egy utasítást egy adathalmaz minden elemére végrehajtunk, de a sorrend mindegy. Ha például több processzor végzi az utasítást, ez a módszer gyorsabb lehet, mint az egyszerű ismétlés (iteráció).

2) A feladat robotjai egyidejűleg dolgozhatnak a rönkökön. A vezérlőprogram gondoskodott arról, hogy a fúró és a nyomtató ne induljon el, mielőtt a vágó és a háncsoló befejezte munkáját. Ezt nevezzük folyamatszinkronizálásnak. Egy számítógépen több folyamat is dolgozhat egy időben ugyanazon az adaton.

JELSZAVAK (2020-DE-05A)

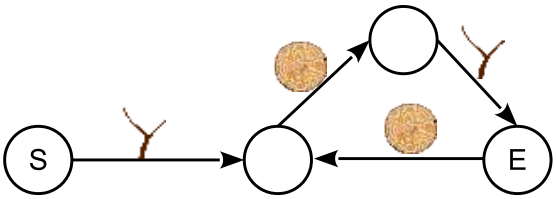
KADÉT - NEHÉZ

JUNIOR - KÖZEPES

SENIOR - KÖNNYŰ

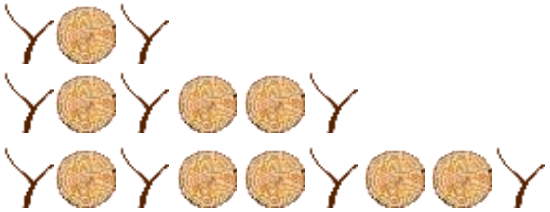
A hódok a jelszavaikat saját szabályaik alapján képzik, és ehhez a farönk  és a faág  szimbólumokat használják.

A jelszavakban mindkét szimbólumnak szerepelnie kell, és az érvényességüket a hódok egy diagrammal ábrázolják:

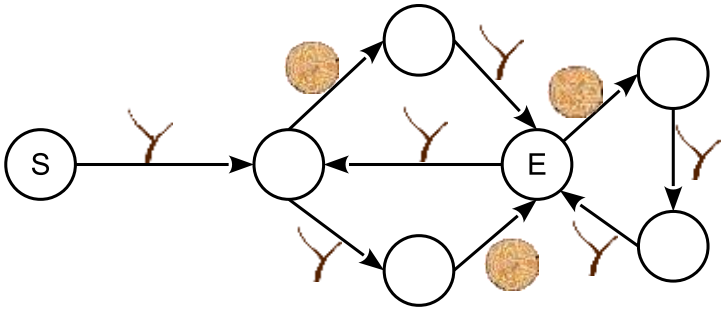


Egy jelszó akkor érvényes, ha van egy út az S körtől az E körig (akár többször érintve azt), amely a nyilakon haladva, sorrendben tartalmazza a jelszó szimbólumait.

Az S körtől az E körig minden lehetséges út érvényes jelszót ír le. Más jelszavak érvénytelenek. A jobb oldali ábra szerint végtelen számú érvényes jelszó van. Például:



A hódok egy új diagrammot készítenek:



A következő jelszavak közül melyik érvényes az új diagramm alapján?

A) 

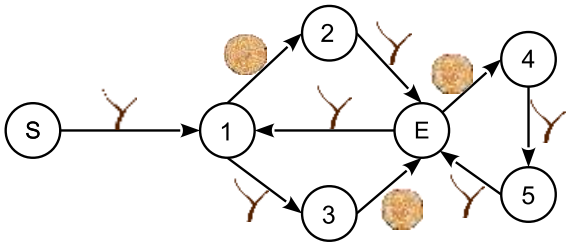
B) 

C) 

D) 

A HELYES VÁLASZ AZ A)

A helyes válasz meghatározásához találnunk kell egy utat S-ből E-be, melyen végighaladva az adott szimbólumok sorakoznak. Ehhez beszámoztuk az üres köröket:



Az „A" válasz helyes. Az út a következő: S-1-3-E-4-5-E-1-2-E.

A „B" válaszhoz biztosan nem találunk utat, mivel az S-ből csak  indul ki (azaz minden érvényes jelszó faággal  kezdődik), míg ez a jelszó  -kel kezdődik.

A „C" válasz helytelen, mivel az egyetlen út, amelyen ez a szimbólumsorozat állítható össze: S-1-2-E-4-5-E-4. De ez nem az E körben végződik.

Észrevehető az is, hogy az érvényes jelszavak hossza 3 vagy annak többszöröse. Mivel a „C" válaszban szereplő jelszó hossza 7, így biztosan nem állítható elő úgy, hogy E-ben végződjön.

A „D" válasz is helytelen: ebben az esetben találhatunk egy utat, S-1-3-E-4-5-E-4-5, mely elvezet az utolsó előtti szimbólumig. Innen azonban egy  -kel jelölt nyílon kellene továbbmennünk, de az 5-sel jelölt körből nem vezet ki ilyen nyíl.

Azt is észrevehetjük, hogy az érvényes jelszavak esetében kétszer annyi  szerepel, mint amennyi  . Ennél a megoldásnál azonban ez nem áll fent..

MIÉRT INFORMATIKA?

A hódok által használt diagramnak megvan az a tulajdonsága, hogy az egyes körökből az egyes szimbólumok legfeljebb egy kimutató nyílon szerepelnek. Ez sokkal könnyebbé teszi a jelszavak ellenőrzését, mert egyszerre csak egyetlen utat kell követni. A diagram ezen formáját az informatika állapotátmenet-diagramként is emlegeti, és egy olyan viselkedés modellje, amely pl. számítógépen futtatható. A kör egy úgynevezett állapot, a nyíl pedig átmenet egyik állapotból a másikba. Ez az ábrázolási forma különösen alkalmas a véges automaták modellezésére, amelyek véges számú állapotban lehetnek, és ezeket az állapotokat események révén megváltoztathatják. Az automaták lehetnek determinisztikusak, azaz minden eseménynél legfeljebb egy nyíl hagyja el az egyes állapotokat (köröket). A jelszavakra vonatkozó diagramunk szintén determinisztikus. Az érvényes jelszavakat elfogadó automata viselkedését írja le.

SZÁMOLÓGÉP (2020-DE-06A)

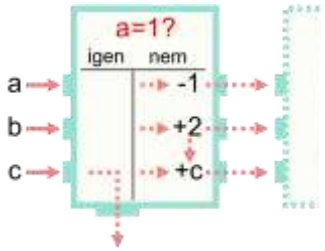
JUNIOR - NEHÉZ

SENIOR - KÖZEPES

A hódok építettek egy számológépet. Megadunk neki egy számot bemenetként és visszakapunk egy másikat.

A számológép belsejében kisebb számolóegységek dolgoznak. Mindegyik egység három számot (a, b és c) kap bemenetként és a következő lépéseket hajtja végre:

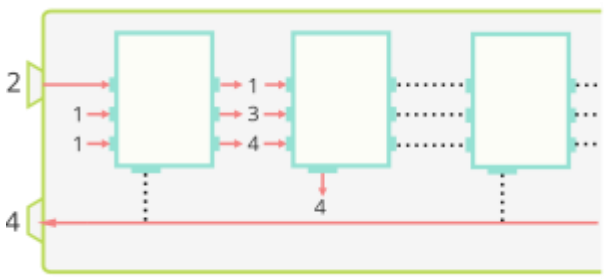
- ha „a" egyenlő 1, akkor visszaadja a „c"-t mint eredményt
- különben a következőket teszi
  - csökkenti „a"-t eggyel és az eredményt átadja a következő egység „a" bemenetére;
  - növeli „b"-t kettővel és az eredményt átadja a következő egység „b" bemenetére;
  - összeadja „c"-t a megnövelt „b"-vel, és az eredményt kiadja a következő egység „c" bemenetére.



Amikor a számológépnek megadunk egy számot, az az első számolóegység első (a) bemenetére kerül.

A számolóegység másik két bemenete (b és c) pedig 1-et kap. Amint az egyik számolóegység eredményt ad vissza, ezt a számot kiadja a számológép, mint eredményt.

A képen látható mi történik, amikor 2-t adunk meg a számológépnek. Ilyenkor két számolóegységet



használ és 4-et ad vissza.

Ha a számológépnek 4-et adunk bemenetként, milyen számot kapunk vissza?

A HELYES VÁLASZ A 16

Az első számolóegység a (4,1,1) bemeneteket kapja meg. A 4-et csökkenti eggyel, az első 1-est növeli kettővel, majd ezt hozzáadja a harmadik számhoz (ami 1). Így a kimenet (3,3,4) lesz, ami továbbítódik a második számolóegységhez. Itt (2,5,9) lesz a kimenet, ami a harmadik számolóegységbe kerül. Ez a (1,7,16) értékeket adja tovább a negyedik számolóegységnek. Ekkor viszont az első érték (a) 1, így a „D" kimenetünkre kikerül a 16 és befejeződik a számolás.

Tehát a visszakapott szám a 16 lesz.

MIÉRT INFORMATIKA?

A számológépben az egyik számolóegység továbbítja a kimenetét egy másik „szerkezetileg azonos" egységnek. Ez pontosan ugyanúgy dolgozza fel ezeket a továbbított számokat, mint az első egység az eredetieket. Új egységek kerülnek játékba, amíg egy olyan feltétel nem teljesül, ami miatt befejeződik a számolás. Ha ez a feltétel nem létezik, akkor továbbra is új egységekre lesz szükség, és a számológép soha nem áll le. Ezt a technikát feltételes ismétlésnek nevezzük.

BOMINÓK (2020-DE-08)

BENJAMIN - NEHÉZ

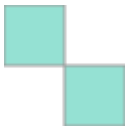
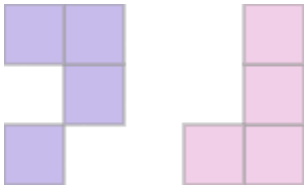
KADÉT - KÖZEPES

JUNIOR - KÖNNYŰ

A bominók színes négyzetekből álló, rácsba rendezett ábrák. A négyzetekre az alábbi szabályok vonatkoznak:

1. Minden négyzetnek érintenie kell legalább egy másik négyzetet (teljes oldalával vagy csúcsával fent, lent oldalt vagy átlósan)
2. Egy bominóban legalább két négyzetre igaz, hogy
    a. átlósan érintik egymást, és
    b. a bominó semelyik másik négyzete nem érinti egyszerre mindkét négyzetet.

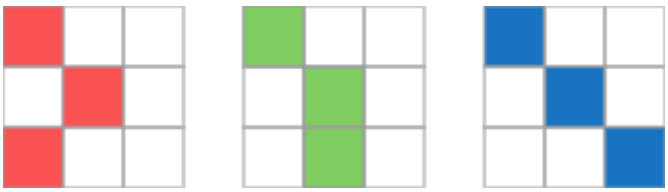
Egy n négyzetből álló bominót n-bominó-nak hívunk. Példák:

	Az egyetlen lehetséges 2 négyzetből álló bominó (2-bominó)
	A bal oldali ábra egy 4-bominó A jobb oldali ábra nem bominó, mivel a második szabály b pontja nem teljesül rá.

Hány lehetséges 3-bominó létezik? (a szimmetrikus bominókat egynek számítjuk)

A HELYES VÁLASZ A 3

A három létező 3-bominó:



Hogyan állíthatjuk elő ezeket? Induljunk ki a 2-bominóból, hiszen ezt mindenképpen tartalmaznia kell a 3-bominónak. Azaz van két négyzetünk, melyek a csúcsukkal érintkeznek, de nincs olyan négyzet, ami egyszerre érintené mindkét négyzetet. Nézzük meg, hova helyezhetjük el a harmadik négyzetet úgy, hogy csak az egyiket érintse. A szimmetria miatt elég, ha az egyiket vizsgáljuk: legyen ez az alsó. Ehhez három négyzetet illeszthetünk egy képzeletbeli rács harmadik sorában: balra átlósan, alá és jobbra átlósan.

A 2-bominót elforgatva, vagy a harmadik négyzetet máshova helyezve vagy nem bominót kapunk (2b pont) vagy a már előállított 3-bominók tükrözésével vagy forgatásával előállítható 3-bominót kapunk.

MIÉRT INFORMATIKA?

A hód feladatunkban megadott szabályok definiálják a bominókat, de nem adnak kifejezett utasítást, leírást egy érvényes bominó felépítésére vagy megrajzolására. Ez a fajta leírás (amelyet gyakran deklaratívnak is neveznek) nagyon gyakori és hasznos az objektumok általános tulajdonságainak megalapozásához, de nem megfelelő, ha a cél az, hogy megadjuk az egyikük (vagy mindegyikük) elkészítéséhez szükséges utasításokat. Arra viszont alkalmas, hogy egy adott objektumról eldöntsük, rendelkezik-e a meghatározott tulajdonságokkal, és így beletartozik-e a definiált objektumaink közé.

Az informatika számos területén az objektumok halmazának meghatározására kidolgozott szabályokat nyelvtannak nevezzük. Ez meghatározza, hogy a halmaz tárgyai hogyan épülnek fel, mint ahogy a természetes nyelv nyelvtana meghatározza a mondatok felépítését. A nyelvtanok felhasználhatók akár halmaz objektumainak létrehozására, akár annak eldöntésére, hogy egy adott objektum a halmazhoz tartozik-e vagy sem. Ebben a Bebras feladatban az összes 3-bominót generáltuk a feladatban megadott „3-bomino nyelvtan" alapján.

NUM2 (2020-HU-01)

SENIOR - NEHÉZ

Virág és Ati társasjátékot játszanak: 7 fehér és 3 fekete kövük van. A játékosok felváltva kerülnek sorra és elvehetnek 1 vagy 2 vagy 3 fehér követ vagy elvehetnek 1 vagy 2 fekete követ az asztalról. Az a játékos nyer, aki az utolsó követ veszi el.

Virág kezd. Hány darab és milyen színű követ vegyen el, hogy biztosan nyerjen függetlenül attól, hogy Ati ezután mit „lép”?

A) 1 fehér követ
B) 2 fekete követ
C) 3 fehér követ
D) mindegy hány és milyen színű követ vesz el

A HELYES VÁLASZ A C)

Gondolkodjunk visszafele: Az nyer, aki az utolsó köveket távolítja el, azaz 1, 2 vagy 3 fehér vagy 1 vagy 2 fekete kő maradt előtte, és ő jön. Ehhez kényszerítenie kell a másik játékost, hogy a fekete 3-as kupacból vegyen el egy követ (pl. fehér kő már nem maradt) vagy a fehér kupacban már csak 4 kő legyen és a fekete kupac már ne legyen az asztalon.

Tehát hogy Virág nyerjen, az utolsó előtti lépésben 3 fekete követ vagy 4 fehér követ kell az asztalon hagynia.

Mindkét esetet akkor tudja biztosan elérni, ha 3 fehér kő eltávolításával kezd. Ekkor az asztalon 3 fekete és 4 fehér kő marad. Bármelyik kupacból is vesz el követ Ati a szabályok szerint, az abban maradt utolsókat távolítja el Virág a rákövetkező lépésében. Ati előtt így vagy három fekete vagy négy fehér kőből álló kupac marad.

Ha Virág bármilyen más lépéssel kezd, Ati megfordíthatja a stratégiát: 1 fehér kő elvételével a lépésében. Azaz nem nyerne Virág Ati lépésétől függetlenül, mert Ati lépése után még két körre való kő maradna az asztalon, tehát Atié lenne az utolsó kör.

MIÉRT INFORMATIKA?

