Bober 2015/16

Naloge in rešitve

Naloge za tekmovanje je izbral, prevedel, priredil in oblikoval Programski svet tekmovanja:

Alenka Kavčič (FRI, Univerza v Ljubljani)
Andreja Filipič (Osnovna šola Spodnja Idrija)
Janez Demšar (FRI, Univerza v Ljubljani)
Matej Črepinšek (FERI, Univerza v Mariboru)
Nataša Kermc (Osnovna šola Bistrica ob Sotli)
Špela Cerar (PeF, Univerza v Ljubljani)

Razvoj tekmovalnega sistema:

Gašper Fele Žorž (FRI, Univerza v Ljubljani)

Organizacijski odbor državnega tekmovanja:

Alen Ajanović, Andrej Brodnik, Janez Demšar, Valentin Kragelj, Nataša Mori, Jure Rogelj (FRI, Univerza v Ljubljani)

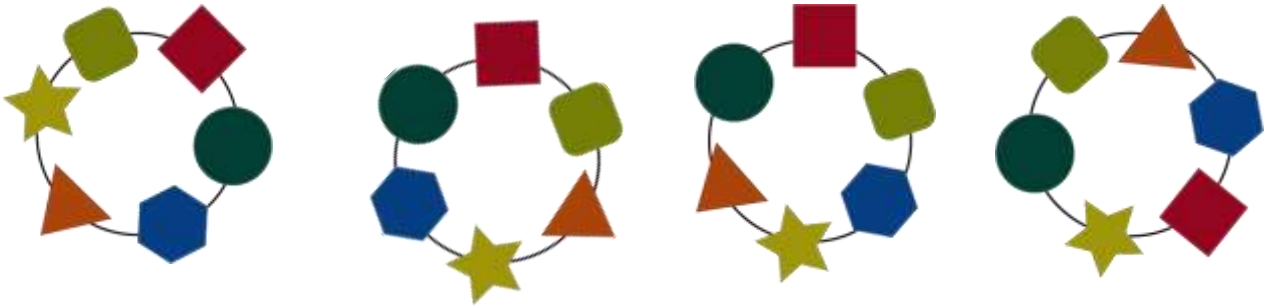
Kazalo nalog

Zapestnica

2. – 3. razred

BOBROVKI EMI SE JE STRGALA ZAPESTNICA. VIDETI JE TAKO:

ZDAJ KUPUJE NOVO. KATERA OD SPODNJIH JE ENAKA TISTI, KI JO JE IMELA PREJ?



REŠITEV

DRUGA ZAPESTNICA.

V PRVI STA TRIKOTNIK IN ZVEZDA ZAMENJALA MESTI.

V TRETJI STA TRIKOTNIK IN ŠESTKOTNIK NA NAPAČNIH MESTIH.

V ČETRTI STA ZAMENJANA ZELENA ZVEZDA IN ŠTIRIKOTNIK.

Računalniško ozadje

Prepoznani vzorci pomagajo najti podobnosti v stvareh, ki so sprva videti različne, a imajo kaj skupnega. Ob spoznanju, da je nov problem podoben problemu, ki ga že znamo rešiti, lahko na podoben način poskusimo rešiti tudi novi problem. To je uporabno tudi v matematiki in na drugih področjih znanosti.

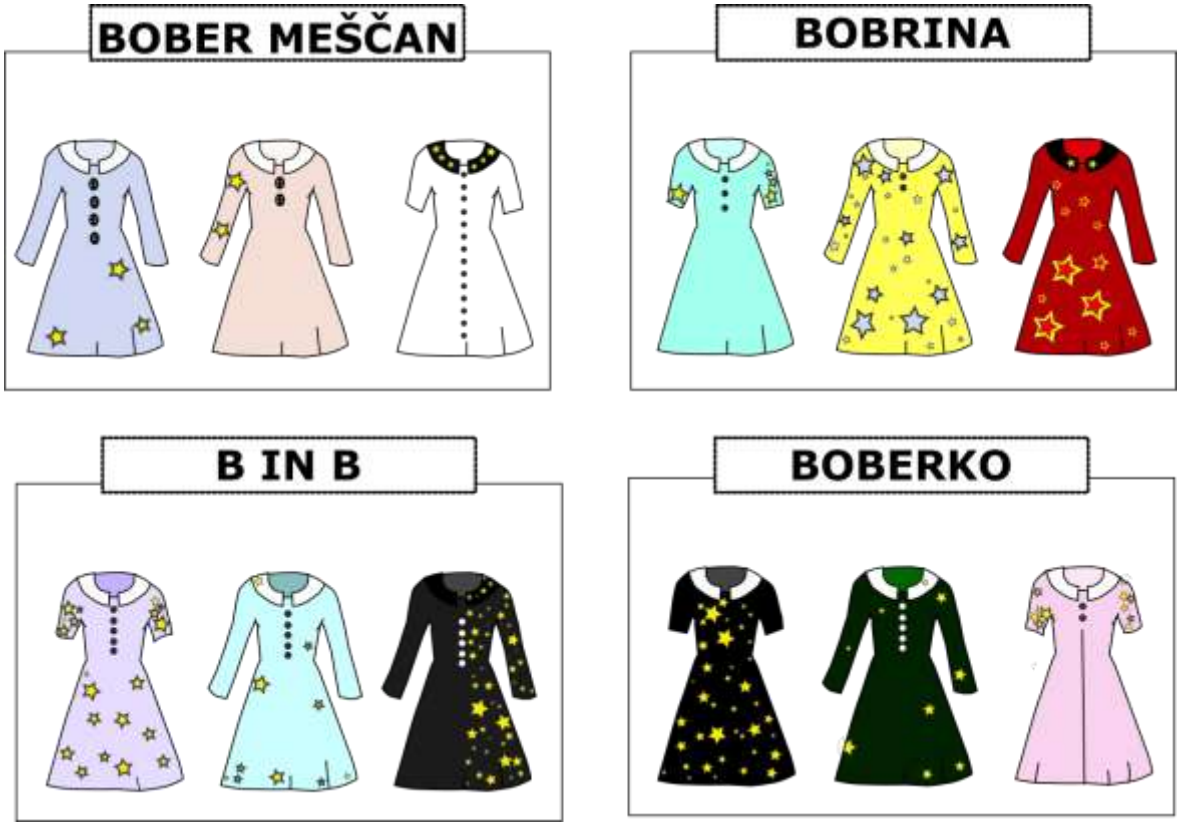
Sanjska obleka

2. – 3. razred

BOBROVKA KATJA BI RADA KUPILA SANJSKO OBLEKO. OBLEKA MORA IMETI

- KRATKE ROKAVE,
- VEČ KOT TRI GUMBE,
- ZVEZDICE NA ROKAVIH.

V KATERI TRGOVINI PRODAJAJO KATJINO SANJSKO OBLEKO?



REŠITEV

PRAVILNI ODGOVOR JE »B IN B«. PRI REŠEVANJU NALOGE SI MORAL HKRATI UPOŠTEVATI TRI ZAHTEVE. TO SI STORIL TAKO, DA SI IZLOČIL OBLEKE, KI NE IZPOLNJUJEJO KATEREKOLI OD ZAHTEV. PO OPRAVLJENEM IZLOČANJU JE OSTALA V TRGOVINI “B IN B” PRVA OBLEKA.

Računalniško ozadje

Naloga vsebuje izjave oz. pogoje, ki jih je potrebno oceniti kot resnične ali neresnične pri vsaki obleki. Pogoji in njihov izračun so pomemben del programiranja in algoritmičnega razmišljanja. Pogoje lahko sestavljamo s pomočjo logičnih *operatorjev*, kot so IN, ALI, NE. V tej nalogi je uporabljen operator IN.

Pogrinjek

2. – 3. razred

BOBER BOB JE PRIPRAVIL POGRINJEK. V KAKŠNEM VRSTNEM REDU JE POSTAVIL PREDMETE NA MIZO?

A. PRT, PRTIČEK, SKODELICA S KROŽNIČKOM, NOŽ, KROŽNIK
B. PRT, SKODELICA S KROŽNIČKOM, PRTIČEK, KROŽNIK, NOŽ
C. PRT, PRTIČEK, SKODELICA S KROŽNIČKOM, KROŽNIK, NOŽ
D. PRTIČEK, NOŽ, PRT, SKODELICA S KROŽNIČKOM, KROŽNIK

REŠITEV

PRAVILEN JE ODGOVOR B.

- NAJPREJ JE POSTAVIL PRT, SAJ SO VSE OSTALE STVARI NA NJEM.
- NATO JE POSTAVIL SKODELICO S KROŽNIČKOM.
- NATO JE DODAL PRTIČEK, SAJ VIDIMO, DA JE NAD KROŽNIČKOM.
- KROŽNIK JE POSTAVIL NA PRTIČEK,
- NOŽ JE NA KROŽNIKU.

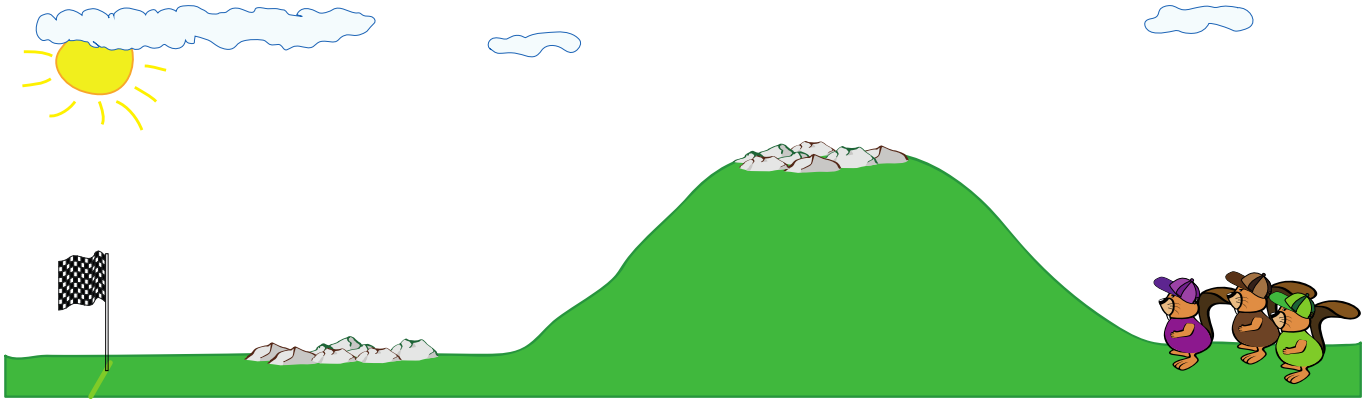
Računalniško ozadje

Pri programiranju moramo razmišljati o tem, v kakšnem vrstnem redu izvajati ukaze, da bomo dobili želeni rezultat. Kadar programi ne delujejo, pa pogosto razmišljamo tudi v obratni smeri: opazujemo (napačni) rezultat programa in poskušamo odkriti, kako je prišlo do njega..

Čez drn in strn

2. – 5. razred

BOBRI ROZIKA, BORUT IN ZALA SE BODO POMERILI V TEKU. NAJPREJ TEČEJO NAVZGOR, NATO ČEZ SKALE, NAVZDOL IN SPET ČEZ SKALE.



ZAČELI BODO V VRSTNEM REDU ROZIKA, BORUT, ZALA.

BORUT BO PREHITEL ENEGA TEKAČA PRED SEBOJ MED TEKOM NAVZGOR.

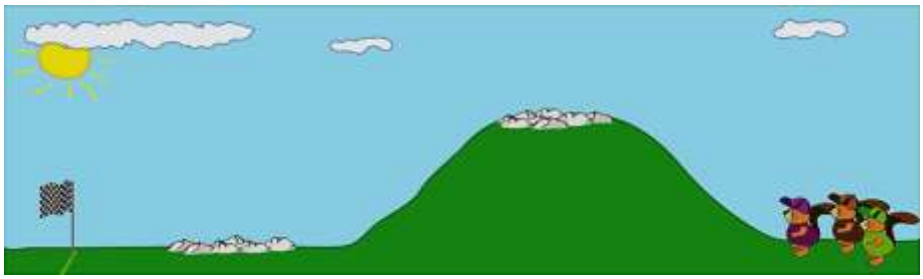
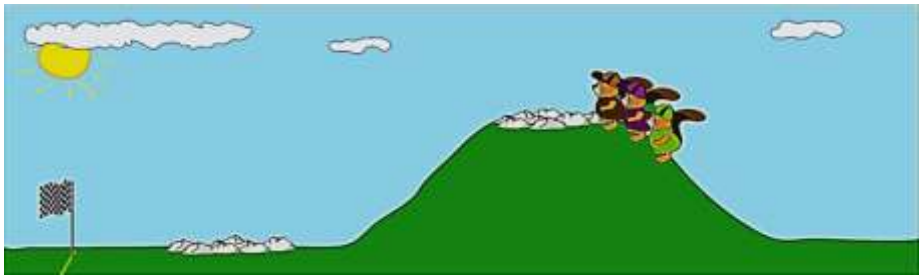
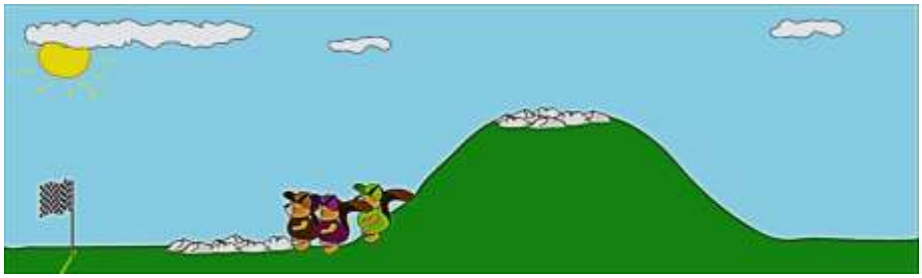
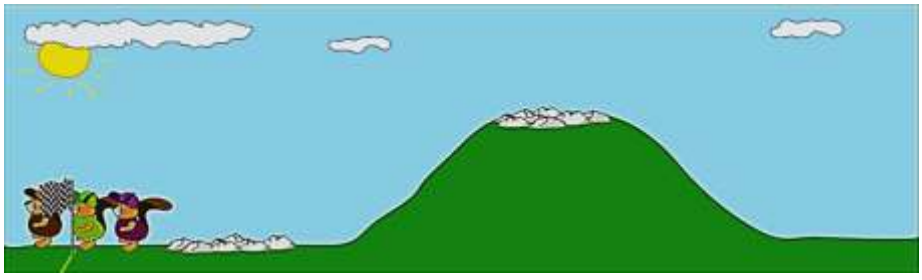
ROZIKA BO PREHITELA ENEGA TEKAČA PRED SEBOJ MED TEKOM NAVZDOL.

ZALA BO PREHITELA ENEGA TEKAČA PRED SEBOJ MED VSAKIM TEKOM ČEZ SKALE

V KAKŠNEM VRSTNEM REDU BODO KONČALI TEKMOVANJE?

REŠITEV

1. BORUT
2. ZALA
3. ROZIKA

ZAČETEK	1. ROZIKA 2. BORUT 3. ZALA	
NAVZGOR BORUT PREHITI ROZIKO	1. BORUT 2. ROZIKA 3. ZALA	
SKALE ZALA PREHITI ROZIKO	1. BORUT 2. ZALA 3. ROZIKA	
NAVZDOL ROZIKA PREHITI ZALO	1. BORUT 2. ROZIKA 3. ZALA	
SKALE ZALA PREHITI ROZIKO	**V CILJU:** 1. BORUT 2. ZALA 3. ROZIKA	

Računalniško ozadje

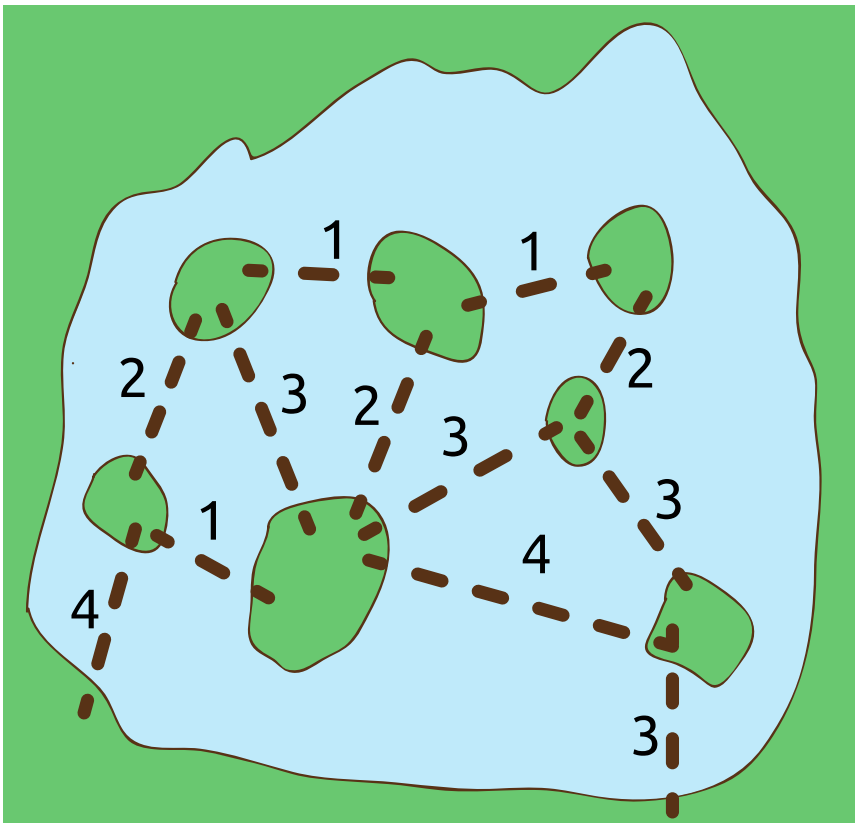
Programerji morajo pogosto pregledovati, kako deluje njihov program – posebej, če ne deluje pravilno. Takrat morajo slediti njegovemu delovanju vrstico za vrstico.

Naloga Čez drn in strn je podobna. Imaš nekaj podatkov, to je zaporedje tekačev. V programu poznaš korake: vzpon, skale, spust, skale. Natančno moraš spremljati posledico vsakega koraka v zaporedju in slediti delovanju programa, to je odkriti končni vrstni red v cilju.

Povezani otoki

3. – 5. razred

Na jezeru je sedem otokov. Črte kažejo, kje je možno postaviti mostove. Številke povejo, koliko hlodov je potrebnih za vsak most. Bobri se morajo odločiti, kako zgraditi mostove, da bi z obale dosegli vse otoke brez plavanja.

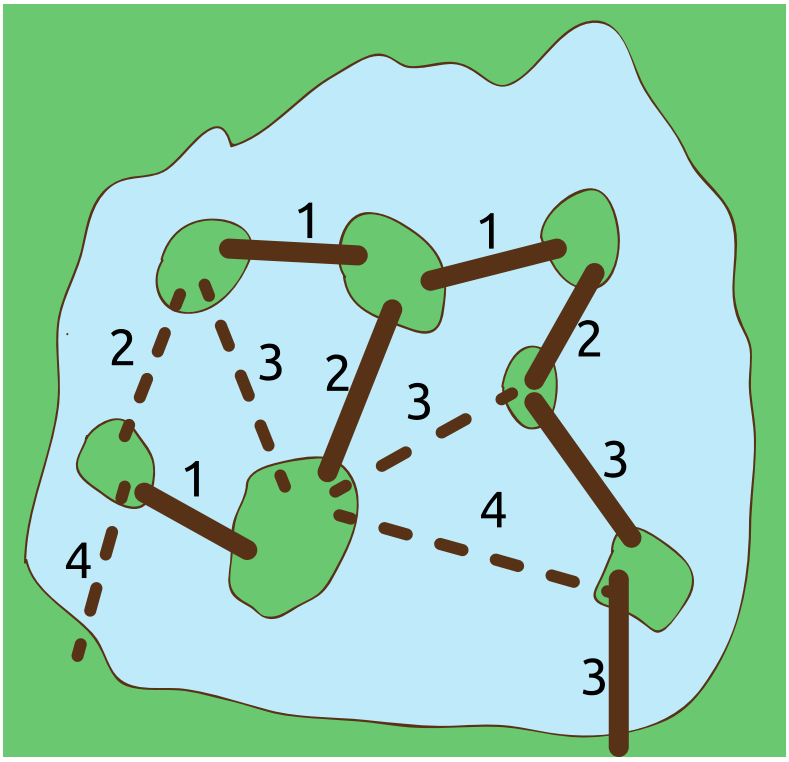


Najmanj koliko hlodov potrebujejo za izgradnjo mostov?

Rešitev

13 dreves

Da bi uporabili čim manj hlodov, bi morali zgraditi mostove, označene na spodnji sliki. Most iz dveh hlodov lahko nadomestijo z drugim mostom iz dveh hlodov. V vsakem primeru potrebujejo 1 + 1 + 1 + 2 + 2 + 3 + 3 = 13 hlodov.



Takšne naloge se je najboljše lotiti tako, da najprej povežemo obalo z enim od otokov. Izbiramo lahko med levim mostom, za katerega potrebujemo štiri hlode, in desnim, za katerega potrebujemo tri. Seveda izberemo drugega.