Az informatikusok előszeretettel tanulmányozzák a stratégiai játékokat, mivel sok valós interakció modellezhető hasonló módon. Az egyik leginkább vizsgált kérdés a győztes stratégia – hogy létezik-e, illetve hogyan határozható meg.

Az egyes állások (állapotok) amik meghatározhatóak egy-egy játék folyamán, felrajzolhatóak döntési fákként, gráfként, ahol a győztes pozícióból visszafele lépkedve meghatározhatjuk a játékot. A győztes stratégiák vizsgálatát komplexebb, összetettebb játékok esetében is használják, mint pl. a sakk vagy a go. De ezeknél a játékoknál sokkal bonyolultabb és nagyobb döntési fát kell felépíteni, ehhez sokszor a számítógép kapacitása sem elég. Emiatt van, hogy részlegesen felépített fákkal dolgoznak, illetve a mesterséges intelligencia tanulási mechanizmusát is integrálják egy-egy játék során.

A hód feladatunkban szereplő Nim-típusú játékok voltak az elsők, melyeket számítógéppel megoldottak és olyan stratégiákat, algoritmusokat fejlesztettek ki a vizsgálatuk során, melyekkel optimális megoldásokat találhatunk.

Az ilyen típusú megoldások a kombinatorikus játékelmélethez vannak hozzárendelve, és még gyorsabb számításokat tesznek lehetővé a számítógépekkel anélkül, hogy tárolnák az összes játékpozíció leírását, az egész döntési fát.

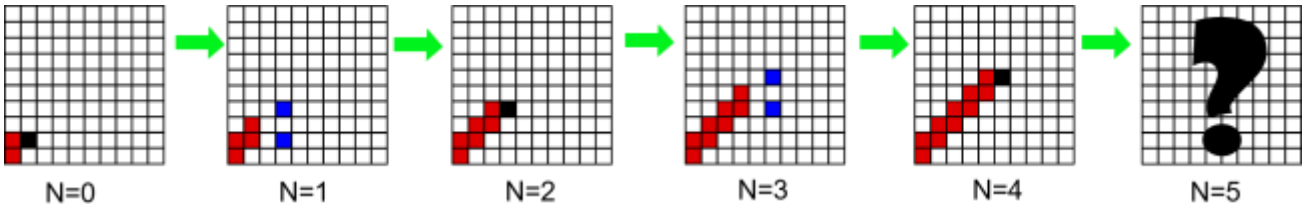
ANIMÁCIÓ (2020-HU-04)

BENJAMIN - NEHÉZ

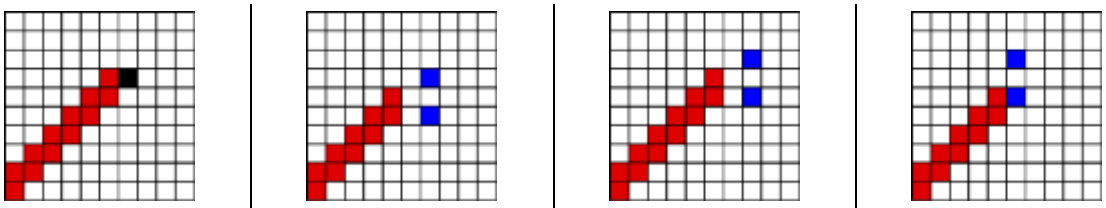
KADÉT - KÖZEPES

JUNIOR - KÖNNYŰ

Amikor a számítógép egy „ismeretlen” programot hajt végre, az N változó egyes értékei az alábbi mintákat generálják. N minden változásakor a színes minta is változik.



Milyen ábrát mutat a számítógép, ha N=5?



A

B

C

D

A HELYES VÁLASZ A C)

Ha az ábrákból egy kellően nagy (például 100) N esetén egy (képzeletbeli) videót készítünk, akkor azt tapasztalhatjuk, hogy a piros négyzetek a jobb felső sarok irányába haladnak. Az alapötlet az ismétlődés kialakítása: elsőként két piros négyzetet kapunk, majd újabb kettő egy-egy pozícióval magasabban és jobbra eltolva következik, és így tovább. Hasonlóan ismétlődik az is, hogy minden páros N szám esetén egy fekete négyzet jelenik meg a felső új piros négyzettől jobbra lévő pozícióban. Másrészt, a páratlan N számok esetén két kék négyzetet kapunk az új piros négyzeteinktől 2 pozícióra jobbra úgy, hogy az alsó kék négyzet az alsó pirossal kerül egy sorba, a felső pedig efölé két pozícióval.

Ez a gondolat képezi a feladat alapját.

Ha le szeretnénk írni ezt az algoritmust (programot), a következő utasításokat fogalmazhatnánk meg:

1. Kérjük be N értékét. N nem lehet negatív.
2. Számoljunk 0-tól N+2-ig (egyesével), ahol minden számláló lépést „i"-nek nevezünk és i-ként az éppen aktuális számot helyettesítjük be.

Minden lépésben:

a. Színezzük pirosra az (i,i) és (i,i+1) pozícióban lévő négyzeteket

b. Ha N páros, akkor színezzük az (N+2, N+2, azaz x) pozícióban lévő négyzetet FEKETÉRE (és az előző lépésben kirajzolt kék négyzetet töröljük le)

c. Ha N páratlan, akkor színezzük az (N+3,N+1 azaz y és y-2) és (N+3,N+3 azaz y) pozícióban lévő négyzeteket KÉKRE (és az lépésben kirajzolt lévő fekete négyzetet töröljük le).

MIÉRT INFORMATIKA?

Az informatikában ha készítünk egy programot, akkor azt le kell tesztelnünk. A tesztelésnek és hibakeresésnek számos módszere létezik. Ha nem ismerjük a forráskódot és azt, hogy hogyan működik a program, akkor használhatjuk a black-box (fekete doboz) tesztelés módszereit. Erre az egyik lehetőségünk az, hogy a program feltételezett működése alapján előre jelezzük a kimenetet, majd ellenőrizzük, hogy a valós bemenet feltételezésünknek megfelelően működik-e.

A feladat emlékezés John Horton Conway matematikusra, aki az életjáték atyjának tekinthető. Az életjáték alapja is egy hasonló pálya, ahol megadott szabályok alapján jelennek meg és „tűnnek el", azaz halnak meg az egyes négyzetek, azaz sejtek.

TRIÁD CSILLAG (2020-HU-07A)

BENJAMIN - NEHÉZ

KADÉT - KÖZEPES

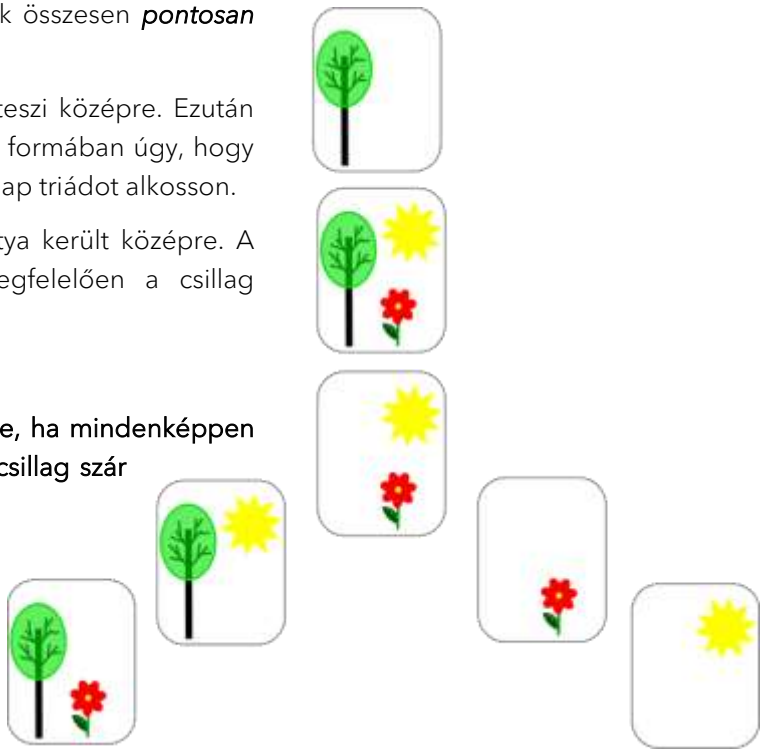
JUNIOR - KÖNNYŰ

Attila kártyáin a következő ábrák szerepelhetnek: fa, virág és nap. Minden kártya egyedi, minden ábra legfeljebb egyszer szerepelhet rajtuk. Attila nem használja az üres kártyát (amin egyik ábra sem szerepel).

Három kártya „**triád**ot" alkot, ha az egyes ábrák összesen ***pontosan kétszer vagy egyszer sem*** szerepelnek rajtuk.

A játék kezdetekor Attila húz egy kártyát és leteszi középre. Ezután megpróbálja elhelyezni a többi 6 kártyát csillag formában úgy, hogy a csillag minden szárán lévő kettő és a középső lap triádot alkosson.

A példában a virágot és napot tartalmazó kártya került középre. A többi kártya pedig a triád szabályainak megfelelően a csillag száraiba.



A hét kártya közül melyik NEM kerülhet középre, ha mindenképpen meg akarjuk oldani a feladatot (a három triád csillag szár kialakítását)?

A) A mindhárom ábrát tartalmazó kártya.

B) Azok a kártyák, melyek két ábrát tartalmaznak.

C) Azok a kártyák, melyek csak egy ábrát tartalmaznak.

D) Mindegyik kártya középre helyezésekor megoldható a feladat.

A HELYES VÁLASZ A D)

A feladat bármelyik kártya középre kerülésekor megoldható.

Kövessük a következő lépéseket (algoritmust):

1. Válasszunk egy kártyát és tegyük középre.

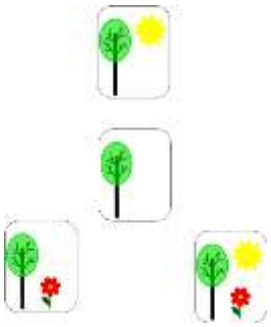
2. Válasszunk ki a középső kártyáról egy ábrát és az ezt az ábrát tartalmazó kártyákat helyezzük el, mint a csillag szárainak kezdetét.

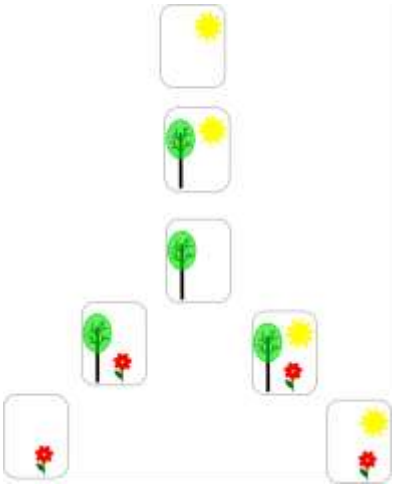
3. Helyezzük el a maradék három kártyát a triád-szabálynak megfelelően a csillag szárainak befejezéseként.

Például

1. Az első kártyánk az egy fát tartalmazó:

2. A három megmaradt fát tartalmazó kártyát elhelyezzük a csillag száraiba:





3. Minden szár folytatásába pontosan egy kártyát helyezhetünk el.

A „felső" szár esetében már szerepel két fa és egy nap. Így az egy napot tartalmazó kártyával kell kiegészítenünk.

A „jobb alsó" szárnál már szerepel két fa, egy nap és egy virág. Tehát egy napot és egy virágot tartalmazó kártyára van szükségünk.

A „bal alsó" szárnál két fa és egy virág szerepel, tehát az egy virágot tartalmazó, megmaradt kártyánk illik oda.

MIÉRT INFORMATIKA?

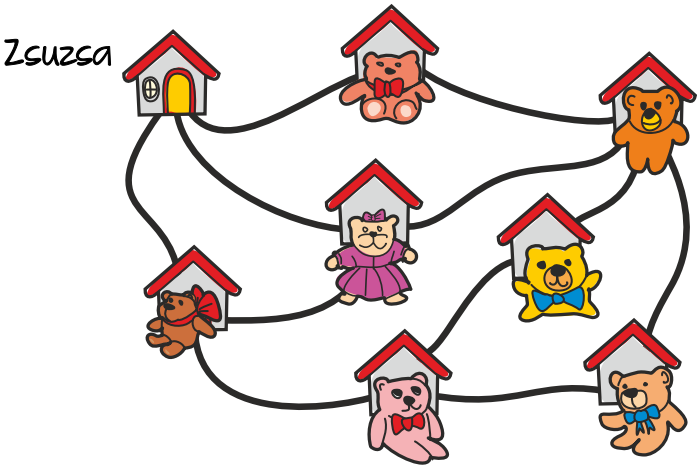
Az informatikusok gyakran objektumokkal dolgoznak, melyekről megadott tulajdonságokat tárolnak és ezek alapján csoportosítják azokat. Ebben a hód feladatban minden kártyának három tulajdonsága van (a nap, a virág és a fa ábra). Ezek a tulajdonságok két értéket vehetnek fel: szerepelnek a kártyán vagy sem.

Az informatikusok nagy mennyiségű adatoknál nem csak azonos tulajdonságokat keresnek, de például olyan objektumokat is, melyek „kilógnak a sorból", azaz minden vagy kiemelkedően sok tulajdonságukban különböznek a többitől. Ez segíthet anomáliák, betegségek vagy egy rendszer hibás működésének felderítésében.

Maga a hód feladatban leírt játék Dienes Zoltán ciklusjátékainak egyik módosítása, de logikájában a korábbi évek hód feladatai között szereplő „Quatris" játékhoz is hasonlít.

JÁTÉKMACKÓ VADÁSZAT (2020-IS-02)

KISHÓD - KÖNNYŰ



A térképen Zsuzsának és barátainak háza látható. Mindegyik barátjának a legkedvesebb játékmackójával.

Zsuzsa négy ház előtt sétált el úgy, hogy egyik előtt sem sétált el egynél többször. Egy játékmackót nem vett észre, a másik három az alábbi képen látható:

Melyik játékmackót nem vette észre Zsuzsa?

A)

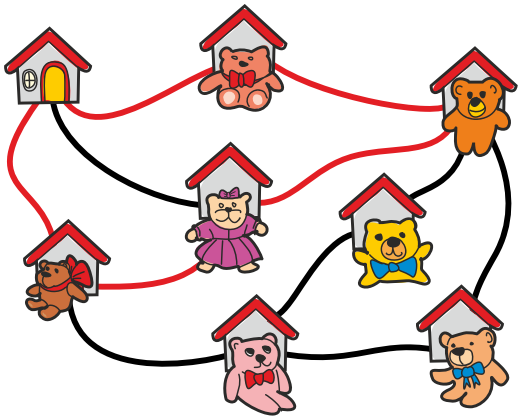
B)

C)

D)

A HELYES VÁLASZ A C):

A séta során Zsuzsa elment a  ,  és  játékmackós házak előtt. Ezeket a házakat



közvetlenül egy úttal összekötöttük (pirossal jelöltük a térképen).

Minden más úton négynél több ház előtt ment volna el Zsuzsa, vagy valamelyik ház előtt többször. Ezen az úton viszont az utolsó ház a hiányzó játékmackós:  .

MIÉRT INFORMATIKA?

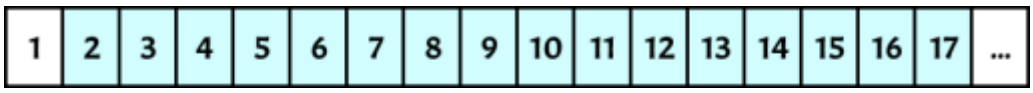
A nyelvórákon a hiányzó szövegek, matekórán az üres mezők kitöltése vagy a játékmackó vadászatban a hiányzó kép (mackó) megtalálása: mind olyan feladat, melyben egy hiányzó információt keresünk. De a feladatok mindig úgy vannak összeállítva (más szóval strukturálva, megalkotva), hogy a hiányzó elem logikus gondolkodás segítségével ki-, illetve megtalálható legyen.

Az informatikában gyakran találkozunk hasonló problémákkal: az adatátvitelnél vagy adatok mentésekor hibák történhetnek – elveszhetnek bizonyos információk. Ezért a hibák felismeréséhez, illetve kijavításához különböző eljárásokat dolgoztak ki. Az egyik ilyen, amikor több információt tárolunk, mint amennyire szükségünk van, és ha kicsi a hiba, a meglévő információkból kiszámolható, kikövetkeztethető, azaz előállítható a hiányzó elem.

A játékmackó vadászatot egyébként 2020-ban egyes országokban meg is hirdették, hogy a gyerekek távolságtartási korlátozások ellenére élvezhessék a közös bújócska és felfedezés élményét. Az ötlet Michael Rosen „We're Going on a Bear Hunt" című könyvére vezethető vissza. A kihívás lényege, hogy az emberek játékmackót raknak az ablakukba és a sétáló gyerekek (is) megcsodálhatják, megszámolhatják, „begyűjthetik" azokat, felvidulhatnak a láttukon.

MEGFORDÍTHATÓSÁG (2020-IT-02)

SENIOR - NEHÉZ



Egy robot az ábrán látható hosszú felületen elhelyezkedő tárgyakat tudja mozgatni. A felület végtelen sok sorszámozott rekeszre van felosztva, ahol a rekesz sorszáma a tárgy helyét határozza meg. A robot néhány szabály megadásával programozható a tárgyak mozgatására. Egy szabály alkalmazását követően a robot nem tudja felidézni a tárgy előző helyét.

A tárgy mindig valamelyik sorszámmal ellátott rekeszbe kerül. A robot a `MOZGAT` vagy `VISSZA` parancsokat kaphatja. A „`MOZGAT szabály 1`” parancsra végrehajtja az 1-es szabályt. A „`VISSZA szabály 1`” parancsra az 1-es szabály fordítottját hajtja végre. A `VISSZA` parancs csak a `MOZGAT` parancsot követően adható meg, és mindig a legutolsó parancsra vonatkozik.

A szabály akkor tekinthető megfordíthatónak, ha a robot a `MOZGAT` parancs után a szabály fordítottját végre tudja hajtani a `VISSZA` paranccsal anélkül, hogy összezavarodna a teendőiben. Nem minden parancs fordítható meg. *Például,*

> *szabály A*: mozgasd a tárgyat a jobb oldali rekeszbe.
>
> *szabály B: mozgasd a tárgyat az 1-es rekeszbe.*

Amikor a robot a „`MOZGASD szabály A`” parancsot kapja, a tárgyat a jobb oldali rekeszbe fogja átmozgatni. Amikor a robot a „`VISSZA szabály A`” parancsot kapja, a tárgyat balra fogja átmozgatni. *Tehát az A szabály megfordítható.*

Amikor a robot a „`MOZGAT szabály B`” parancsot kapja, a tárgyat az 1-es számú rekeszbe helyezi. Amikor a „`VISSZA szabály B`” parancsot kapja, akkor nem tudja meghatározni, hova helyezze a tárgyat , mert nem emlékszik az tárgy előző helyére és a `MOZGAT` parancs az 1-es rekeszre vonatkozott. *Vagyis a B szabály nem megfordítható.*

Nézzük az alábbi szabályokat:

1. Ha a tárgy a 8-as rekesznél nagyobb sorszámú rekeszben található, akkor mozgasd a jobb oldali rekeszbe, különben hagyd a helyén.
2. Ha a tárgy a 8-as rekesznél nagyobb sorszámú rekeszben található, akkor mozgasd a bal oldali rekeszbe, különben hagyd a helyén.
3. Ha a tárgy páros sorszámú rekeszben található, mozgasd 2 rekesszel jobbra, különben mozgasd a bal oldali rekeszbe
4. Ha a tárgy páros sorszámú rekeszben található, mozgasd 2 rekesszel jobbra, különben mozgasd a 2 rekesszel balra

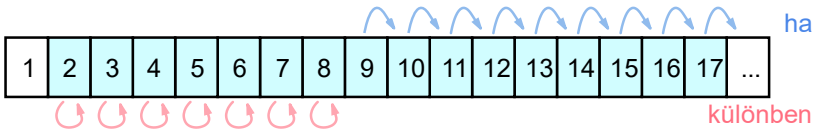
A fenti szabályok közül melyek megfordíthatóak?

A HELYES VÁLASZ: AZ 1-ES ÉS A 4-ES SZABÁLY FORDÍTHATÓ CSAK MEG

Csak az 1-es és 4-es szabályok hatása fordítható meg.

Számozzuk be a rekeszeket balról jobbra 1-gyel kezdődően és vegyük sorra az 1-es és 4-es szabályokat.

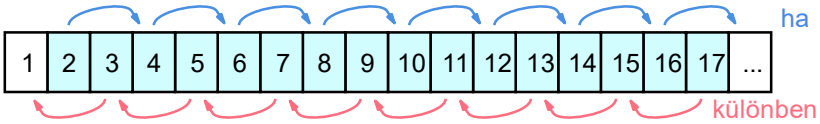
Az egyes szabály hatása a következő:



Ha a tárgy a 8-as rekesznél nagyobb sorszámú helyen áll, akkor végül a 9-es rekesznél nagyobb sorszámú rekeszbe kerül (kék nyilak); ha a tárgy helyzete kisebb, mint 9 (figyelem, a 8-as is ide tartozik), akkor a tárgyat nem mozgatjuk, vagyis a 9-es rekesznél kisebb rekeszben marad a helyén (piros nyilak). (Vegyük észre, hogy a tárgy nem kerülhet a 9-es rekeszbe.) Vagyis könnyű a két lehetséges esetet elkülöníteni és ezeknek megfelelően a tárgyat vissza(fele) mozgatni, ezért az 1-es szabály a következőképpen fordítható meg:

Ha a tárgy a 9-nél nagyobb sorszámú rekeszben található mozgasd a bal oldali rekeszbe, különben hagyd a helyén.

A 4-es szabály hatása a következő:

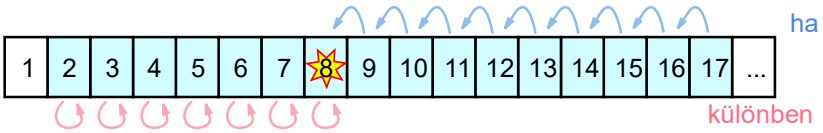


ha a tárgy páros sorszámú rekeszben található, akkor egy másik páros sorszámú rekeszbe kerül (kék nyilak), míg ha a tárgy egy páratlan sorszámú rekeszben található, szintén egy páratlan sorszámú rekeszbe kerül (piros nyilak). Vagyis a két esetet mindig el lehet különíteni, és ezeknek megfelelő a tárgyat vissza(fele) mozgatni. Ezért a 4-es szabály a következőképpen fordítható meg:

Ha a tárgy egy páros sorszámú rekeszben van, akkor mozgasd 2 rekesszel balra, különben mozgasd két rekesszel jobbra.

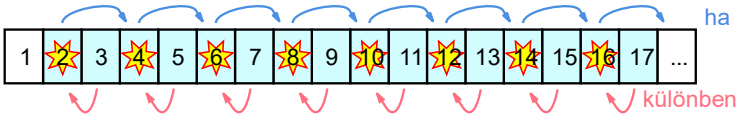
Nézzük a magyarázatát annak, hogy a többi szabály miért nem megfordítható!

A 2-es szabály hatása a következő: ha a tárgy a 8-as sorszámú rekeszben van, akkor nem mozgatjuk, vagyis a 8-as rekeszben marad (piros nyíl); ha a tárgy a 9-es sorszámú rekeszben van, akkor a nyolcas rekeszbe kerül (kék nyíl).



Tehát a szabály végrehajtását követően, ha a tárgy a 8-as rekeszben van, nem tudjuk megállapítani, hogy oda lett mozgatva vagy már eleve ott volt. Ez az, amiért a 2-es szabály nem megfordítható.

A 3-as szabály alkalmazása a következőt eredményezi: ha a tárgy egy páros sorszámú rekeszben van, egy másik páros sorszámú rekeszbe kerül (kék nyilak); ha egy páratlan sorszámú rekeszben van, akkor egy páros sorszámú rekeszbe kerül (piros nyilak).



Vagyis a szabály alkalmazását követően, ha a tárgy egy páros helyen van, nem tudunk különbséget tenni a két eset között (eleve ott volt vagy oda mozgatta a `MOZGAT` parancs), és nem tudjuk visszaállítani az eredeti helyére. Ezért a 3-as szabály sem megfordítható.

MIÉRT INFORMATIKA?

A feladatban szereplő szabályok alakja: *ha… akkor… különben…* Az informatikában ezt feltételes vezérlési szerkezetnek nevezzük. Ez a fogalom azt írja le, hogy a vezérlés csak akkor hajtja végre a parancsot, ha a feltételünk az igaz értékelést kapja. A feltételes vezérlési szerkezet a programozási nyelvek egyik alapvető eszköze, amivel az automata eszközök – mint például a feladatban szereplő robot vagy egy számítógép – viselkedését írhatjuk le.

Ha egy program feltételes vezérlési szerkezetet tartalmaz, akkor a végrehajtás két különböző irányban folyhat tovább. A kezdeti beállításoktól és bemenetektől függően a feltétel értéke lehet igaz vagy lehet hamis, és ennek megfelelően különböző parancsokat hajt végre.

Egy feltételes vezérlési szerkezet megfordítható, ha a hatása visszavonható, vagyis lényegét tekintve a kimenetből visszaállítható a kiinduló állapot. Általában a feltételes vezérlési szerkezet hatása nem fordítható vissza, mert a kimenet ismerete általában nem elegendő az eredeti állapot visszaállításához. Annak érdekében, hogy ebben a feladatban megállapíthassuk, hogy melyik feltételes vezérlés fordítható vissza, minden lehetséges kimenetet meg kell vizsgálnunk, és ellenőriznünk kell, ha „átfedések" jelennek meg, mint ahogy azt a 2-es és 3-as szabályok esetében tapasztaltuk.

A programozási feltételekkel és ezek tulajdonságaival járó hatások – például a visszafordíthatóság – vizsgálata a programozók egyik fontos képessége.

SZÍNES LAKÓNEGYED (2020-JP-02)

KISHÓD - KÖZEPES

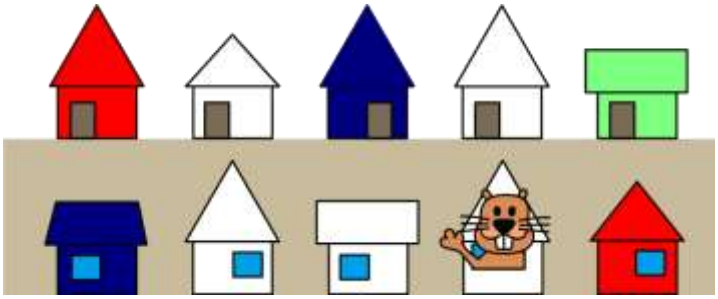
BENJAMIN - KÖNNYŰ

Egy utca lakói a házukat színesre festik: világoszöldre, pirosra vagy sötétkékre.

Pár házat már be is festettek. Hogy ne legyen egyhangú és unalmas, a lakók a következő szabályokat hozták:

- Két egymás mellett álló ház nem lehet ugyanolyan színű.
- Két ház, melyek az utca ellentétes oldalán, éppen egymással szemben állnak, nem lehet ugyanolyan színű.

Be-Taro balról a negyedik házban lakik az utca egyik oldalán.



.

Milyen színű lehet Be-Taro háza végül, ha minden fehér házat befestenek és a már befestett házakat nem akarják újra festeni?

A HELYES VÁLASZ: BE-TARO CSAK SÖTÉTKÉKRE FESTHETI A HÁZÁT.

A házak színe egyértelműen meghatározható egyszerű logikai lépésenként:

A „felső" utcarész házainak színe egyértelműen világoszöld és piros, mivel mindkettő két, már különböző színűre festett ház között található, így nekik csak a harmadik szín (lehetőség) maradt:

Az „alsó" utcarészen a balról a második ház egyértelműen csak pirosra festhető, mivel a mellette álló sötétkék, a vele szemben álló pedig világoszöld:

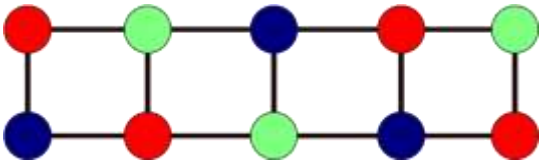
Az „alsó" utcarész középső háza pedig csak világoszöldre festhető, mivel a mellette álló piros, a vele szemben álló ház pedig sötétkék:

Végül meghatározhatjuk Be-Taro házának színét is: mivel egy piros és egy világoszöld ház között áll, ezért csak sötétkékre festhető. Ezt nem akadályozza a szemben álló piros ház sem:

MIÉRT INFORMATIKA?

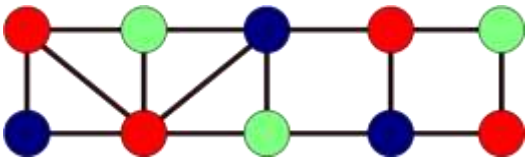
A feladatban szereplő házak és kapcsolatuk, szomszédságuk (jobb, bal és szemközti) könnyen modellezhető (leírható) gráf segítségével, mely az informatika gyakran használt adatszerkezete.

Ehhez minden házat egy csúccsal (bogyó) és a köztük lévő kapcsolatot egy éllel (összekötés) jelölünk:



Ebben az esetben elég volt három szín ahhoz, hogy kiszínezzük a gráfot úgy, hogy az összekötött csúcsoknak ne legyen ugyanaz a színe.

Ha a gráfunkban lenne még két plusz él (ld. ábra), mindez nem lenne lehetséges, a színezéshez nem lenne elég 3 szín.



Legkevesebb hány színre van szükségünk ahhoz, hogy tetszőleges számú csúcsot ki tudjunk színezni úgy, hogy ne legyenek azonos színű szomszédok (azaz összekötött csúcsok)?

A helyes válasz a négy: négy szín elég addig, amíg nincsenek metsző éleink. Az ilyen gráfokat „síkba rajzolható gráfoknak” hívjuk.

Magát a problémát „Négyszín-tétel”-nek is nevezik. Annak bizonyítása, hogy négy szín elég, meglehetősen bonyolult. Csak 1976-ban tudta Kenneth Appel és Wolfgang Haken matematikus bebizonyítani, és számítógépet is használtak az ellenpéldák és kivételek vizsgálatára. Emiatt sok matematikus nem is fogadja el ezt a bizonyítást

A Négyszín-tételt sok helyen használják a gyakorlatban. Például reptereken a repülési folyosók használatakor, vagy a mobiltelefon adótornyok frekvenciakiosztásához, hogy azok ne zavarják egymást, a vétel az adótornyok nagy számának ellenére se romoljon.

.