Ta otok povežemo s tistim sosednjim otokom, do katerega potrebujemo dva hloda.

Tako nadaljujemo do konca: v vsakem koraku povežemo obalo ali enega od že povezanih otokov s kakim še nepovezanim otokom. Pri tem vedno izberemo tisto povezavo, ki zahteva najmanj hlodov.

Računalniško ozadje

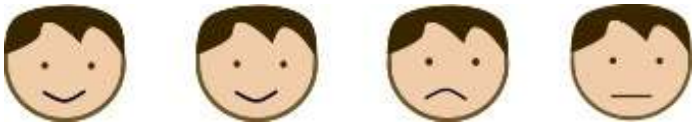
V nalogi rešujemo znan problem, ki se v različnih preoblekah pojavlja na najrazličnejših področjih matematike, računalništva in drugje. Imenujemo ga problem *iskanja minimalnega vpetega drevesa v grafu*. Za reševanje problema obstaja več postopkov. Ta, ki ga opisujemo zgoraj, se imenuje Primov algoritem. Drugi znani postopek je Kruskalov algoritem.

Obrazi z očali

3. – 5. razred

Vsak izraz na obrazu se ujema z eno obliko očal.

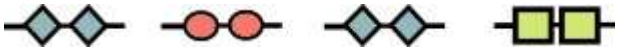
Tule imamo štiri obraze.



Očala jim bomo razdelili na enega od spodnjih načinov. Na katerega bomo naredili najmanj napak?

A. 

B. 

C. 

D. 

Rešitev

Najboljša možnost je B, saj so napačna le ena očala, namreč prva.

Računalniško ozadje

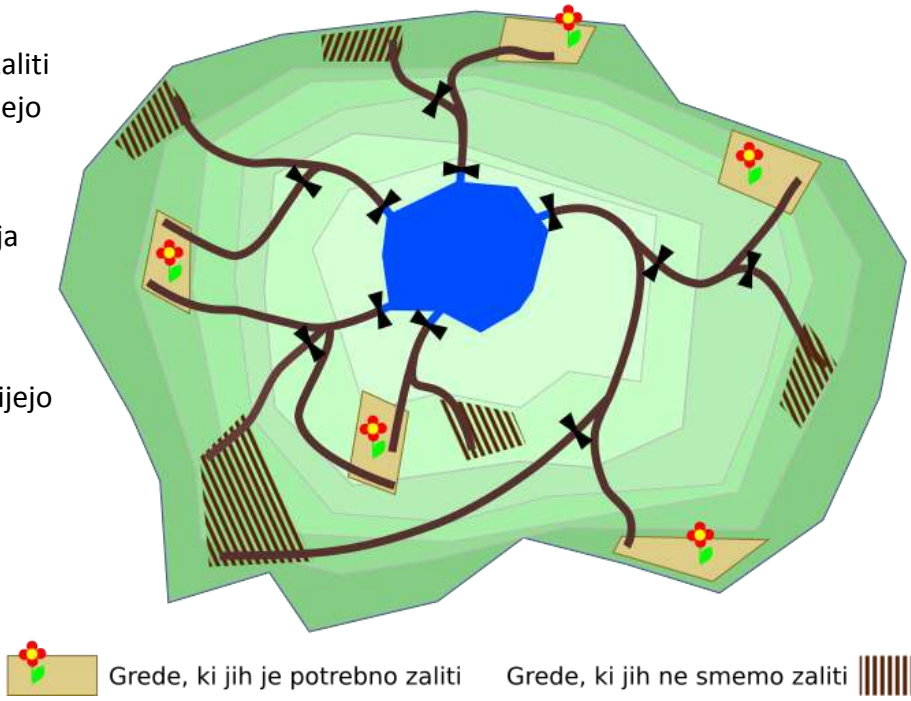
Številu razlik med dvema zaporedjema pravimo Hammingova razdalja. Uporabljamo jo, recimo, v genetiki, kjer s preštevanjem razlik v genskem zapisu določamo, kako različna sta dva gena.

Zalivanje

3. – 7. razred

Bobrova družina Breznik želi zaliti svoja cvetlična polja. Če dvignejo vodno zaporo (⋈) bo voda tekla po vodovodnih ceveh iz jezera na vrhu hriba do vznožja hriba.

Pomagaj jim! Katere vodne zapore morajo dvigniti, da zalijejo **samo** cvetlična polja?



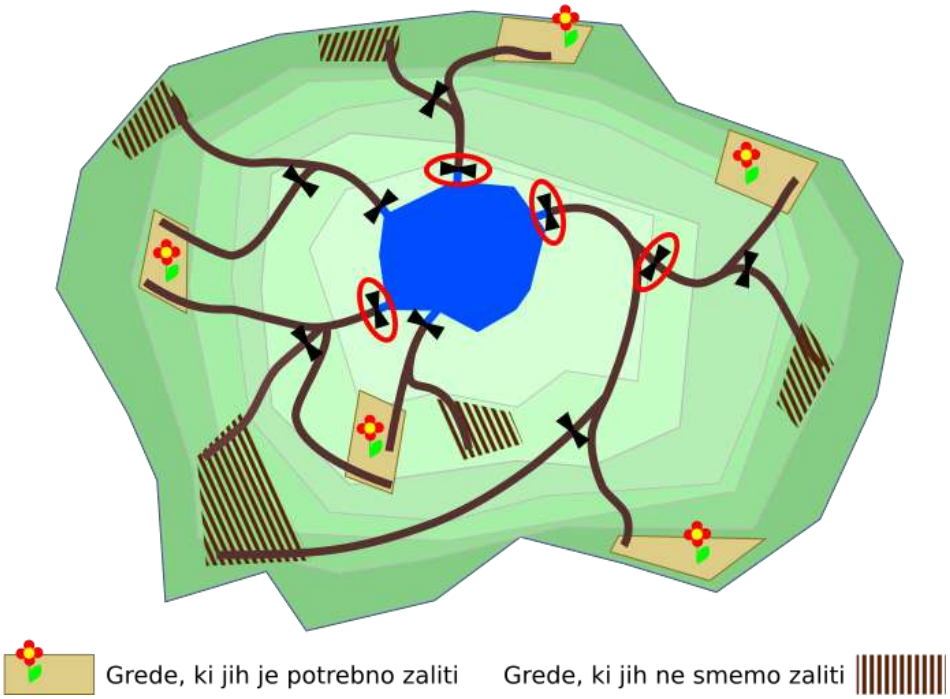
Rešitev

Rešitev je na sliki.

Ob reševanju naloge si moral biti pozoren na to, da lahko voda do nekaterih polj priteče po različnih ceveh in da lahko ena cev zalije več polj.

Računalniško ozadje

Zapore so podobne logičnim vratom, ki so osnova računalniških vezij. Ogromno takih vrat je v procesorjih med seboj ustrezno povezanih v vezja.

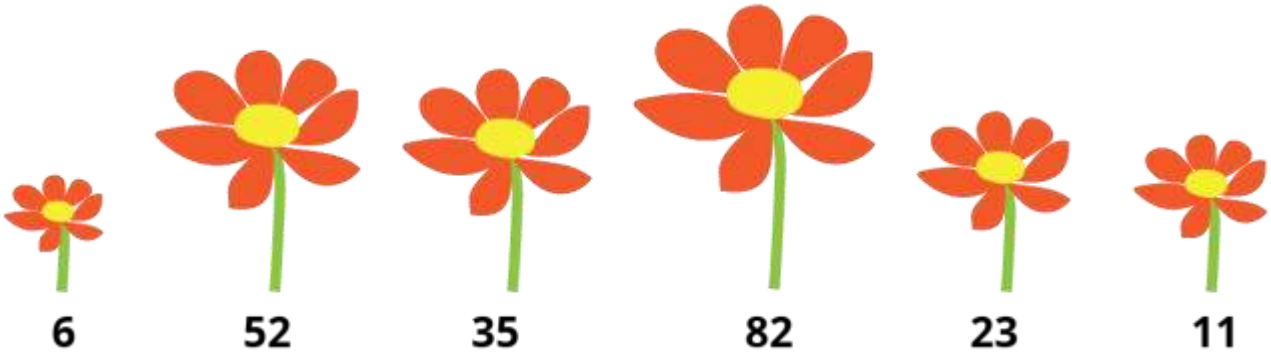


Nabiranje nektarja

4. – 7. razred

Čebela leta nabirat nektar. Na vsakem poletu gre le do enega cveta in nazaj do panja. Na njem pobere 10 mg nektarja (več ga ne more nositi) ali manj, če ga je na cvetu manj. Na isti cvet se lahko vrne večkrat.

Slika kaže količino nektarja na posameznem cvetu. Največ koliko ga lahko čebela nabere v dvajsetih poletih?



Rešitev

196 mg.

Čebela bo šla petkrat na drugi cvet, trikrat na tretjega, osemkrat na četrtega, dvakrat na petega in enkrat na šestega. Tako bo v 19 letih nabrala 190 mg nektarja. Na cvetovih bo ostalo, po vrsti, še 6, 2, 5, 2, 3, 1 mg nektarja. V dvajsetem letu bo šla tako po 6 mg nektarja na prvi cvet.

Računalniško ozadje

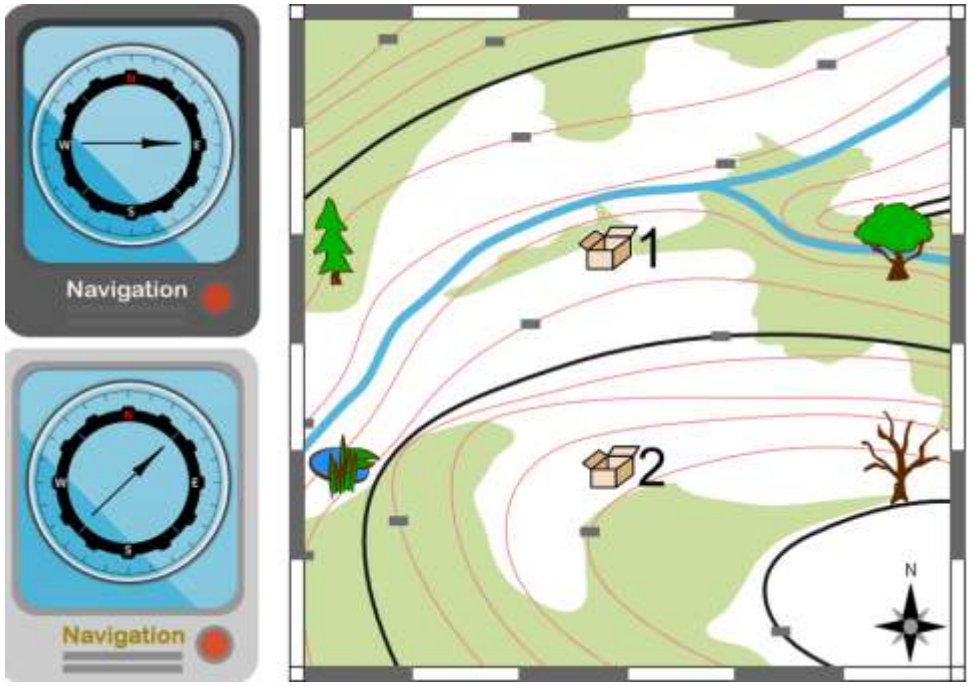
Lahko opišeš, na kakšen način se odloča čebela? Ni težko: čebela gre vedno na cvet, na katerem je največ nektarja.

Postopkom, ki delujejo na ta način, pravimo *požrešni postopki*. Navadno so preprosti in hitri, vendar pa z njimi ne dobimo vedno najboljše rešitve. Pri tej čebeli postopek vedno deluje, pri kakšnih drugačnih problemih pa bomo z njim pogosto dobili le *dovolj dobro*, čeprav ne nujno tudi *najboljšo* rešitev.

Iskanje zaklada

4. – 7. razred

Ana in Bob iščeta zaklada, skrita v malih škatlicah na mestih, ki ju kaže slika – Ana in Bob teh mest seveda ne poznata. Ana išče zaklad številka 1 in Bob zaklad številka 2.



Trenutno stojita oba na istem mestu. Na telefonih imata program, ki jima kaže smer, v kateri je zaklad. Sliki s telefonov sta prikazani na levi. Kateri telefon je čigav, ne vemo.

Kje stojita?

Rešitev

Pri mlaki. Pri zeleni smreki in zelenem drevesu ne moreta biti, saj nobena izmed puščic ne kaže navzdol. Tudi pri suhem drevesu ne moreta biti, saj nobena izmed puščic ne kaže levo.

Računalniško ozadje

Satelitska navigacija je vedno pomembnejša, saj jo uporabljamo v vedno več programih in za vedno več namenov. Če je bila prvotno zamišljena za to, da bi vedeli, kje smo in znali načrtovati pot do tja, kamor želimo, jo danes uporabljamo tudi za nadzor prometa, iskanje ukradenih reči, in v programih, ki nas, ko opazijo, da gremo pravkar mimo trgovine, spomnijo, da se moramo ustaviti in kupiti mleko.

Ideja za nalogo prihaja iz geocachinga. Če še ne veš, kaj je to, le pogooglaj. Morda te zagrabi.

Pogovor z robotom

4. – 7. razred

Končno so sestavili robota, kakršne vidimo v filmih: robota, s katerim se lahko pogovarjamo.

Ko ga kaj vprašamo, na zaslonu izpiše **?** in začne razmišljati. Ko odgovori, izpiše **!**. Na vsako vprašanje da samo en odgovor. Ker včasih tudi zelo dolgo razmišlja, mu lahko zastavljamo nova vprašanja, medtem ko še razmišlja o prejšnjih.

Človek: Koliko je 2 + 2?
Človek: Koliko je od Kranja do Škofje loke?
Robot: 4!
Človek: Si človek ali robot?
Robot: 12 kilometrov!

Po tem pogovoru bi bilo na robotovem zaslonu izpisano **??!?!**. Na zadnje vprašanje še ni odgovoril; o njem še razmišlja in s tem ni nič narobe.

Robot je pokvarjen, *če odgovarja na vprašanje, ki ga (še?) ni*. V trgovini vidimo štiri robote z naslednjimi izpisi. Samo eden od njih deluje pravilno. Kateri?

A. **!???!?!!?!**
B. **??!?!!!!**
C. **????????**
D. **?!?!!??!??**

Rešitev

C.

Robot A je začel govoriti, preden ga je kdo kaj vprašal. Robotu B smo postavili tri vprašanja, odgovarjal pa je petkrat. Robot D je najprej odgovoril na prvo vprašanje; nato je dobil drugo vprašanje in povedal nanj dva odgovora.

Robot C ni odgovoril še ničesar, vendar s tem ni nič narobe. Je pač bolj razmišljujoče vrste.

Računalniško ozadje

Ko popravljamo programe, pogosto kaj izpisujemo in potem poskušamo iz izpisa razbrati, za kakšno napako gre. Podobno si počel tule: na podlagi nekoliko zapletenega izpisa si poskušal odkriti, kateri program (no, robot) deluje pravilno in kateri ne.

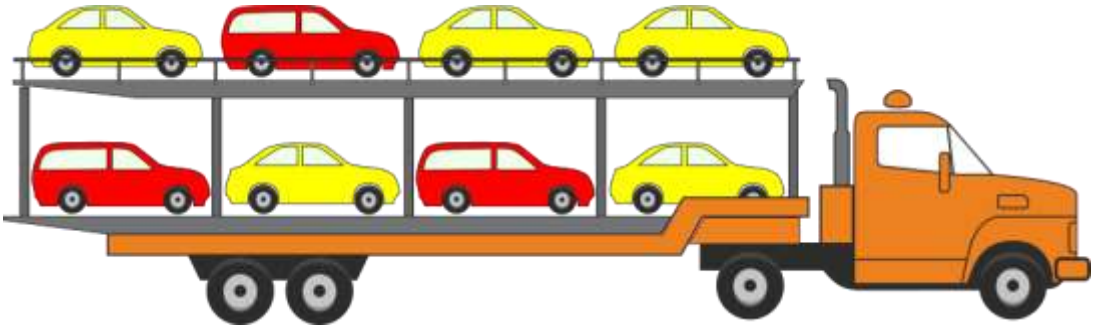
Prevoz avtomobilov

4. – 9. razred

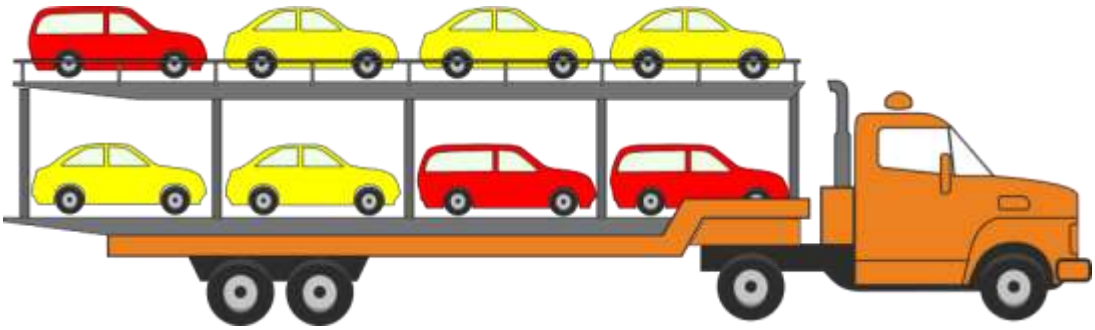
V tovarni avtomobilov so pravkar začeli proizvodnjo rdečih (temnejših) in rumenih (svetlejših) avtomobilov. Rdeči pride s proizvodne linije vsakih sedem minut, rumeni pa vsakih pet minut. Nakladajo jih na tovornjak – najprej zgoraj, nato spodaj.

Kako bo izgledal prvi tovornjak, ki ga bodo napolnili?

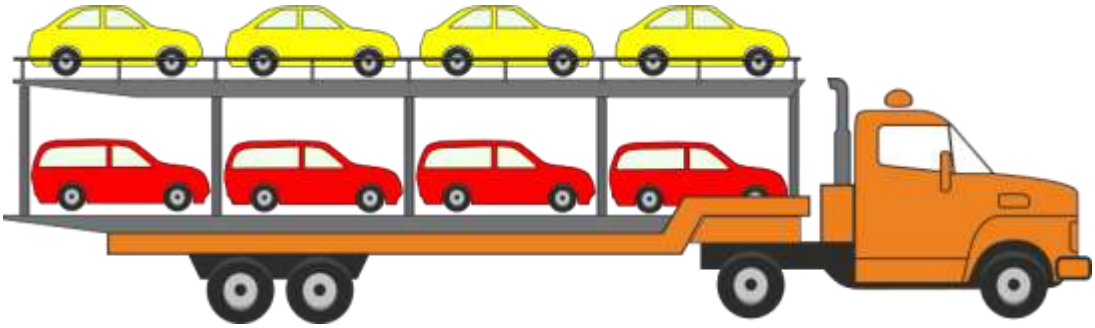
A.



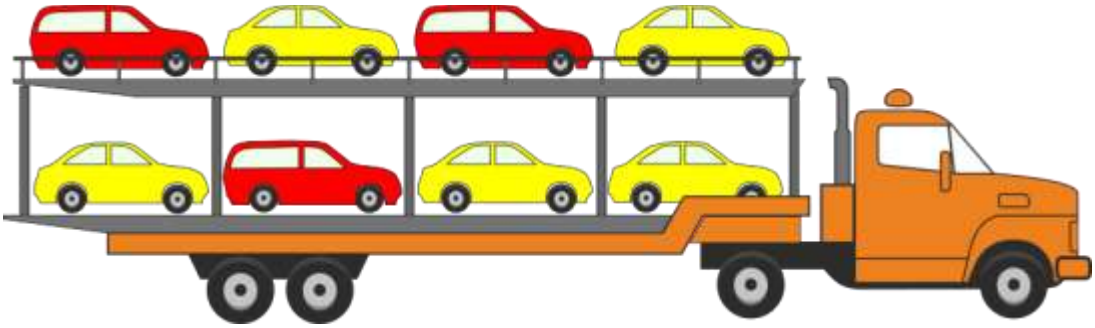
B.



C.



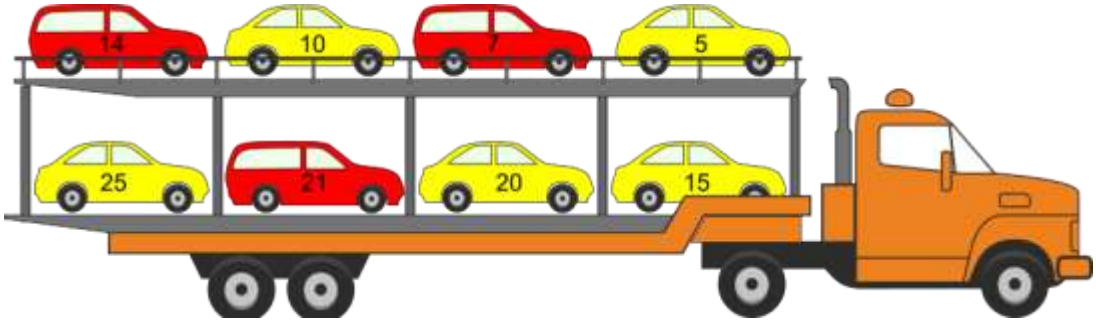
D.