SÚLYMÉRÉS (2020-JP-04)

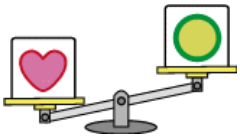
KISHÓD - NEHÉZ

BENJAMIN - KÖZEPES

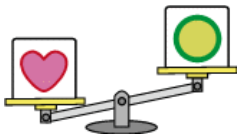
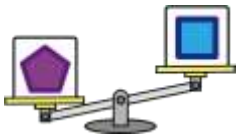
KADÉT - KÖNNYŰ

Öt dobozt öt különböző szimbólummal jelöltünk meg:  .

Egy mérleg segítségével pontosan két doboz súlyát tudjuk összemérni. A következő mérés például azt mutatja, hogy a  nehezebb, mint a  :



Összesen öt mérést végeztünk:


Melyik doboz a legnehezebb?

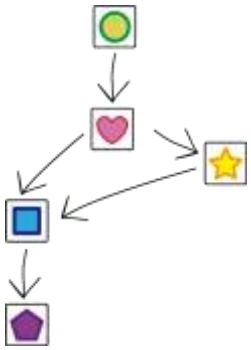
A HELYES VÁLASZ: AZ ÖTSZÖG SZIMBÓLUMMAL JELÖLT DOBOZ A LEGNEHEZEBB.

Az ötszög szimbólumos doboz nehezebb, mint a négyzettel jelölt (ld. második mérés). A négyzet szimbólumos doboz nehezebb, mint a csillaggal jelölt (ld. negyedik mérés). Azaz az ötszöges nehezebb a csillagosnál is. A csillaggal jelölt doboz nehezebb, mint a szíves (ld. ötödik mérés), ami nehezebb, mint a kör szimbólummal jelölt (ld. első mérés). Azaz mindegyik doboz könnyebb az ötszögesnél.

Ha cselesen szeretnél gondolkodni, akkor kereshetsz egy olyan dobozt, amelyik egyik mérésnél sem könnyebb egy másiknál. Ez pedig az ötszöggel jelölt.

MIÉRT INFORMATIKA?

Az a feladat, hogy állítsuk sorrendbe a dobozokat a súlyuk alapján. Az informatikában sokszor használnak gráfokat a rendezési feladatok megoldásához. Ebben az esetben a dobozok a gráf csúcsai, míg a köztük lévő viszony („nehezebb") adja az éleket, méghozzá irányítottan (a nyíl a nehezebb felé mutat):



Az ilyen rendezést topologikus rendezésnek (topologikus sorrendnek) hívjuk. A számítógép egyszerűen meg tudja oldani a feladatot: mindig eltávolít egy csúcsot, amibe nem mutat nyíl. Ha ezt következetesen, lépésről lépésre hajtja végre, megkapja a csúcsok helyes sorrendjét. De vigyázat! Az olyan gráfoknál, melyekben kör van (több csúcs éleivel körben egymásra mutat), nem megállapítható a sorrend.

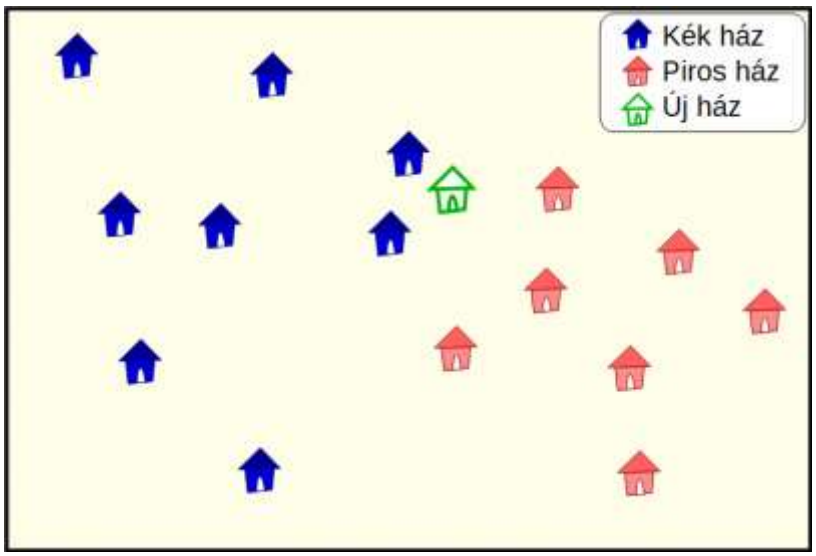
ÚJ HÁZAK (2020-KR-01)

KADÉT - NEHÉZ

JUNIOR - KÖZEPES

A falu megállapított egy **k** számot és a következő szabályokat határozta meg:

Az új ház színe legyen olyan, mint a **k** darab legközelebbi ház legtöbbjének színe. Ha nincs többség, akkor a **k+1** legközelebbi házat kell megvizsgálni.

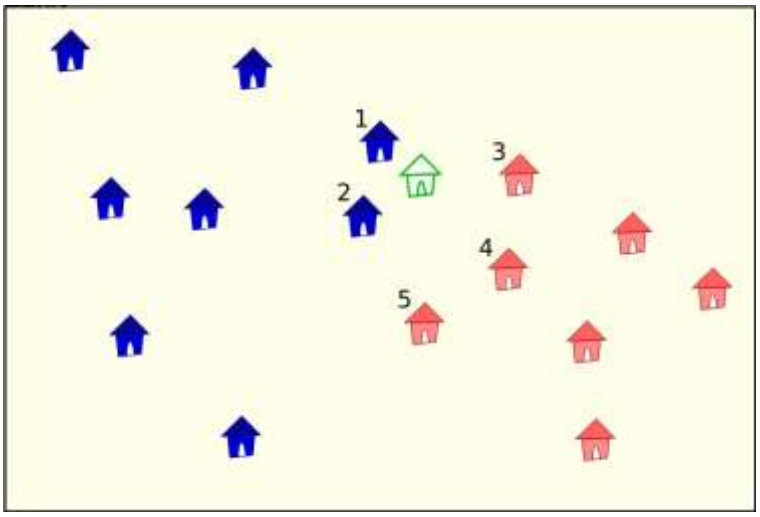


A faluban egy új házat építenek. A képen látható a falu térképe. Az új háznak még nincs színe.

Eldöntötték, hogy az új ház színe piros lesz. Mi volt a legkisebb k (szám), ami ehhez a döntéshez vezetett?

A HELYES VÁLASZ A 4

A megoldás megtalálható úgy, hogy növekvő számokat helyettesítünk **k**-val, és minden alkalommal ellenőrizzük, hogy a szabály szerint a pirosat kell-e választani.



Kezdjük **k=1**-gyel. Ez csak egy legközelebbi ház vizsgálatát jelenti. A legközelebbi ház kék. Ebben az esetben a szabály szerint a háznak kéknek kellene lennie, nem pirosnak, ahogy a feladat mutatja.

Tegyük fel, hogy **k=2**. A két legközelebbi ház kék. Ebben az esetben a szabály szerint a háznak megint csak kéknek kellene lennie, nem pirosnak.

Tegyük fel most azt, hogy **k=3**. A 3 legközelebbi ház közül 1 piros és 2 kék. Ezzel a 3 többsége a kék és az új háznak ismét kéknek kellene lennie.

Tegyük fel, hogy **k=4**. A 4 legközelebbi ház közül 2 piros és 2 kék. Ez azt jelenti, hogy nincs többségünk a 4 legközelebbi házból. A szabály szerint így a **k+1=5** legközelebbi házat kell vizsgálnunk. Ezek közül a házak közül 2 kék, 3 piros színű. Ebben az esetben tehát eldönthetjük, hogy az új ház színe piros legyen. A vizsgálatunk után ez az első, hogy az eredmény a piros. Tehát a legkisebb **k**-érték a 4.

Tegyük fel végül, még azt, hogy `k=5`. Az 5 legközelebbi ház közül 2 kék, 3 piros. Ez pontosan ugyanannak a helyzetnek felel meg, mint ahol `k=4`. Tehát `k=5` is a piros színhez vezet minket, de nem ez a legkisebb ilyen `k`.

MIÉRT INFORMATIKA?

A feladat a legközelebbi szomszéd keresés algoritmusát (Nearest neighbor search – NNS) használja fel, illetve mutatja be leegyszerűsítve.

Az NNS-algoritmust a gépi tanulásban használják az adatok osztályozására (mondván, hogy ha valami úgy totyog, mint egy kacsa, úgy úszik, mint egy kacsa és úgy hápog, mint egy kacsa, akkor az egy kacsa).

A teszteset attribútumaihoz (tulajdonságaihoz) viszonylag hasonló valamennyi tanulóesetet megkeresik. Ezek az esetek, amelyeket legközelebbi szomszédoknak (nearest neighbors) nevezünk, felhasználhatók a teszteset besorolásához.

Ezzel ismerik fel például a virágokat a fényképen vagy rendezik sorrendbe gazdasági növekedés szerint az országokat.

FA STRUKTÚRA (2020-LT-08)

KISHÓD - NEHÉZ

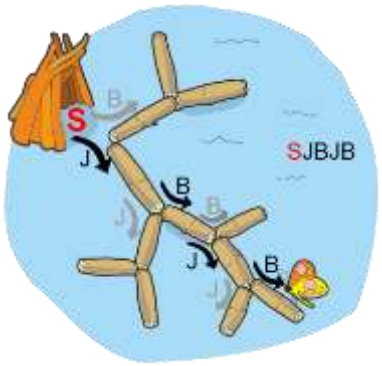
BENJAMIN - KÖZEPES

KADÉT - KÖNNYŰ

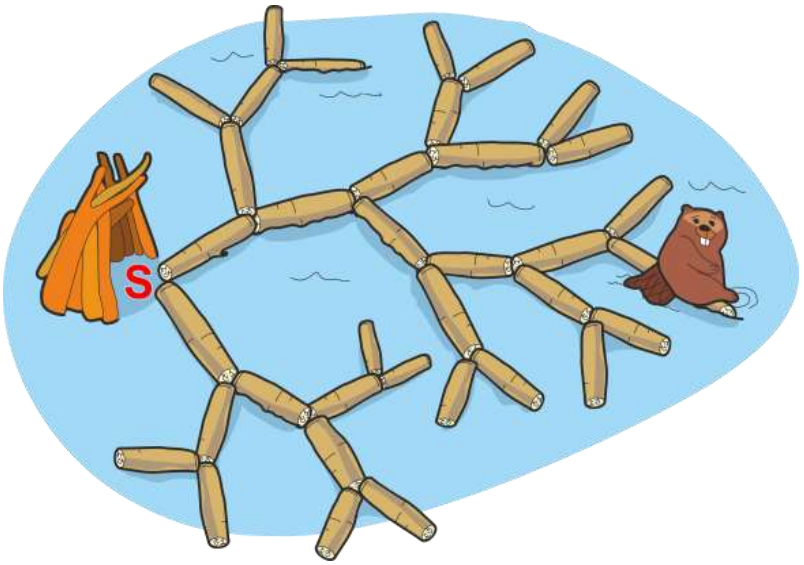
Hódék hihetetlen szerkezeteket építenek farönkökből az S-sel jelzett kunyhójukból elindulva.

Bármely rönkhöz vezető útvonal leírható a következő két paranccsal: B (balra) és J (jobbra).

Például a pillangóhoz vezető útvonal leírása: S J B J B



Vajon melyik leírás adja meg az S kunyhótól a pihenő hódig tartó útvonalat?



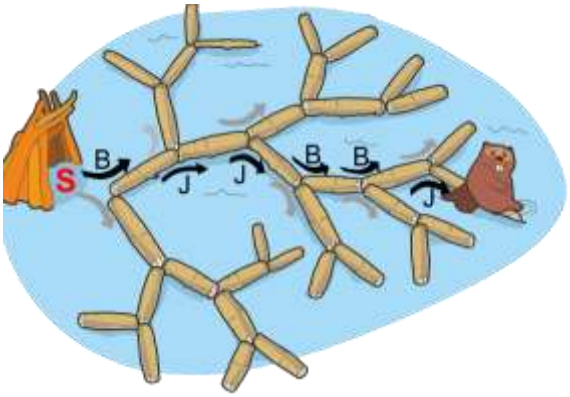
A) S B J J B B J

B) S B J B J J

C) S J B B J J B

D) S J B J J

A HELYES VÁLASZ AZ A): S B J J B B J



Az alábbi képet megfigyelve láthatjuk, hogy ha az első lépés jobbra történik, az nem vezethet a pihenő hódhoz, vagyis B-nek kell lennie. Ezt követően az első elágazásig kell haladnunk, ahol figyelembe kell vennünk, hogy melyik oldalon található a hód. És ezzel a módszerrel haladunk végig a hódhoz vezető úton, ahogy a kép is mutatja.

MIÉRT INFORMATIKA?

Az adatok több alakot is felvehetnek, attól függően, hogy hogyan reprezentáljuk (ábrázoljuk/tároljuk) őket. Az adatokat szemlélve olyan dolgok sorát látjuk, melyek hozzásegítenek egy probléma megoldásához. Ebben a feladatban a rönkök fa struktúrát (szerkezetet) alkotnak. Ez jól mutatja a gráf hierarchikus adatszerkezetét.

Azért nevezik „fa struktúrának", mert ha klasszikusan ábrázoljuk, akkor egy valódi fa képét mutatja, annak ellenére, hogy az ábra egy valódi fához képest, általában fejjel lefelé van, a növény föld alatti része kerül felülre, a „levelek" pedig alul helyezkednek el.

A fa adattípust, ami a dolgok (egyedek) közötti kapcsolatot mutatja, igen széles körben használják.

Az informatikában bináris fának nevezzük azt a fa adatszerkezetet, melynél minden csomópont legfeljebb két irányba ágazik (melyekre bal ágként illetve jobb ágként hivatkozhatunk).

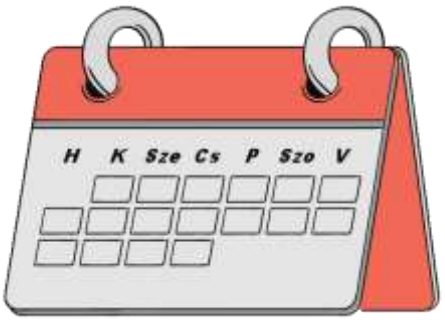
A bináris fákat gyakran az adatok gyors elérésére használjuk. Relatív kevés betűvel az ágak igen nagy számát be tudjuk azonosítani.

Tulajdonképpen 10 betű 1024 (kettő a tizediken) különböző elhelyezkedését írhatja le az ágaknak.

NAPTÁR (2020-LV-02)

KISHÓD - KÖNNYŰ

Tegnap előtt három nappal egy vasárnap előtti nap volt.



Milyen nap lesz holnap?

A) Hétfő

B) Szerda

C) Csütörtök

D) Vasárnap

A HELYES VÁLASZ A C): CSÜTÖRTÖK

A tegnap előtt három nappal ezelőtti nap négy nappal ezelőtt volt. Ez a nap pedig szombat volt (vasárnap előtt).

Ez pedig azt jelenti, hogy ma szerda van, tehát holnap csütörtök lesz.

MIÉRT INFORMATIKA?

Ez a feladat a logikus gondolkodást használatára épül.

A logikus gondolkodásnak kulcsfontosságú szerepe van az informatikában. Adatbázisok, számítási komplexitás, programozási nyelvek, mesterséges intelligencia, hardver- és szoftvertervezés és ellenőrzés területein is találkozhatunk az alapjaival.

De a mindennapjainkban is sokszor kell „logikusan" végiggondolnunk dolgokat és következtetéseket levonnunk az adott információk alapján.