Rešitev

Avtomobili bodo izdelani v takšnem vrstnem redu (številke na njih kažejo minuto, v kateri bodo narejeni).

Ko jih v tem vrstnem redu naložijo na tovornjak, bo izgledal tako, kot kaže odgovor D.



Računalniško ozadje

Ko si razmišljal o tej nalogi, si moral slediti rezultatom, ki jih bo dal nek postopek. Programerji pogosto počnejo prav to – le da je postopek, s katerim se ukvarjajo, program.

V sodobnih tovarnah je v resnici vedno manj delavcev, saj njihovo delo opravijo roboti. Načrtovalci tovarn morajo zato dobro razmisliti in uskladiti delovanje robotov, da pravočasno dobijo ustrezne dele, ki jih potrebujejo za nadaljnje sestavljanje končnega izdelka.

Polnjenje kadi

4. – 9. razred

Bober Samo se vsak večer kopa v polni kadi vode. Vodo mora sam znositi iz bližnjega jezera. V kad gre sedem malih veder. Poleg tega ima še veliko vedro, v katerega gre dvakrat toliko vode kot v malo.

Pot od jezera do hiše s polnim malim vedrom mu vzame štiri minute, s polnim velikim vedrom pa pet minut. Pot od hiše do jezera s praznim vedrom mu vedno vzame tri minute. Nosi lahko le eno vedro naenkrat.

Najmanj koliko minut potrebuje, da napolni kad?

Rešitev

31 minut.

Različnih možnosti je dovolj malo, da lahko pregledamo vse. Pot do jezera in nazaj z malim vedrom mu vzame $3 + 4 = 7$ minut, pod do jezera in nazaj z velikim pa $3 + 5 = 8$ minut.

- 7 malih veder: $7 \times 7 = 49$ minut
- 5 malih veder + 1 veliko: $5 \times 7 + 1 \times 8 = 43$ minut
- 3 mala vedra + 2 veliki: $3 \times 7 + 2 \times 8 = 37$ minut
- 1 malo vedro + 3 velika: $1 \times 7 + 3 \times 8 = 31$ minut

Nalogo lahko rešimo tudi preprosteje. Pot z enim velikim vedrom mu vzame 8 minut. Če hoče prinesti enako količino vode z malim vedrom, mora na pot dvakrat, kar mu vzame 14 minut. Torej se mu splača znositi čimveč vode z velikim vedrom.

Računalniško ozadje

Problem, ki ga rešujemo v tej nalogi, se učeno imenuje *problem polnjenja nahrbtnika*. Obstaja veliko različic problema. Tale je bila preprosta, nekaterim drugim, bolj zapletenim, pa niso kos niti najhitrejši računalniki.

Avtobusne proge

4. – 9. razred

Bober Dani živi v centru mesta. Avtobusne proge so oštevilčene skladno s spodnjimi pravili.

Prva števka pove, ali vozi avtobus v smeri centra, ven iz centra ali nič od tega:

- **1** proti centru
- **2** ven iz centra
- **3** ne proti centru ne ven iz centra

Druga števka pove smer vožnje:

- **1** proti severu
- **2** proti jugu
- **3** proti zahodu
- **4** proti vzhodu

Tretja števka pove hitrost:

- **0** počasnejši avtobus
- **1** hitrejši avtobus

Dani je preživel dan na plaži južno od mesta. Zdaj bi se rad čim hitreje vrnil domov. S katerim avtobusom se bo peljal?

Rešitev

Iti mora proti centru, proti severu in čim hitreje. To je avtobus 111.

Računalniško ozadje

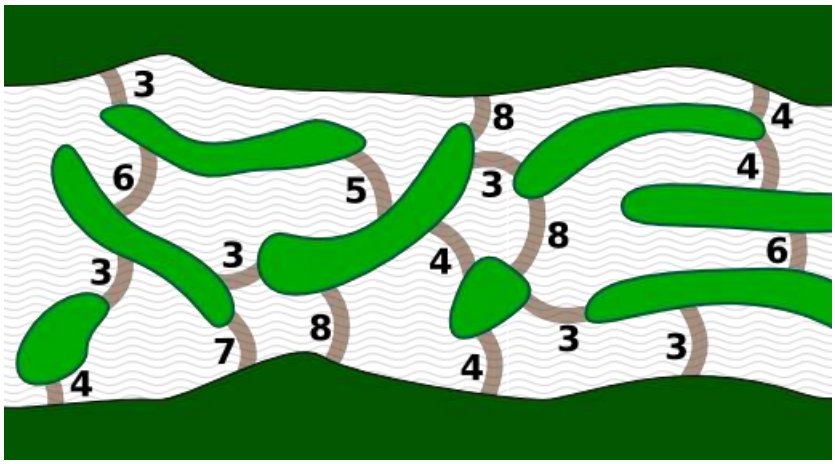
Tole oštevilčenje je res koristno, mar ni? Potnikom si tako ni potrebno zapomniti številk avtobusov, saj zadošča, da vedo pomen posameznih števk.

Tudi računalnikarji se morajo pogosto domisliti učinkovitih in uporabnih načinov zapisa podatkov.

Novi jez

4. – 9. razred

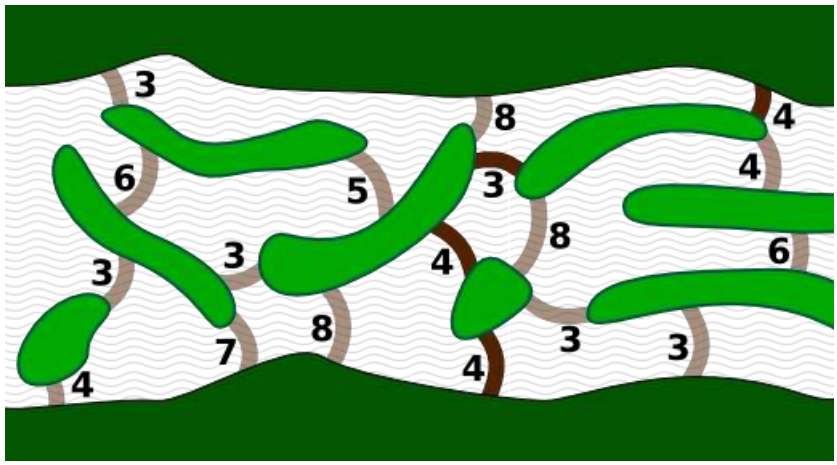
Bobri želijo zajeziti reko. Namesto da bi postavljali en sam jez, bodo gradili jezove med posameznimi otoki tako, da bo na koncu zajezena celotna reka.



Skica kaže mesta, na katerih bi lahko postavili jezove, in koliko hlodov bi potrebovali za vsakega od njih. Najmanj koliko hlodov bodo potrebovali za zajezitev reke?

Rešitev

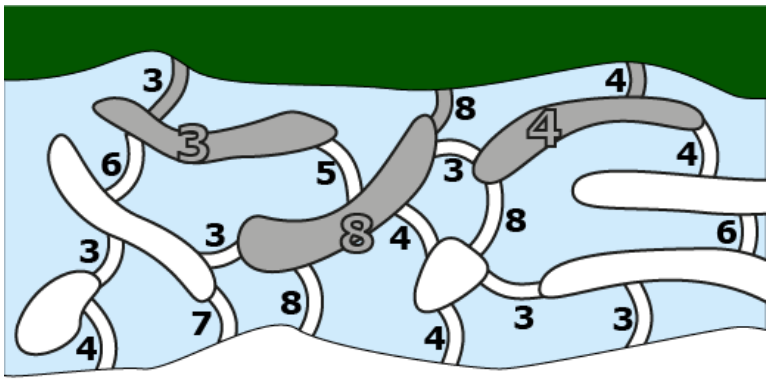
Potrebovali bodo 4 + 3 + 4 + 4 = 15 hlodov.



Bobri pogosto iščejo najkrajšo pot med dvema mestoma, hišama, prijateljema... Tudi iskanje jezu s čim manj hlodi je enako iskanju najkrajše poti med enim in drugim bregom, pri čemer ne upoštevamo hoje po otokih, temveč štejemo le hlode.

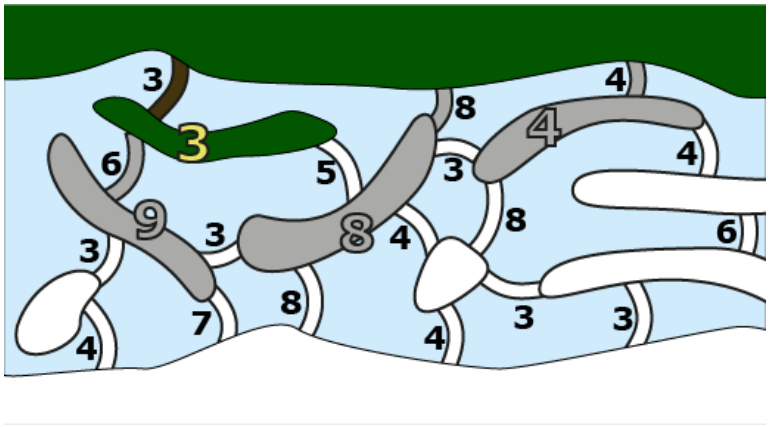
Nalogo si verjetno reševal s poskušanjem. Kako pa takšno nalogo rešimo sistematično?

V začetku vemo, kako doseči zgornje tri otoke s 3, 8 ali 4 hlodi. Vendar ne smemo hiteti: zagotovo lahko trdimo le, da potrebujemo vsaj tri hlode do prvega. Do drugih dveh bo mogoče možno priti tudi z manj hlodi. (Pravzaprav celo vidimo, da lahko do osrednjega otoka pridemo s 7 hlodi. Do levega zgornjega pa z manj kot tremi ne bo šlo, ker z manj kot tremi ne moremo nikamor.)



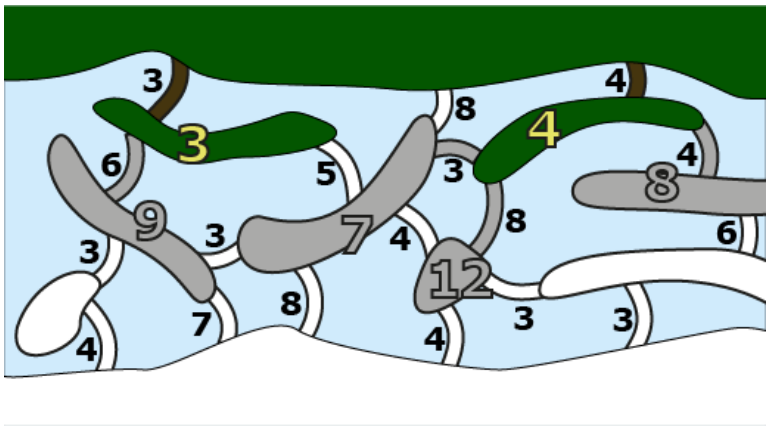
Vemo torej, da lahko dosežemo levi gornji otok s tremi hlodi. Z njega lahko dosežemo otok pod njim s skupno devetimi hlodi (tri do prvega in še šest naprej). Prav tako lahko pridemo do srednjega otoka z osmimi (5 + 3), a to že vemo.

Te otoke smo obarvali sivo. Med njimi bomo spet izbrali tistega, do katerega pridemo z najmanj hlodi, namreč desnega. Zakaj tako? Vemo, da do njega ne bomo mogli priti z manj kot štirimi hlodi. Kakršnakoli pot prek kakega drugega otoka, bo lahko samo daljša.



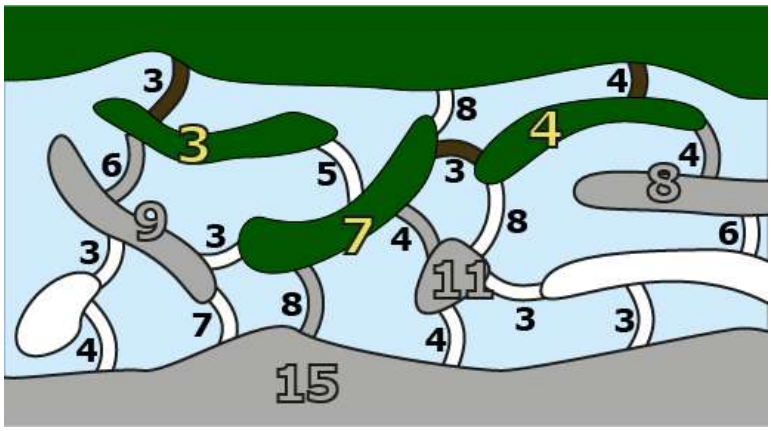
Za ostale sive otoke to ne velja: do njih bomo morda prišli po krajši poti. To pravzaprav vidimo takoj. Z gornjega desnega lahko dosežemo osrednji otok s skupno 4 + 3 = 7 hlodi! Popravimo osmico v sedmico. Poleg tega lahko z njega dosežemo dva otoka s skupno 4 + 8 = 12 in 4 + 4 = 8 hlodi.

Ozremo se po trenutno dosegljivih otokih. Do osrednjega gotovo ne bomo mogli z manj kot 7 hlodi, saj bi bile vse poti prek drugih otokov lahko kvečjemu daljše.



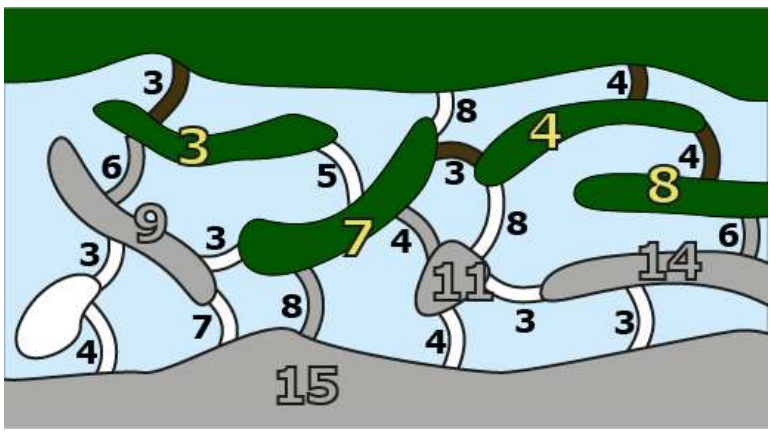
Osrednji otok označimo kot dosegljiv s sedmimi hlodi. Z njega bi lahko dosegli levega z desetimi, vendar se to ne splača, saj znamo do njega s 3 + 6 hlodi z levega gornjega. Pač pa zdaj vidimo, da lahko pridemo do otoka pod osrednjim s 7 + 4 = 11 hlodi, torej zamenjamo 12 z 11.

Z osrednjega otoka bi že lahko prišli tudi na drugi breg s skupno 7 + 8 = 15 hlodi. Vendar počakajmo, morda se pokaže tudi kaj boljšega.



Naslednji otok, za katerega zagotovo vemo, koliko hlodov potrebujemo do njega, je desni. Do njega potrebujemo 4 + 4 = 8 hlodov.

Z njega lahko pridemo na desni spodnji otok z 8 + 6 = 14 hlodi.

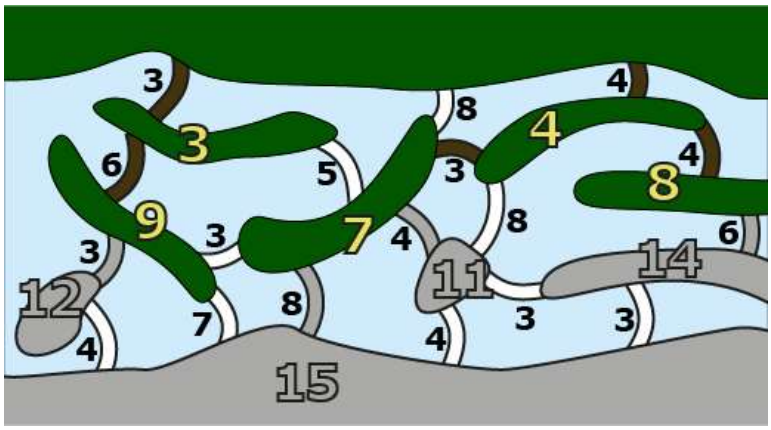


Zdaj dodamo levi otok.

Z njega lahko gremo navzdol; pot z gornjega brega do levega spodnjega otoka bi zahtevala 9 + 3 = 12 hlodov.

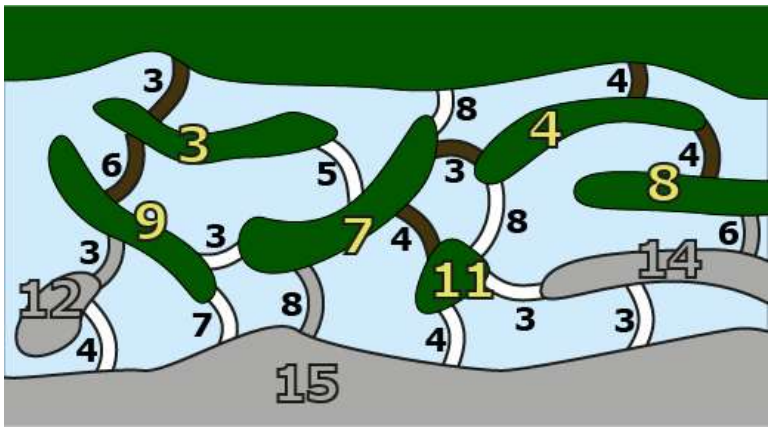
S tega otoka lahko gremo tudi na spodnji breg s skupno 9 + 7 = 16 hlodi, vendar se to ne splača, saj že poznamo krajšo pot.

(V tem trenutku je rešitev sicer že očitna, vendar sledimo postopku do konca.)



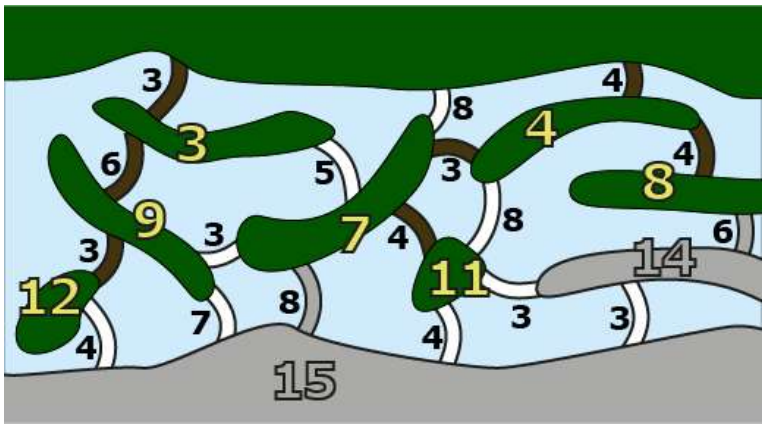
Dodamo otok, do katerega pridemo s skupno 11 hlodi. Z njega dosežemo spodnji breg z 11 + 4 hlodi. Ta rešitev je enako dobra kot pot z osrednjega otoka.

Prav tako lahko s tega otoka pridemo do otoka na desni z 11 + 3 = 14 hlodi, vendar že vemo, da lahko ta otok z enaki številom hlodov dosežemo od zgoraj.



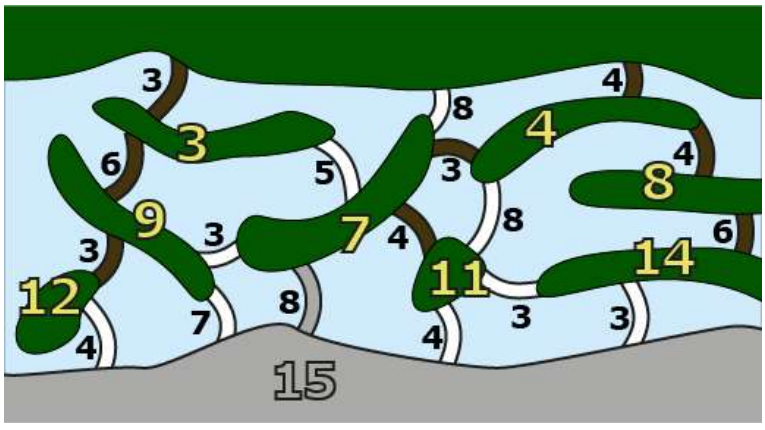
Med otoki označenimi s sivo spet izberemo otok, do katerega pridemo z najmanj hlodi. To je levi spodnji otok.

Z njega pridemo na spodnji breg z 12 + 4 = 16 hlodi, vendar je to slabše od trenutno najboljše rešitve.