KOMMUNIKÁCIÓS HÁLÓZAT (2020-MK-03)

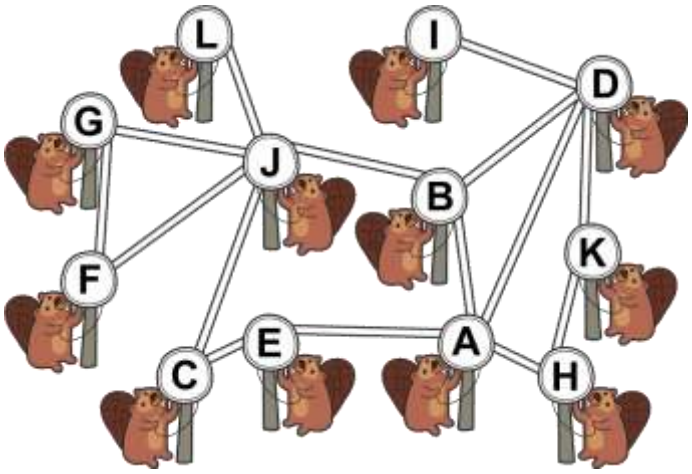
KADÉT - NEHÉZ

JUNIOR - KÖZEPES

SENIOR - KÖNNYŰ

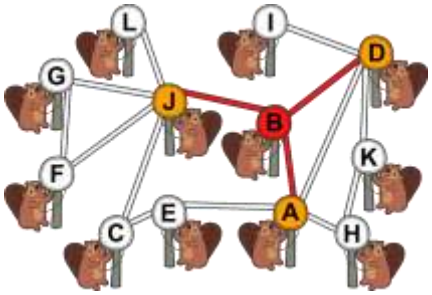
A hódok szívesen osztanak meg híreket egymással. Ha egy hód új hírt kap, azonnal és egyidejűleg megosztja az összes szomszédjával. (Szomszédai azok, akikkel közvetlen fehér vonal köti össze az ábrán.).

Az ilyen megosztások körökben zajlanak: a szomszédoknak való küldéstől a fogadásig mindig egy körről beszélünk, és tetszőleges számú hír lehet úton egy időben.



Melyik hódtól indítva ér el a leggyorsabban, azaz a legkevesebb körben egy hír az összes hódhoz?

A HELYES VÁLASZ A „B" JELŰ HÓD

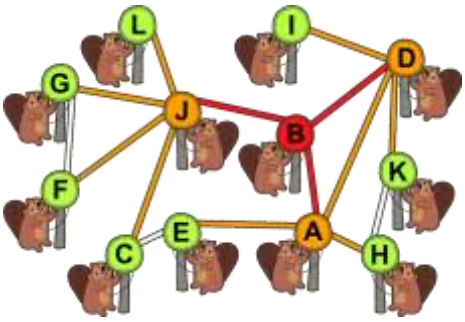


Tőle kiindulva két kör múlva már minden hód értesül a hírről. Az első körben a „B" jelű hód elküldi a hírt a szomszédainak, azaz az „A", a „D" és a „J" jelűeknek:

A második körben „A", „D" és „J" jelű hódok továbbküldik a hírt saját szomszédaiknak:

- Az „A" jelű hód elküldi „E"-nek és „H"-nak;
- A „D" jelű elküldi „I"-nek és „K"-nak;
- A „J" jelű elküldi „C"-nek, „F"-nek, „G"-nek és „L"-nek.

A „B" jelű hód háromszor is megkapja a hírt, hiszen a második körben visszaküldik neki a szomszédai („A", „D" és „J"). De mivel ez neki már nem új hír, nem küldi ismét tovább. A második körben „A" és „D" egymásnak is elküldik a hírt, de mivel számukra sem lesz már új, nem is küldik tovább.



A kép mutatja a második körben kialakult helyzetet:

A hír tehát minden hódhoz elért két kör után.

Gyorsabban nem megy, hiszen ehhez kellene lennie egy hódnak, aki mindenkivel össze van kötve közvetlenül, és így egyszerre mindenkinek el tudja küldeni a hírt.

A „B" jelű hód az, aki mindenkit elér két kör után. Az „A", „E", „F", „G", „H", „J" és „L" jelű hódok „I"-t, míg az „A", „D", „E", „H", „I" és „K" jelű hódok „L"-et nem érik el két kör után.

MIÉRT INFORMATIKA?

A hódok kommunikációs hálózata leírható egy gráfként. Minden hód egy csúcsot jelent, melyet ebben a feladatban egy betűvel jelöltünk. A fehér vonalakat éleknek nevezzük, melyek pontosan két csúcsot kötnek mindig össze. A hírek úgynevezett szinkronizált körökben terjednek, minden hód egyidejűleg tud hírt továbbítani. Ezt egy kommunikációs hálózaton belüli csomagtovábbításnak (broadcasting) is nevezzük. Ebben a hód feladatban azt vizsgáljuk, milyen gyors lehet egy ilyen csomagtovábbítás, azaz milyen gyorsan kapja meg minden hód a hírt.

Egy komoly kihívást jelentő feladat egy olyan hálózat összeállítása, ahol minden csúcs gyorsan elérhető, mégis a lehető legkevesebb az összekötések száma.

A B-vel jelölt hód csúcsát egy gráf középpontjának vagy Jordan-centrumának is nevezzük (Camille Jordan (1838–1922) miatt). Általánosan leírva a gráf középpontja egy olyan csúcs, mely estében a legtávolabbi csúcs elérése a legrövidebb (minimális).

Természetesen egy gráfon belül több ilyen csúcs is létezhet. Összetettebb gráfok esetében nehezebb megtalálni a középpontot. Az informatikusok több algoritmust is kitaláltak, melyek hatékonyan segíthetnek ebben a feladatban. Ilyen például a Floyd-Warshall algoritmus is.

EGYÜTT (2020-NZ-03)

JUNIOR - NEHÉZ

SENIOR - KÖZEPES

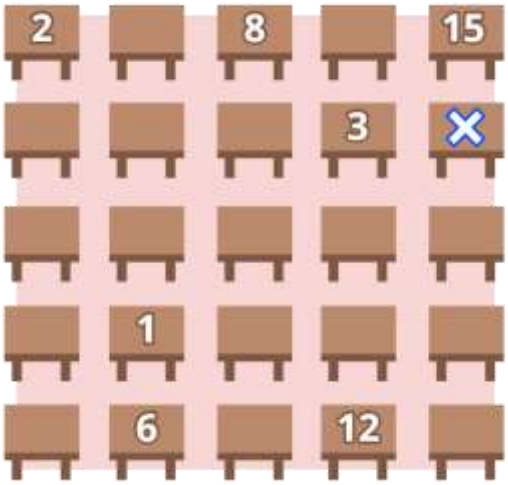
Az informatikatanár hét tanulónak adott ki szorgalmi feladatot. A tanulóknak a megoldásukat egy közös lapon kell beadniuk. A lapot az osztályban asztalról asztalra adják tovább, míg végül megérkezik a tanári asztalra  .

A lenti ábra mutatja az asztalok elrendezését. A hét tanuló a számokkal jelölt asztalnál ül. A számok azt jelentik, hány percre van szüksége a tanulónak a feladat megoldásához (melyet már azelőtt megtehet, hogy nála lenne a lap).

Egy percet vesz igénybe a lap továbbjuttatása egy szomszédos asztalra (jobbra, balra, előre vagy hátra), függetlenül attól, hogy írnak-e a lapra vagy sem.

A tanár 16 percet ad a tanulóknak.

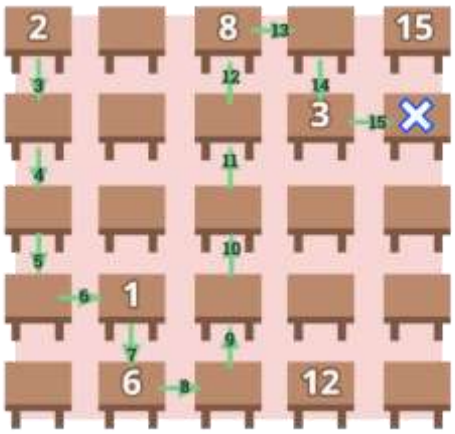
Sajnos emiatt nem lehetséges, hogy a lapon minden feladat megoldása időben elérjen a tanári asztalra.



Maximum hány feladat megoldása juthat el időben a tanári asztalra?

A HELYES VÁLASZ AZ 5

Maximum 5 feladat juthat el a tanári asztalra. Például ezzel a megoldással:



Természetesen más sorrenddel is megoldható a feladat 5 lépésben: pl. a 6 perces feladat után felírva az 1 perceset, vagy például a 15 perces feladat begyűjtésével a 3 perces helyett.

Miért nem gyűjthetjük be mind a 7 megoldást?

Könnyen belátható, ha megnézzük a 12-es és 15-ös asztalt (azokat az asztalokat, amelyeknél a megoldás előállítása a leghosszabb idő). Ha a 12-es asztaltól 12 perc elteltével választ kapunk, akkor további 5 percre van szükség ahhoz, hogy a 15-ös asztalhoz eljuttassuk a lapot. (a 15-ös asztalnál a megoldás nem gyűjthető össze korábban, mint egy perccel a megszabott idő vége előtt.) Tehát ez nem lehetséges a 16 megengedett percben.

Miért nem gyűjthetünk össze 6 megoldást?

Tehát tudjuk, hogy nem lehet mind a 12-es, mind a 15-ös asztalról begyűjteni a feladatot. De ahhoz, hogy összesen 6 megoldásunk legyen, legalább az egyiket rá kell írni a lapra.

Ha megvan a 12-es vagy a 15-ös, akkor a 16 perc alatt eljuthat a lap a tanári (X) asztalhoz, mivel a 12-es asztal 4, a 15-ös pedig 1 perc távolságra van. Tehát most az a probléma, hogy be kell bizonyítanunk, hogy eljuttathatjuk a 12-es asztalhoz, az összes többi megoldást pontosan 12 perc alatt (vagy a 15 asztalhoz pontosan 15 perc alatt).

Nézzük meg a 12-es asztalt. A következő legmagasabb megoldás 8 perc. Tehát, ha pontosan 8 perc alatt összegyűjthetjük az 1, 2, 3 és 6 perces feladatokat, akkor sajnos további 5 percet vesz igénybe, hogy eljusson a lap a 12-es asztalra. Ez pont eggyel több, mint amennyit megengedhetünk.

Nézzük meg tehát a 15-ös asztalt, és hogy összegyűjthetjük-e 1, 2, 3, 6 és 8 perces feladatokat 13 perc alatt. (További 2 perc szükséges, hogy a 8-as vagy a 3-as asztalról a lap elkerüljön a 15-re.) Ezt kissé nehezebb kieséssel bizonyítani. 120 permutáció létezik, és nem lehet mindet feltárni ebben a magyarázatban.

Az azonban belátható, hogy a 6-os asztalra legkorábban a 6. percben érdemes a lapnak eljutnia (különben még az 1 percen felül várakozni kell a megoldás felírására és a lap továbbadására). Innen viszont a 8-as asztal 5 perc „távolságra" van. Ami belefér a 13 percbe, de akkor a 3-as asztalnál még nem jártunk. Hiszen az nem fér bele, hogy a 3-6-8-as sorrendet valósítsuk meg, azaz ide-oda küldözgessük a teremben a lapot.

MIÉRT INFORMATIKA?

Ez egy optimalizálási probléma. Ebben az esetben a meglátogatott asztalok számát maximalizálni kell. A problémának van néhány további korlátja is: mennyi idő telik el a megoldás megtalálása előtt, azaz mikor „gyűjthető be" maga a feladat.

A megoldást visszavezethetjük egy gráf bejárására is. Ebben az esetben az asztalok csomópontként, az asztalok közötti kapcsolatok (szomszédos) pedig élként vannak ábrázolva. Az útvonal a gráf éleinek sorozata. Az első csomóponttól az utolsó csomópontig tartó percek számát felírjuk az útvonal minden élére. Az elérési út érvényes megoldás a problémára, ha egy él száma legalább eggyel több, mint annak a csomópontnak a száma, ahonnan kivezet. Az összes érvényes elérési út közül a megszámozott asztalok maximális számát keressük.

Az ilyen típusú problémákat különböző módszerekkel lehet megoldani. A válaszmagyarázat ellentmondásokkal mutatja meg, hogy nem léteznek érvényes utak, amelyek 6 vagy 7 számozott asztalt keresnek fel. És egyetlen példát mutat be, amely bizonyítja, hogy létezik megoldás 5 számozott asztalra. Ez formailag helyes módszer annak bizonyítására, hogy a megoldás helyes.

Amikor először szembesül egy ilyen problémával, a legtöbb ember konstruktív megközelítést alkalmaz. Megpróbálnak érvényes utat felépíteni, esetleg belefutnak egy problémába, néhány lépéssel hátrébb lépnek, más megközelítést próbálnak ki, amíg nem kapnak ötletet, hogy mi lehet jó vagy kevésbé jó megoldás.

Ezt a megoldási módszert, amit itt használtunk úgy hívják, hogy „elágazás és korlátozás" (*branch & bound*) és általában az informatika eredendően összetett problémáira alkalmazzák.

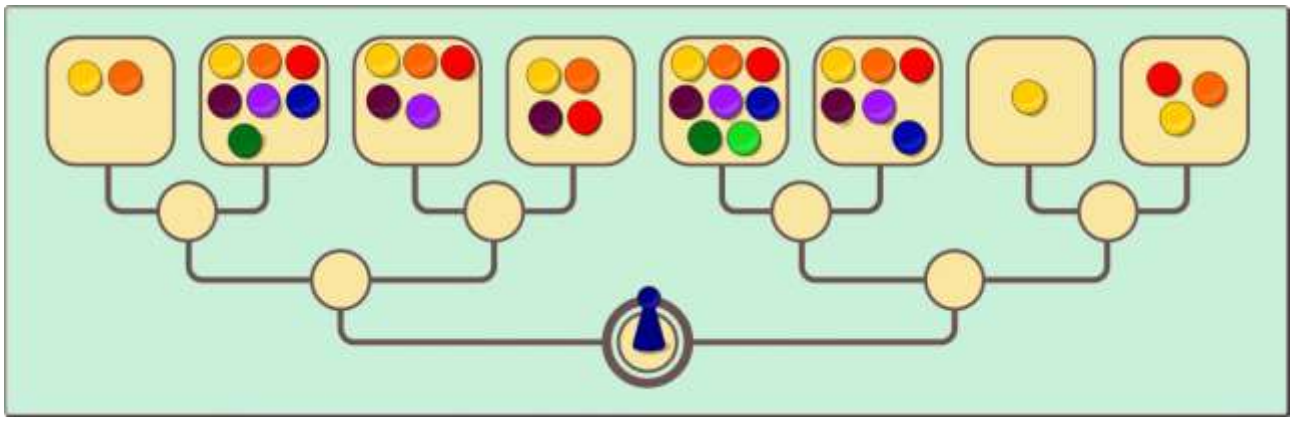
BAL-JOBB JÁTÉK (2020-PH-02)

KADÉT - NEHÉZ

JUNIOR - KÖZEPES

SENIOR - KÖNNYŰ

Anna és Bálint ebben a játékban egy bábut irányít: minden lépésben eldöntik, hogy a bábu jobbra vagy balra lépjen a különböző dobozokban lévő kövek felé. A bábus mezőről indulnak. Az első elágazásnál Anna dönt, aztán a következőnél Bálint és végül megint Anna. A végén Anna megnyeri az elért köveket, Bálint pedig elveszti ezeket.



Ezzel Annának és Bálintnak eltérő célja van. Anna a lehető legtöbb követ akarja megszerezni, míg Bálint a lehető legkevesebbet.

Anna és Bálint tudja, hogy mindketten jók ebben a játékban. Ha például Bálint a 2 és 7 követ tartalmazó doboz közti elágazásra irányítja a bábut, akkor tudhatja, hogy Anna a 7 követ fogja választani.

Anna és Bálint is megpróbálja a saját célját a legjobban elérni.

Melyik köves dobozt fogják így végül elérni?

A HELYES VÁLASZ: AZ 5 KÖVESET

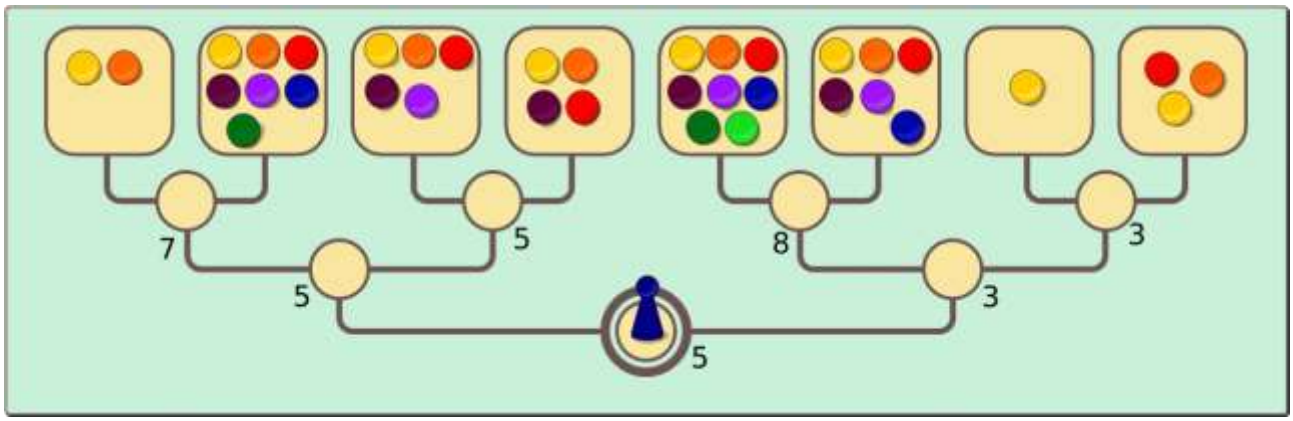
Anna tudja: Ha kezdéskor a bábut jobbra mozdítja, Bálint biztosan jobbra viszi tovább, így Anna legfeljebb 3 követ érhet el. Ha Anna a bábut kezdéskor balra viszi, akkor a végén minden esetben több mint 3 követ szerezhet.

Bálint tudja: Ha a bábut balra viszi, Anna a 7 kő felé lép tovább. Így a jobb oldali irányt választja, ahol Anna legfeljebb 5 kőhöz juthat hozzá.

Végül Anna természetesen balra megy, amivel 5 követ visz el.

MIÉRT INFORMATIKA?

Ennek a feladatnak az eredménye szisztematikusan kiszámolható: Már tudjuk, hogy Anna az eredményt maximalizálni szeretné, míg Bálint minimalizálni. Ezért például az úti cél előtti utolsó elágazásoknál, ahol Anna maximalizálja eredményét, a két elérhető doboz maximális értéke felírható. A lentebb lévő elágazásokhoz azonban Bálint mindig a kisebb változatot választja, így ott megadható a két elérhető érték minimuma. És az egész lent lévő elágazásoknál, ahol Annáé a lépés, a végeredmény maximuma számolható ki.



Ezt az eljárást MiniMa-algoritmusnak nevezzük, és a mesterséges intelligenciában a játékhoz hasonló helyzetekben használják. Különösképpen a sakk-számtógépek alkalmazzák ezt az algoritmust a legjobb lépés kiszámításához. Ám a sakk túl sok lépést kínál, melyeket a számítógép belátható időn belül kiszámol, a lépés sorozatokat néhány lépés után megszakítja, és az állást kiértékeli. Ezek a kiértékelések megfelelnek a dobozoknak a játékban. A MiniMax-algoritmussal egy aktuális állás kiértékelése után a lépések visszaszámolódnak a legjobb eredményhez vezető lépés kerül kiválasztásra.

KINCSVADÁSZAT (2020-PK-05)

KISHÓD - NEHÉZ

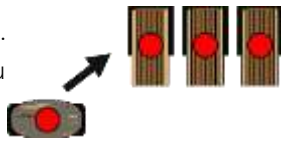
BENJAMIN - KÖZEPES

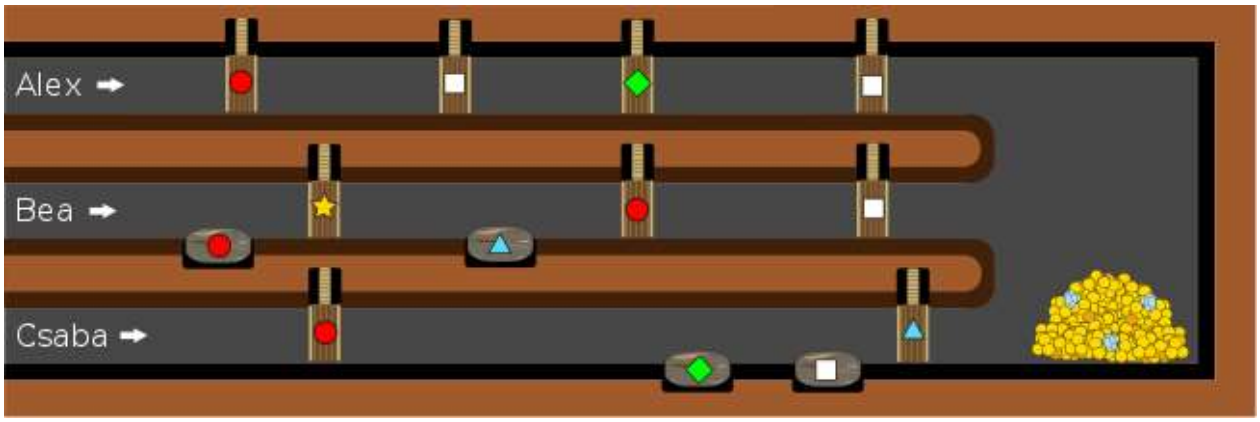
KADÉT - KÖNNYŰ

A három felfedező, Alex, Bea és Csaba meg akarja találni a kincset. Különböző járatokon indulnak el.

A járatokban kapuk  és kövek  vannak, melyek színes szimbólumokkal vannak megjelölve. Kezdetben minden kapu zárva van.

Ha egy felfedező egy zárt kapuhoz érkezik, vár, amíg kinyílik a kapu. Ha egy felfedező egy kőre lép, minden azonos színű szimbólummal jelzett kapu végleg kinyílik.





Ki fogja elérni a kincset?

A HELYES VÁLASZ ALEX

Bea először kinyitja a piros ponttal jelzett kaput. Aztán Csaba továbbmehet és minden kaput kinyit, amit zöld káró vagy fehér kocka jelez. Aztán Alex átmegy minden kapun és elérheti a kincset.

Mivel nincs kő, amin sárga csillag van, Bea a saját járatán csak az első kapuig jut és nem mehet tovább.

Csaba járatában az utolsó kaput csak Bea tudja kinyitni, ha a kék háromszöggel jelölt kőre lép. Ám azt a követ Bea sosem érheti el.

MIÉRT INFORMATIKA?

A három felfedező ebben a feladatban három processzor, melyek párhuzamos adatokkal dolgoznak és közös zárómechanizmusokat alkalmaznak.

Egy processzor egy számítógép szíve. Ez végezi el egy program minden számítását és lehetőleg gyorsnak kell lennie. És mert mind nehezebb egy processzor sebességét növelni, a modern számítógépekben több processzor van, melyek megosztják a munkát. Az informatikusok kifejlesztették annak lehetőségét, hogy a számítások részekre bonthatóak legyenek, párhuzamosan – ez azt jelenti, hogy több processzoron egyidőben – fussanak.

Egy párhuzamos program futásakor néha egy processzornak várnia kell, amíg egy másik processzor befejezi a számításokat. Egy zár (lock) megállít egy futó programszálat. Ugyanez történik egy felfedezővel a feladatban, aki a járatot lezáró kapu előtt várakozik. Egy másik processzor (amikor egy meghatározott részfeladattal készen van) kinyithat egy zárolást, úgy mint a feladatban egy felfedező, aki a járat kapunyitó kövére lép.

HOTSPOT-PADLÓFŰTÉS (2020-PK-06)

BENJAMIN - NEHÉZ

KADÉT - KÖZEPES

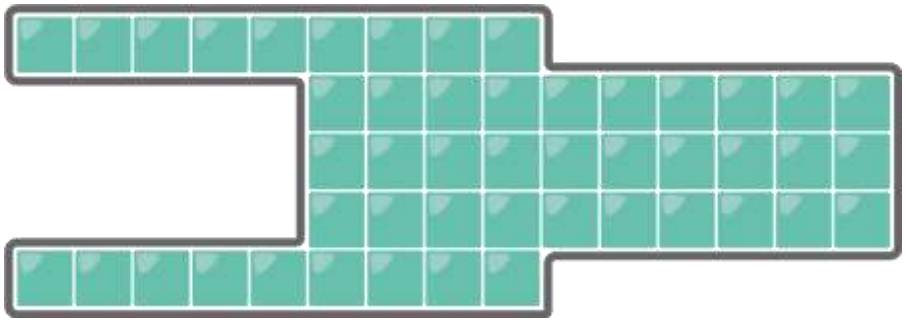
JUNIOR - KÖNNYŰ

Lajos nem szeret reggelente hideg fürdőszobában átöltözni, ezért az új házába padlófűtést szeretne beépíttetni. A fűtésszerelő az innovatív hotspot-padlófűtést ajánlja neki. Amikor bekapcsolják a hotspotot , a járólap, ami alá közvetlenül beszerelik, rögtön meleg lesz.

Egy perc múlva minden szomszédos járólap is felmelegszik, tehát minden járólap, amelyik élével vagy sarkával a már felmelegített járólapot érinti. A járólapokon szereplő szám azt mutatja, hány perc alatt melegszik fel az adott járólap.

Lajos azt szeretné, ha az új fürdőszoba minden járólapja a lehető leggyorsabban felmelegedne.

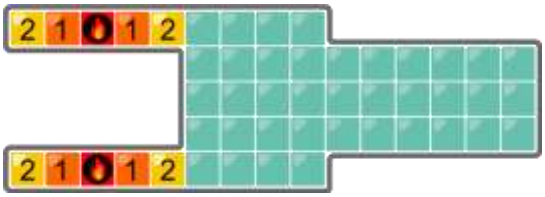
Hány hotspotot telepítsen, hogy 2 perc alatt meleg legyen az összes járólap?



A HELYES VÁLASZ A 4

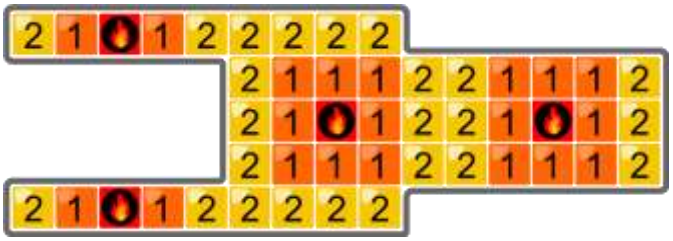
Ha 4 hotspotot telepítünk, akkor a fürdőszoba minden csempéje 2 percen belül felmelegszik.

Nem lehetséges a csempék felmelegítése négy hotspottal egy percen belül. Minden hotspot az első percben legfeljebb 9 csempét melegíthet fel, a második percben már 25 csempe lehet meleg. A négy hotspot tehát az első percben legfeljebb 36 csempét, a második percben legfeljebb 100 csempét melegít fel. A fürdőszobában 48 csempe van, ami két percen belül érhető el.



Tehát így kell a hotspotokat telepíteni, hogy minden csempe két percen belül meleg legyen. A helyiség rossz elrendezése miatt két hotspotot érdemes úgy telepíteni, hogy a két folyosót melegítse fel:

A másik két hotspotot pedig így kell telepíteni:

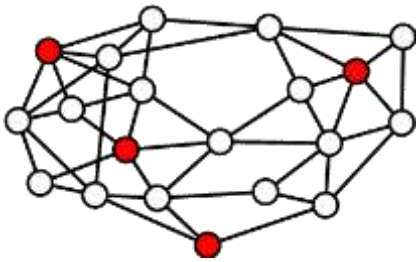


Kedvezőbb elrendezés helyiségkialakítás esetén Lajosnak csak 2 hotspotra lenne szüksége, hogy 2 percen belül meleg legyen a fürdőszobája.

MIÉRT INFORMATIKA?

Ebben a feladatban megoldott probléma egy ismert optimalizálási probléma: Itt a csomópontok legkisebb mennyiségét keressük egy gráfon belül. Ezt dominál halmaznak nevezzük.

Egy dominál halmaz definíciója a következő: A gráf minden csomópontját vagy fel kell vennünk a dominál halmazba, vagy rendelkeznie kell egy szomszéddal, amely már szerepel a dominál halmazban. A fürdőszoba járólapjai jelentik a csomópontokat. A csomópontok élekkel vannak összekötve, ha a járólapok szomszédosak. Az adott gráf egy dominál halmaza ezután jelzi azokat a helyeket, ahol hotspotok helyezhetők el, hogy a fürdőszobát 2 perc alatt felmelegítsék.



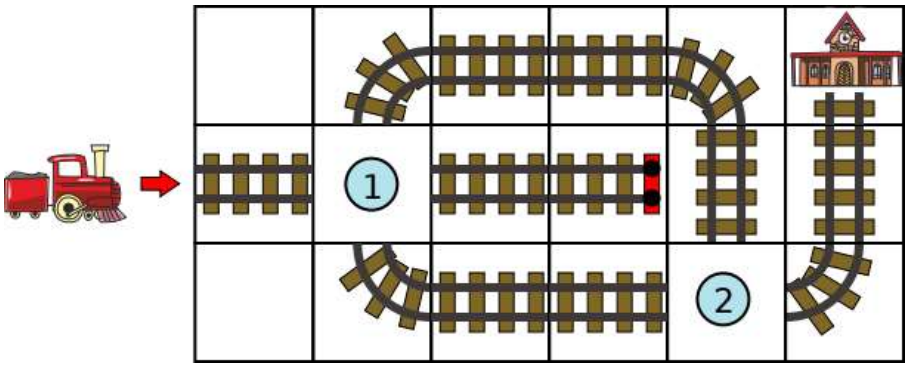
Általában nagyon nehéz egy minimális dominál halmazt megtalálni. Speciális gráfokhoz létezik hatékony algoritmus. A következő rajz mutat erre egy példát. Ahogy láthatjuk, minden fehér csomópontnak legalább egy piros szomszédja van. Tehát a piros csomópontok egy dominál halmazt alkotnak.