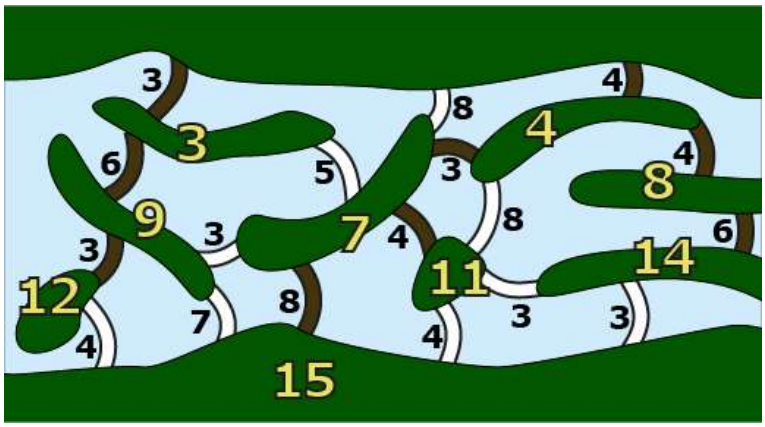


Zdaj je na vrsti desni spodnji otok.

Z njega bi dosegli nasprotni breg z 14 + 3 = 17 hlodi.



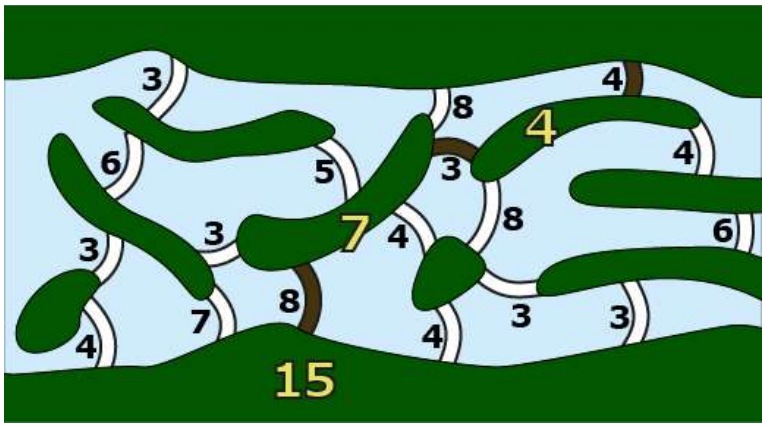
Končno dodamo še nasprotni breg, ki ga lahko s 15 hlodi dosežemo z osrednjega otoka (7 + 8) ali otoka desno pod njim (11 + 4).



Zdaj odstranimo nepotrebne jezove in ohranimo le te, ki vodijo na nasprotni breg.

Hm, ta rešitev ni enaka tej, ki smo jo objavili zgoraj!

Res ni, vendar potrebuje enako hlodov. S postopkom, ki smo ga opisali, bomo vedno našli najboljšo rešitev. Kadar obstaja več enako dobrih, bomo pač našli le eno od njih.



Računalniško ozadje

Računalnikarji so leni in zviti hkrati, kar je odlična kombinacija. Naučijo se vreče trikov in kadar naletijo na problem, uporabijo najprimernejšega med njimi. V tem primeru bi opazili, da je iskanje najcenejšega jezu enako iskanju najkrajše poti. Na ta način spremenijo en problem (gradnjo jezu) v drugega (iskanje poti), ki ga znajo dobro rešiti. Eden najpomembnejših računalnikarjev 20. stoletja, E. W. Dijkstra, se je namreč že pred 60 leti domislil postopka, ki ga opisujemo zgoraj in po katerem še danes delujejo postopki za iskanje poti na spletu, v telefonih, navigacijskih napravah...

Se ti zdi naloga znana? Pred dvema letoma so bobri gradili mostove do otokov, ki so bili videti natančno tako kot tile. Vendar je bila naloga drugačna. Medtem ko tu gradimo sistem jezov, pri katerem nam je vseeno, če vanj niso vključeni vsi otoki (to bi bilo pravzaprav potratno – tule smo z jezovi povezali le dva otoka, saj to zadošča), je bilo v tisti nalogi potrebno postaviti tak sistem mostov, da bodo dosegljivi vsi otoki. To je popolnoma drugačen problem – a računalnikarjev to ne moti, le drug trik potegnejo iz svoje velike vreče.

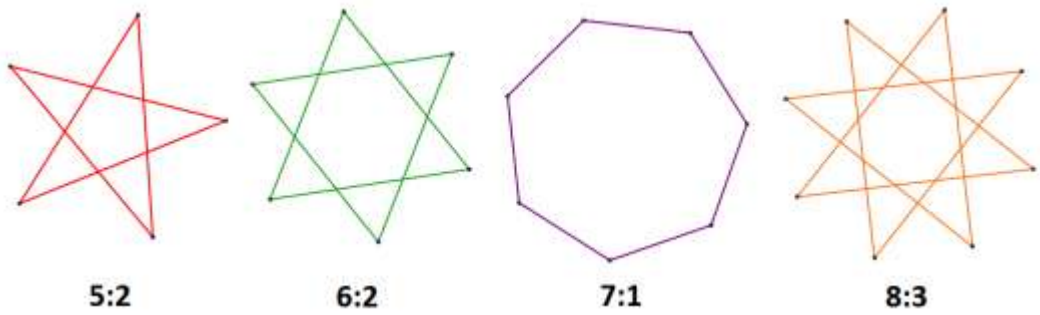
Opisovanje zvezd

6. – 9. razred

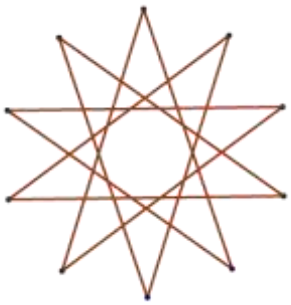
Stela si je izmislila način za opisovanje zvezd s parom številk:

- prva številka pove število krakov in
- druga številka pove, kako so kraki povezani med seboj.

Nekaj primerov kažejo slike.



Kako bi Stela označila tole zvezdo?



Rešitev

10:4.

Zvezda ima deset krakov in vsak krak je povezan s četrtim naslednjim krakom.

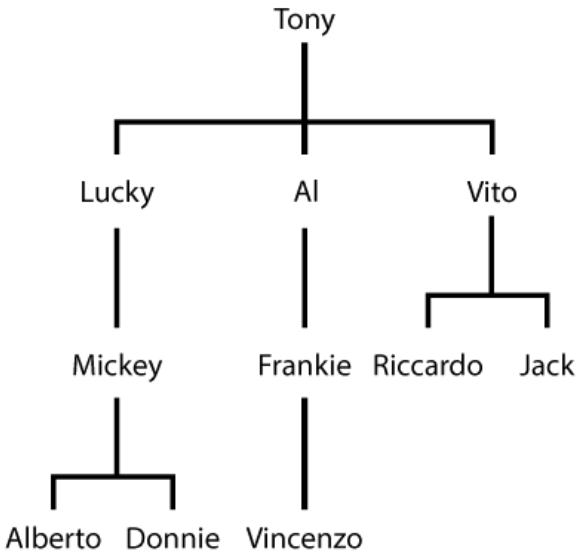
Računalniško ozadje

V računalništvu si je potrebno stalno izmišljati praktične načine za opisovanje različnih reči.

Šefi

6. – 9. razred

Tony je glavni šef določenega gradbenega podjetja. Trije podšefi – po vrsti od najbolj spoštovanega – so Lucky, Al in Vito. Lucky je šef Mickeyu in Micky je šef Albertu in Donnieju. Al je šef Frankieju in Frankie Vincenzu. Vito ima Riccarda in Jacka. Celotno shemo kaže slika.



Če Tony kam izgine, ga na mestu glavnega šefa zamenja Lucky. Če manjka tudi ta, ga zamenja Al … in tako naprej. Mickey ima prednost pred Riccardom. Frankie ima prednost pred Albertom.

Koliko zaposlenih mora izginiti, da bo glavni šef Riccardo?

Rešitev

Šest: Tony, Lucky, Al, Vito, Mickey in Frankie.

Računalniško ozadje

Temu, kar kaže slika, računalnikarji rečejo drevo (in, da, rišejo ga obrnjenega na glavo). V drevesa pogosto shranjujemo različne podatke. Ko jih pregledujemo, preverjamo, izpisujemo … gremo navadno prek njih na enega od dveh načinov, *v globino* ali *v širino*. Pri pregledovanju v globino bi bil vrstni red Tony, Lucky, Mickey, Alberto, Donnie, Al, Frankie, Vincenzo, Vito, Riccardo, Jack. Tile mafijci pa dedujejo položaje glavnega šefa na drugi način, *v širino*, torej Tony, Lucky, Al, Vito, Mickey, Franki, Riccardo, Jack, Alberto, Donnie in Vincenzo.

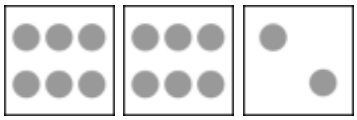
Kam gremo

6. – 9. razred

Da se bobri ne bi kregali, kam se gredo igrat po šoli, trikrat vržejo kocko in se odločijo takole

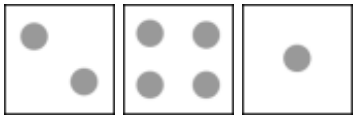
ČE je prvi met večji od drugega
POTEM gredo v gozd
SICER **ČE** je tretji met manjši od prvega
POTEM gredo k reki
SICER gredo na igrišče

V katerem od naslednjih štirih primerov bodo šli na igrišče?

A. 

B. 

C. 

D. 

Rešitev

C.

A. in D. jih peljeta k reki; prvi met ni večji od drugega, vendar je tretji manjši od prvega.

B. jih pelje v gozd, saj je prvi met večji od drugega.

Računalniško ozadje

Bobri izvajajo preprost program, sestavljen iz pogojnih stavkov ČE-POTEM-SICER (v angleščini IF-THEN—ELSE).

Pospravljanje barvic

6. – 9. razred

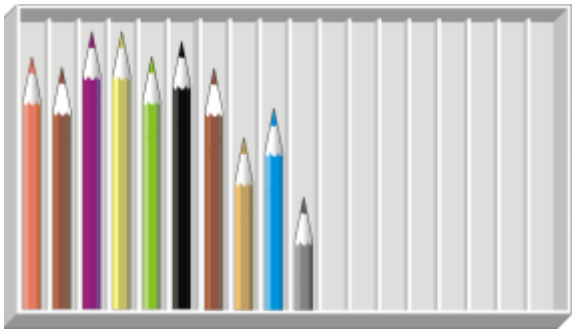
Mali bober Božidar bo zložil barvice v mamino in očetovo škatlo.

- Jemal jih bo z leve proti desni.
- Odlagal jih bo v mamino škatlo, prav tako z leve proti desni.
- Prva barvica gre v mamino škatlo.
- Vsako barvico primerja z najbolj desno barvico v mamini škatli. Če je manjša od nje, jo doda v mamino škatlo. Sicer jo da v očetovo škatlo.