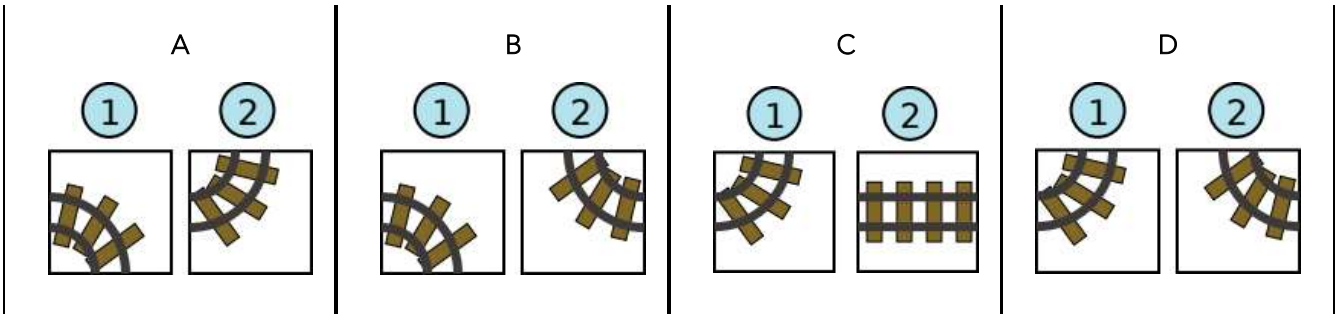
Egy tipikus alkalmazási példa a WiFi-hotspotok elhelyezése egy nagy épületben. A gráf csomópontjai az egyes szobák. Ezek közül kettő a gráfban szomszédos, ha mindkét szoba egy hotspot hatásterületén fekszik. Azok a szobák, amelyek egy minimális dominál halmazt alkotnak, megfelelő helyek a hotspot számára.

KÖVETKEZŐ MEGÁLLÓ, VASÚTÁLLOMÁS! (2020-PT-06)

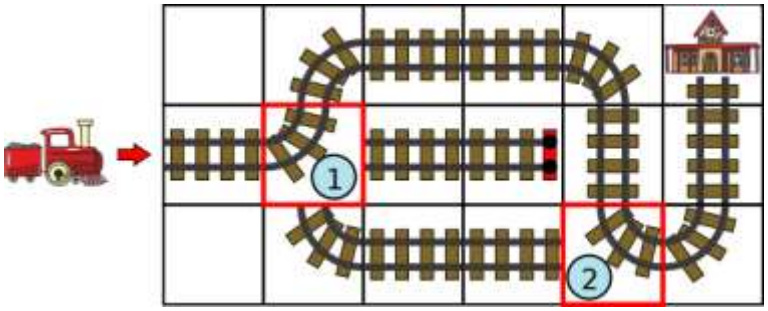
KISHÓD - KÖNNYŰ

Melyik sínek kerüljenek elforgatás nélkül a számok helyére, hogy a vonat  biztonságosan eljusson a vasútállomásra  ?

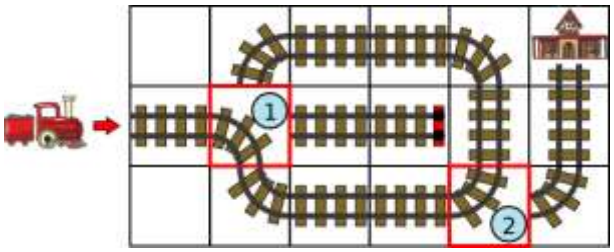


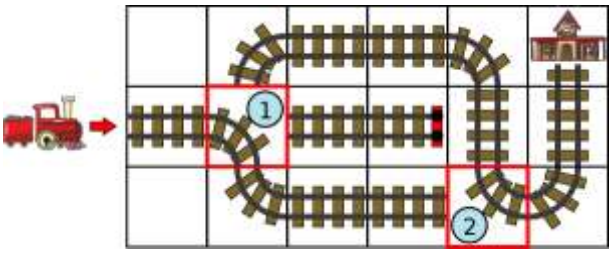


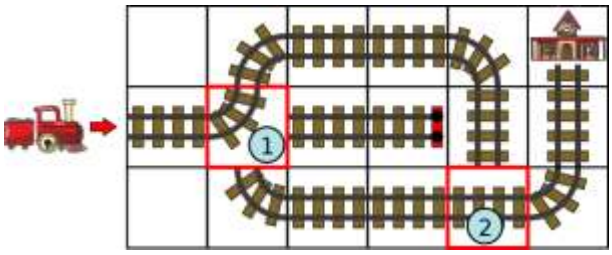
A HELYES VÁLASZ A D)



A többi esetben nincs folytonos sín a vonat indulási helyétől a vasútállomásig:







MIÉRT INFORMATIKA?

Ahogy a vonat makacsul a síneken halad úgy hajtja végre egy számítógép is makacsul egy program utasításait. Nem tudja felismerni, hogy a program hibát tartalmaz és elbukik, ahogy egy vonat kisiklik, ha a sínek rosszul vannak összerakva. Egy program írásakor tehát sokkal gondosabban kell eljárni, mint ahogy elmagyarázzuk valakinek az utat a vasútállomásra.

HAMMING A LEMMING (2020-RU-02)

SENIOR - NEHÉZ

A lemming király, Hamming beállít négy lemminget, hogy a távoli lemmingeknek zászlójeleket küldjenek. Minden ilyen hír-lemming vagy egy piros vagy egy sárga zászlót emel a magasba. Így 16 különböző üzenetet küldhetnek el.

Van olyan hír-lemming, aki nem tud jól különbséget tenni a sárga és a piros között. Megtörténhet tehát, hogy nem megfelelő zászlót emel a magasba. Ezt szeretné kiszűrni a lemming király. Ezért megbíz három segítő-lemminget.

Minden segítő-lemming három hír-lemminget ellenőriz. Ha egy üzenetküldéskor páratlan számú piros zászló van a magasban, akkor a segítő-lemming is egy piros zászlót emel fel, különben sárgát. Tehát ha minden zászló helyesen van a magasban, akkor a segítő-lemming és az általa figyelt hír-lemmingek összesen páros számú piros zászlót tartanak fel.

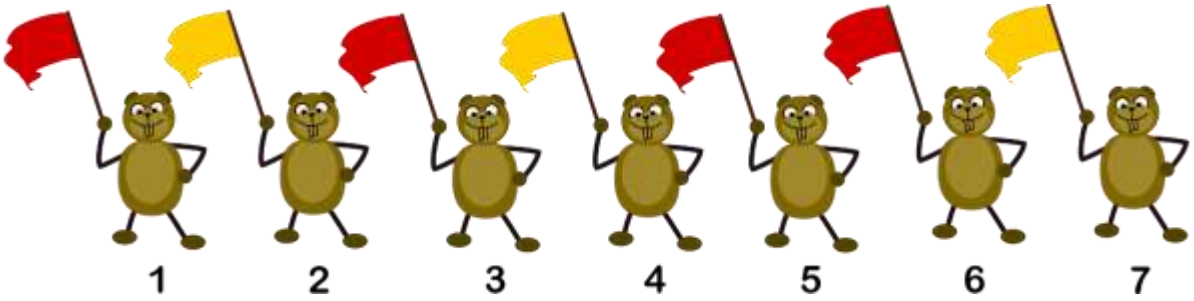
A híreket összesen hét zászlóval küldik.

A lemming király a hír-lemmingeknek és a segítő-lemmingeknek ad egy-egy számot és így rendezi őket össze:

A lemming király amint meglátja a felemelt zászlókat, rögtön tudja, hogy a hét lemming egyike rossz zászlót emelt fel.

hír-lemming	segítő-lemming
1, 2, 3	5
1, 2, 4	6
2, 3, 4	7

Melyik lemming emeli magasba helytelenül a zászlót?



A HELYES VÁLASZ: A 3-AS SZÁMÚ HÍR-LEMMING EMELI FEL HELYTELENÜL A ZÁSZLÓT

Az egyszerűség kedvéért kezeljük külön csoportként az egyes segítő-lemmingeket és az általuk figyelt hír-lemmingeket. A csoportnak adjuk ugyanazt a számot, mint a bele tartozó segítő-lemming száma. Tehát az 5. csoport az 5. számú segítő-lemming és az 1., 2. és 3. számú hír-lemming.

A lemming király által adott szabályok alapján arra következtethetünk, hogy a piros zászlók számának minden csoportban párosnak kell lennie. Ha például az 1-es, 2-es és 3-as hír-lemmingek felemelt zászlói között a pirosak száma páratlan, akkor a segítő-lemming piros zászlója hozzáadódik, és ez összességében páros szám. Ugyanez vonatkozik az összes többi csoportra.

Vizsgáljuk meg ennek fényében az üzenetet, azaz a felemelt zászlókat.

Megállapíthatjuk, hogy valami nem stimmel az üzenettel, mert nem minden csoportban van páros számú piros zászló.

Ha valamelyik segítő-lemming emelt volna fel rossz zászlót, akkor csak ebben az egy csoportban lenne páratlan a piros zászlók száma. Mivel több csoportban is így van, arra a következtetésre juthatunk, hogy az 1-4 hír-lemmingek egyike tévesztett.

Ha az 1-4. hír-lemmingek valamelyike rossz zászlót tart, akkor minden olyan csoportban, amelyben ez a lemming található páratlan lesz a piros zászlók száma. Tehát meg kell találnunk egy olyan hír-lemminget, aki pontosan azokba a csoportokba tartozik, amelyekben páratlan a piros számú zászlók száma (5. és 7.). Ez ebben az esetben csak a 3. számú hír-lemmingre igaz.

Érdemes elgondolkodni a feladaton akkor is, ha nem tudjuk biztosan, hogy csak egy lemming tévedett.

MIÉRT INFORMATIKA?

Az ebben a feladatban használt megoldás jó példa a hibajavító kódra. Az egyik első hibajavító kód a Hamming-kód volt, amely pontosan úgy működik, mint a példánkban.

Az üzeneteket gyakran bináris formátumban (0 és 1 lánc) továbbítják különféle kommunikációs csatornákon. Ez meghibásodásokhoz vezethet, amely során az egyes "0" értékek "1" értékké változhatnak és fordítva. Sok esetben elengedhetetlen, hogy meg lehessen állapítani, hogy a kapott információt érte-e ilyen sérülés, és külön feladat, hogy ki is tudjuk javítani. Ezért olyan fontosak a hibajavító kódok.

Nagyon egyszerű lehetőség lenne minden egyes bitet (0-t vagy 1-est) háromszor elküldeni. Az egy bites zavar könnyen felismerhető és kijavítható, mert a másik két bit továbbra is a helyes információt hordozza. Ez az egyszerű eljárás azonban azt jelenti, hogy háromszor annyi adatot kell továbbítani.

A hibajavító kódok elméletének egy része azzal a kérdéssel foglalkozik, hogy a lehető legkevesebb kiegészítő adattal hogyan lehet jó hibajavító tulajdonságokat elérni. A példánkban használt Hamming-kód az üzenetben szereplő bitcsoportok összegének páros számát (paritását) használja a további bitekkel együtt. Ha az összes csoport paritása helyes, akkor feltételezhetjük, hogy az üzenetet helyesen továbbították. Ha nem minden paritás helyes, akkor az üzenetet „megzavarták”. Ha csak egy bitet változtattak meg, akkor az információk felhasználásával meghatározható, hogy mely csoportok érintettek, melyik a megváltozott bit és mi az eredeti üzenet. Ez sok esetben elég jól működik.

Megfelelő módszerek alkalmazásával azonban lehetséges robusztusabb kód létrehozása is a lehetséges hibák felderítésére és kiküszöbölésére.

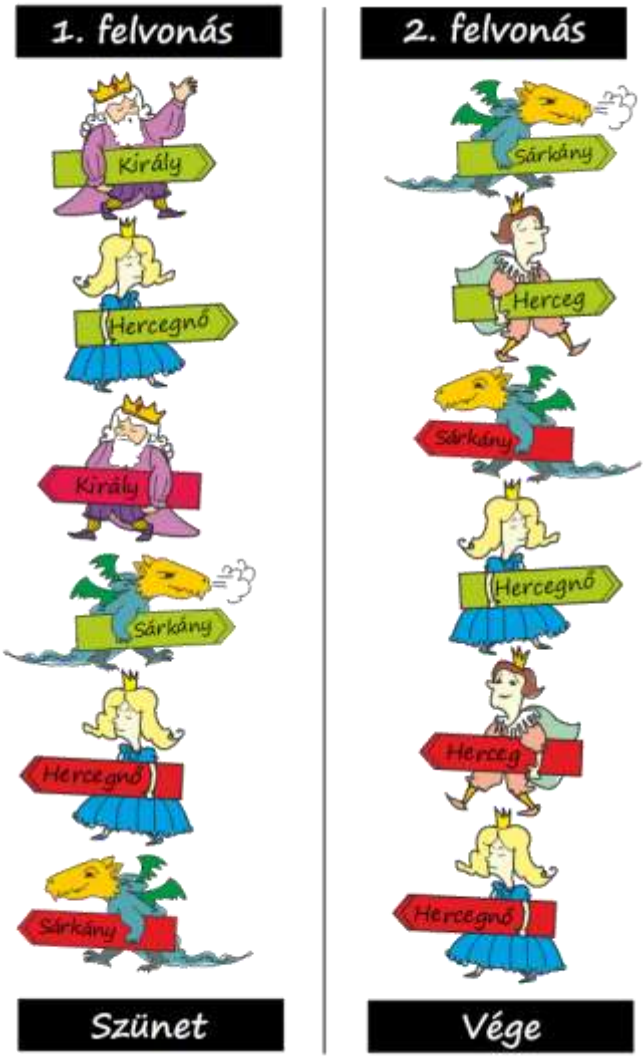
SZÍNDARAB (2020-SK-01)

KISHÓD - KÖZEPES

BENJAMIN - KÖNNYŰ

Egy színdarabban az előadás során a szereplők a következő sorrendben

- lépnek színpadra 
- hagyják el azt 



Mi nem történik meg?

A) A hercegnő és a herceg együtt van a színpadon.

B) A király és a sárkány együtt van a színpadon.

C) A herceg csak a szünet után lép színre.

D) A herceg és a sárkány együtt van a színpadon.

A HELYES VÁLASZ A B): A "KIRÁLY ÉS A SÁRKÁNY EGYÜTT VAN A SZÍNPADON" NEM LEHETSÉGES.

Nézzük lépésről-lépésre:

Cselevés	*Király a színpadon?*	*Hercegnő a színpadon?*	*Sárkány a színpadon?*	*Herceg a színpadon?*	*Megjegyzés*
első felvonás					
	igen	nem	nem	nem	
	igen	igen	nem	nem	
	nem	igen	nem	nem	
	nem	igen	igen	nem	
	nem	nem	igen	nem	
	nem	nem	nem	nem	
szünet					
második felvonás					
	nem	nem	igen	nem	
	nem	nem	igen	igen	C), D)
	nem	nem	nem	igen	
	nem	igen	nem	igen	A)
	nem	igen	nem	nem	
	nem	nem	nem	nem	
vége					

Egy válasz megfelelő (megtörténik), ha a táblázatban egy sor megerősíti ezt.

Az „A" válasznál olyan sort keresünk, ahol a hercegnő is és a herceg is együtt van a színpadon. Ez a második felvonás negyedik sora, amikor a második sorban a herceg színre lép és az ötödik sorig ott is marad, mialatt a hercegnő a negyedik sorban színre lép. Tehát az „A" válasz helyes.

A „D" válasznál olyan sort keresünk, ahol a herceg is és a sárkány is együtt van színpadon. Ez a második felvonás második sora, amikor az első sorban a sárkány színre lép és a harmadik sorig ott is marad, mialatt a herceg a második sorban a színre lép. Tehát a „D" válasz helyes.

A „C" válasznál egész más a kijelentés. Ha ez helyes, akkor a herceg az egész első felvonás alatt nem léphet színre. Itt a herceg oszlopát kell végignézni az első felvonásban. Itt mindenhol „nem" áll, tehát a herceg az első felvonásban nem lép színre. Csak a második felvonás második sorában lép színre, tehát a „C" válasz szintén helyes.

Ha a „B" válasznak helyesnek kell lennie, akkor legalább egy sorban a királynak is és a sárkánynak is együtt kell színpadon lennie. A tizenkét sor egyikében sincs mindkét oszlopban egyszerre „igen". Sőt a király már az első felvonás harmadik sorában elhagyja a színpadot és a végéig nem lép újra a színre. A sárkány

ellenben csak az első felvonás negyedik sorában lép színre. Talán a színfalak mögött találkoznak, de a színpadon nincsenek együtt. Emiatt a „B" kijelentés nem igaz, így ez a keresett válasz.

MIÉRT INFORMATIKA?

Még ha élénken el is tudjuk képzelni a történetet a színdarab alapján, a feladat szempontjából csak egy fontos az egyes szereplők tulajdonságai közül: a színpadon van-e egy bizonyos időpontban, vagy sem? A pillanatnak ezt az egy tulajdonságra való korlátozását nevezzük *absztrakciónak*. Az emberek az absztrakció mesterei, folyamatosan a fontos dolgokra koncentrálunk és figyelmen kívül hagyunk minden mást magunk körül. A művészet az, hogy tudjuk, mire kell koncentrálnunk.

Az informatikában még egy lépéssel tovább mehetünk és a négy szereplő helyét változóként foghatjuk fel, melyek egy meghatározott időpontban egy meghatározott értéket kapnak. A válaszmagyarázó táblázat pontosan így van felépítve: a négy változónak („Király a színen?", „Hercegnő a színen?", „Sárkány a színen?" és „Herceg a színen?") minden sorban van egy értéke „igen" vagy „nem". Az idővel azonban a változó értéke változik.

A matematikában nagy erőfeszítéseket tesznek időnként, hogy ez mindig így is maradjon.

Miközben az emberek számára a változókban és azok értékeiben való absztrakt gondolkodás sokszor nehéz, a számítógép nagyon jól és gyorsan boldogul a változókkal, így könnyen elvégezhet sok ilyen számítást az emberek számára. Cserébe az emberek konkrét helyzetekben sokkal egyszerűbben hoznak létre értelmes modelleket, mint a számítógépek.

Ha az emberi képességeket és a számítógépes lehetőségeket kombináljuk, valóban nehéz feladatokat is meg tudunk oldani.

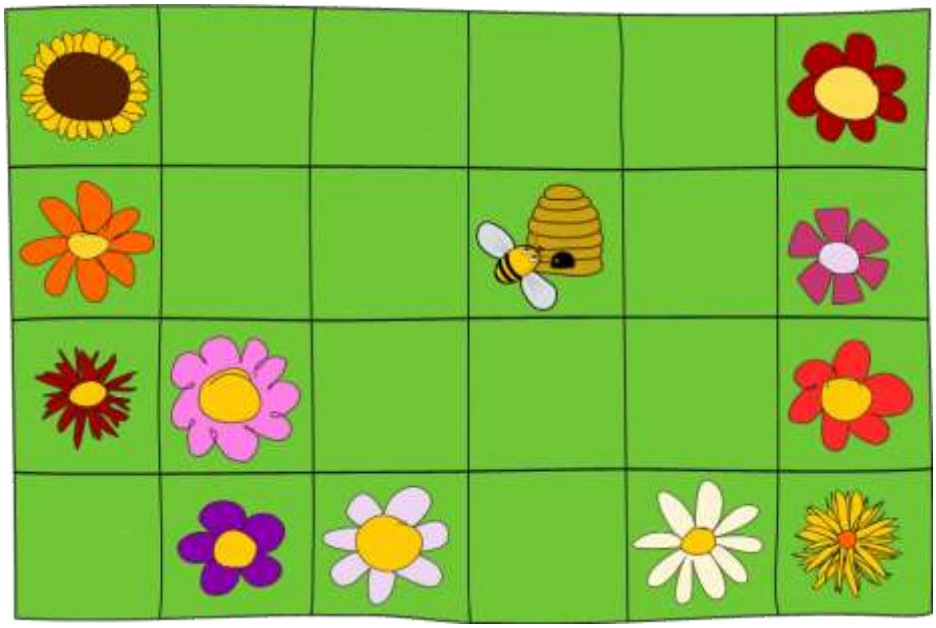
ZÜM, ZÜM, ZÜM, ... (2020-SK-04)

KISHÓD - KÖNNYŰ

BENJAMIN - KÖNNYŰ

Egy méh  10 perc alatt repül át egy mezőt fel, le, balra és jobbra (átlósan nem). A méhkaptártól  legfeljebb egy fél órát repül, mielőtt visszafordul.

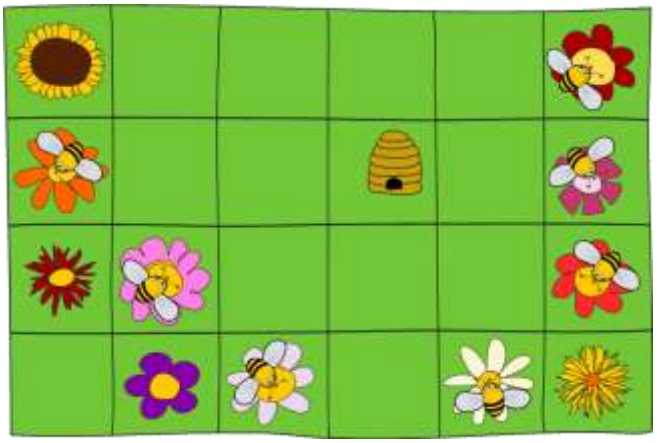
Melyik virág NEM érhető el egy fél órán belül a méhkaptártól?



A)	B)	C)	D)
			

A HELYES VÁLASZ AZ A)

A képen láthatod, melyek azok a virágok, melyeket a méh a méhkaptártól fél órán belül elér:



Az alábbi kép minden mezőn megmutatja, hány percre van szüksége a méhkaptártól a méhnek, hogy elérje. Fél órán belül tehát elérhető minden mező, amelyiken 10, 20 vagy 30 áll.

A számokkal való kitöltés így működik: A méhkaptár melletti mezőbe 10-est írunk, mert a méhnek 10 percre van szüksége, hogy odarepüljön. Aztán 20-ast írunk minden üres mezőbe a 10-es mezők mellett, mert a méhnek 10 perc kell, hogy egyik mezőről a másikra repüljön. Ezt így folytatjuk tovább. Tehát 30-ast írunk minden 20-as melletti üres mezőbe. Aztán 40-est írunk minden, 30-as melletti üres mezőbe. Végül 50-est írunk minden 40-es melletti üres mezőbe.

MIÉRT INFORMATIKA?

A feladat megoldásához a már kiszámított és elmentett eredményeket (a kitöltött mezők száma) felhasználjuk a további eredmények kiszámításához (a szomszédos, még üres mezők száma). Ezt a nagyon általános elvet *dinamikus programozás*nak hívják. Általában fontos, hogy az eredmények milyen sorrendben kerülnek kiszámításra.

A megoldásban szereplő eljárás egy úgynevezett *mélységkorlátozott szélességi keresés*nek is tekinthető egy gráfban. Ehhez a mezőket *csomópont*nak tekintjük, összekötjük a szomszédos mezőket egy *él*lel, és virágokat kezdünk keresni a méhkastól kiindulva. A mélységkorlátozás az időkorlátunk.

AZ ERDŐ REJTETT SZÉPSÉGEI (2020-TR-03)

KISHÓD - KÖZEPES

BENJAMIN - KÖNNYŰ

Emma és Félix felfedeztek egy bokrot sok állattal.

Az állatok négy különböző ágon ülnek:

- Az ágnak, amelyiken a hernyó  ül, van a legtöbb levele.
- Az ágnak, amelyiken a lepke  ül, több levele van, mint annak az ágnak, amelyiken a csiga  ül.
- Annak az ágnak, amelyiken a katicabogár  ül, pontosan egy levele van.

Melyik faágon ülhet a csiga  **?**

A

B

C

D

A HELYES VÁLASZ A B): A KÉTLEVELES ÁGON ÜL A CSIGA

A katicabogár  ül azon az ágon, amelyiknek egy levele van (azaz az „A"-ban szereplő ágon)

A „D" válaszban szereplő ágnak van a legtöbb levele, összesen öt. Ezen ül a hernyó  .

Mivel a lepke ágán több levél van, mint a csigáén, ezért a lepke  ül a „C" válaszban található 3 leveles ágon és a csiga  a „B" válaszban leírt 2 levelesen.

MIÉRT INFORMATIKA?

Ez logikai probléma. Kaptunk egy sor állítást, és a feladat az, hogy úgy határozzuk meg az állatok elrendezését, hogy minden állítás igaz legyen. A logika az állítások tanulmányozása. Annak megállapítása, hogy mindig, soha vagy néha igazak-e, illetve az igaz állítások alapján következtetések levonása.

GYŐZTESEK ÉS VESZTESEK (2020-VN-01)

KADÉT - NEHÉZ

SENIOR - KÖNNYŰ

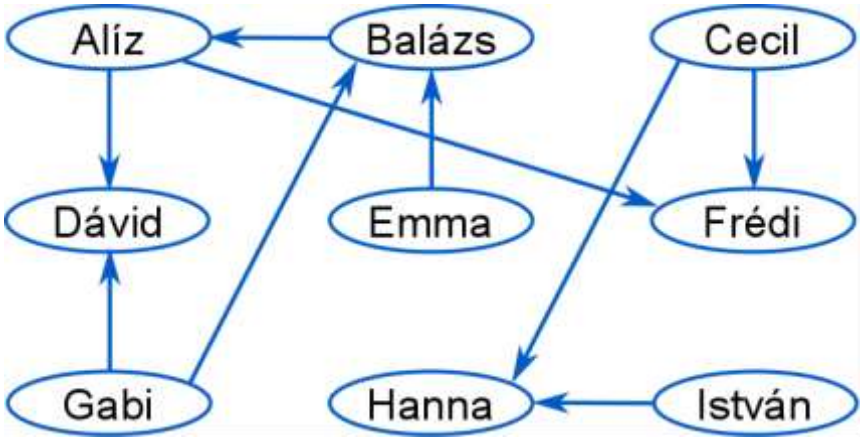
Egy szabadtéri sportversenyen már néhány mérkőzést lejátszottak. A lenti ábrán látható nyilak azt mutatják, ki kit győzött le. Például Balázs legyőzte Alízt, Alíz legyőzte Dávidot.

Sajnos az idő elromlott, és nem tudnak minden tervezett mérkőzést lejátszani.

Annak kell tehát a verseny győztesének lennie, aki jobb mindenki másnál. Ha A győz B ellen egy mérkőzésen, akkor érvényes az, hogy:

- A jobb, mint B.
- Ha B jobb, mint másik néhány játékos, akkor A jobb ezeknél a játékosoknál is.

Pillanatnyilag több olyan játékos van, aki a verseny győztese lehet.

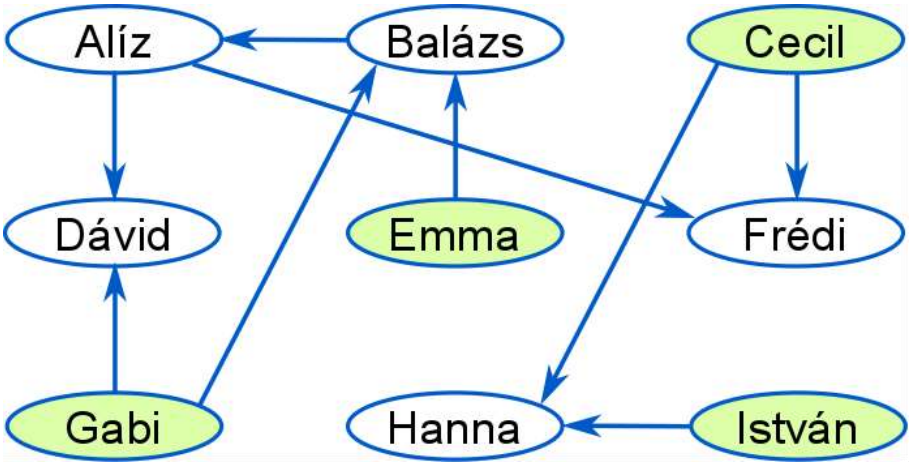


Melyik játékos lehet a verseny győztese?

A HELYES VÁLASZ: CECIL, EMMA, GABI ÉS ISTVÁN LEHET GYŐZTES

Minden játékos, akinél nincs másik jobb játékos, lehet a verseny győztese.

Minden játékosnál, aki már egyszer vesztett, van egy jobb játékos. Tehát csak az a játékos lehet a verseny győztese, aki eddig nem vesztett mérkőzést. Tehát azok, akikre nem mutat nyíl: Cecil, Emma, Gabi és István.



MIÉRT INFORMATIKA?

Az irányított gráf (vagy digráf) az objektumpárok közötti (nem szimmetrikus) kapcsolatok ábrázolásának módja: az objektumokat csúcsként ábrázolják, és az A objektumot egy nyíl kapcsolja a B objektumhoz, ha A kapcsolatban áll B-vel.

Ebben a feladatban a kapcsolat a „győztes”: egyes kapcsolatok a ténylegesen lejátszott mérkőzésekből származnak, mások a szövegben megadott szabályból, amely tranzitívvé teszi a győzelmet. Ez azonban általában nem igaz a sportra: a mérkőzések valóban izgalmasak, mert ha Balázs legyőzi Alízt és Alíz legyőzi Dávidot, semmi sem garantálja, hogy Balázs egy valódi mérkőzésen legyőzze Dávidot.

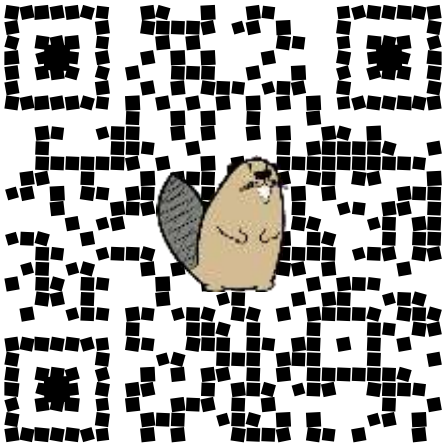
A feladat problémája azonban általános az informatikában: egyes tárgyak nem hasonlíthatóak közvetlenül egymáshoz, és mégis olyan rendezést kell felépítenünk, ahol minden objektumot páronként összehasonlítunk: természetesen több ilyen rendezés is lehet. Példaként említhetjük, ha vannak olyan tevékenységek vagy programok, amelyek másoktól függenek: cipő felvételéhez már fel kellett volna venni a zoknit, és egy kabát felvételéhez már fel kellett volna venni egy inget ... de biztosan többféle, megfelelő öltözködési sorrend létezik!

TÁMOGATÓINK, KÖSZÖNETNYILVÁNÍTÁS

Köszönjük a nemzetközi Bebras kezdeményezés országainak, kiemelten a DACH-csapatnak (Németország, Ausztria, Svájc),

az ELTE IK hallgatóinak, illetve

a kapcsolattartó tanároknak szervezői munkájukat.



Eötvös Loránd Tudományegyetem
Informatikai Kar