Kako bo po tem videti očetova škatla?

A.

B.



C.

D.

Rešitev

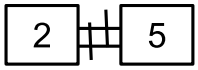
Prvih pet barvic gre v mamino škatlo. Oranžna je prva v očetovi, torej odgovora C in D ne moreta biti pravilna. Naslednja barvica, ki gre v mamino škatlo, je kratka svetlorjava barvica – ki pa jo najdemo v odgovoru B. Pravilna rešitev je torej A.

Računalniško ozadje

Naloga opisuje postopek, program, ki ga izvaja Božidar. Da jo rešimo, ga moramo izvesti tudi mi in pogledati, kakšen rezultat da. Hm, mar res? Pravzaprav ne. Zadoščalo je, da smo razmislili, kakšen rezultat da postopek ter poiskali škatlo, ki ga kaže – oziroma izločili škatle, ki jih ta postopek ne more dati.

Risanje

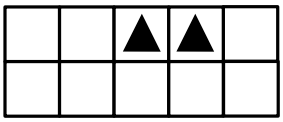
6. – 9. razred

Ukaz 

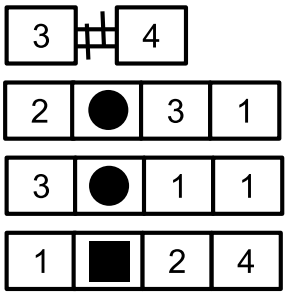
nariše mrežo z dvema vrsticama in petimi stolpci.

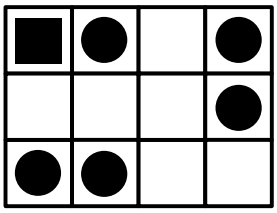
Ukaz 

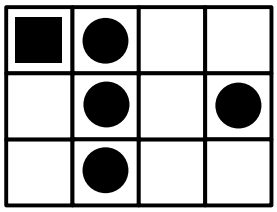
nariše dva trikotnika. Prvi je v prvi vrstici, v tretjem stolpcu. Drugi je desno od njega.

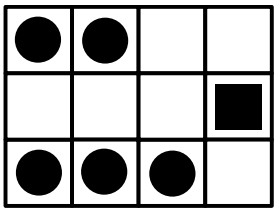
Oba ukaza zapored torej narišeta 

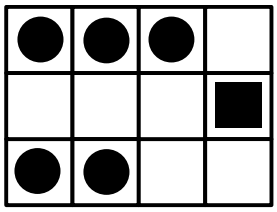
Kakšno sliko nariše spodnje zaporedje ukazov?



A. 

B. 

C. 

D. 

Rešitev

Dva kroga na začetku tretje vrstice in trije na začetku prve – to je lahko le odgovor D.

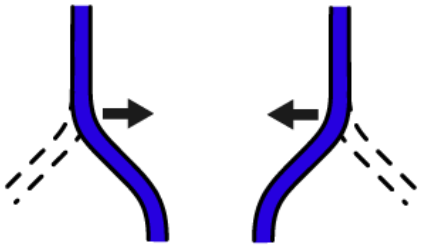
Računalniško ozadje

Kar opisuje naloga, je preprost programski jezik.

Namakalni sistem

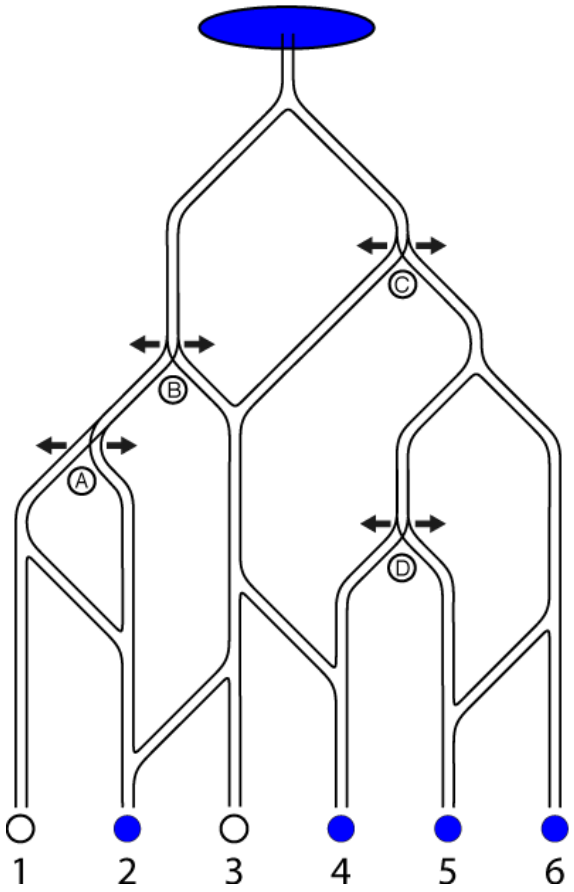
državno, 6. – 7. razred

Bobri so zgradili eleganten namakalni sistem za svoja polja. Vzdolž vodovodnih kanalov so zgradili vodna vrata, označena z A, B, C in D, s katerimi usmerjajo vodo na levo (⬅) ali na desno (➡).



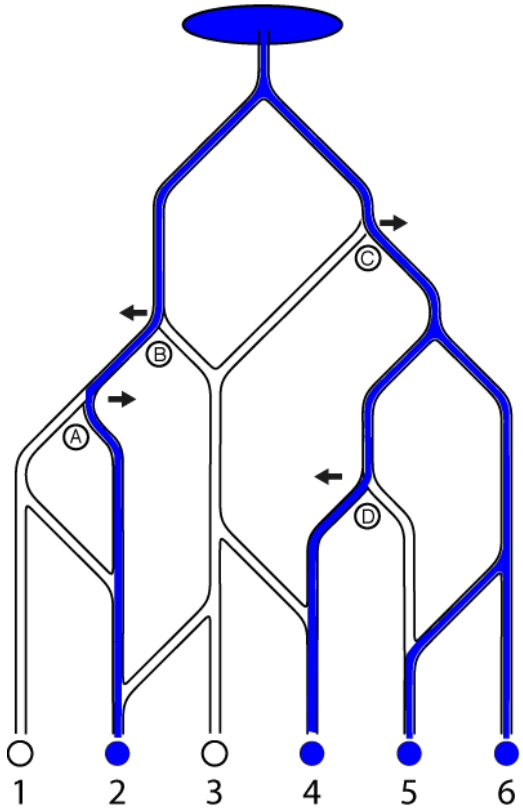
Voda vedno teče le navzdol in nikoli ne zavija nazaj navzgor.

Kakšna je pravilna nastavitev vodnih vrat, da bodo namočena le drugo, četrto, peto in šesto polje?



Rešitev

Vodo je potrebno usmeriti, kot kaže slika na desni. Kako bi do rešitve prišli sistematično? In kako preverili, da morda ne obstaja še katera druga rešitev?



1 se ne sme namakati:
A desno
ali B desno

2 se mora namakati:
B levo
ali B desno
ali C levo

3 se ne sme namakati:
B levo in C desno

4 se mora namakati:
B desno
ali C levo
ali C desno in D levo

5 se mora namakati:
C desno

6 se mora namakati:
C desno

Zaradi polja 3 vemo, da mora biti B obrnjen levo in C desno. Prečrtajmo vse, kar ni skladno s tem.

1 se ne sme namakati:
A desno
~~**ali** B desno~~

2 se mora namakati:
B levo
~~**ali** B desno~~
~~**ali** C levo~~

3 se ne sme namakati:
B levo in C desno

4 se mora namakati
~~B desno~~
~~**ali** C levo~~
ali C desno in D levo

5 se mora namakati
C desno

6 se mora namakati
C desno

In zdaj ni več nobene izbire. Rešitev, ki smo jo našli, je edina rešitev.

Obstaja še en sistematičen način reševanja – in iskanja vseh možnih rešitev: zapišemo in izračunamo vse možnosti. Na desni strani tabele nato poiščemo vse vrstice z želenim vzorcem, ○●○●●●. Ta je natančno ena.

Ta način reševanja je očitno počasnejši.

A	B	C	D	1	2	3	4	5	6
L	L	L	L	●	●	●	●	○	○
L	L	L	D	●	●	●	●	○	○
L	L	D	L	●	●	○	●	●	●
L	L	D	D	●	●	○	○	●	●
L	D	L	L	○	●	●	●	○	○
L	D	L	D	○	●	●	●	○	○
L	D	D	L	○	●	●	●	●	●
L	D	D	D	○	●	●	●	●	●
D	L	L	L	○	●	●	●	○	○
D	L	L	D	○	●	●	●	○	○
D	L	D	L	○	●	○	●	●	●
D	L	D	D	○	●	○	○	●	●
D	D	L	L	○	●	●	●	○	○
D	D	L	D	○	●	●	●	○	○
D	D	D	L	○	●	●	●	●	●
D	D	D	D	○	●	●	●	●	●

Računalniško ozadje

Matematiki bi v narisani sliki prepoznali nekaj, čemur rečejo *usmerjeni aciklični graf*. *Graf* je učeno ime za množico križišč (točk) in povezav med njimi. Ker je *usmerjen*, je določeno, v katero smer smemo potovati po povezavi, in ker je *acikličen*, točke niso povezane na tak način, da bi lahko v neskončno krožili prek enih in istih točk.

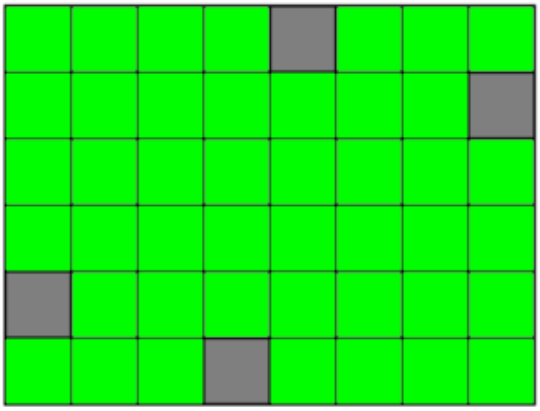
Računalnikarji bi v opazovanju toka vode videli *simulacijo*. Ko razmišljamo o tem, kam bi pritekla voda ob takšni in takšni nastavitvi ventilov, izvajamo “simulacijo toka vode”.

Način, na katerega smo najprej rešili nalogo, pa spada v področje logike. Teh in takšnih nalog smo vajeni z drugih tekmovanj. Lahko bi srečali šest oseb, ki vedno govorijo resnico; prva reče “A je levo ali pa je B desno”, druga zatrdi “B je levo in A desno, ali pa je B desno ali pa je C levo” in tako naprej. Sestavljanje njihovih odgovorov je očitno enako reševanju te naloge.

Pri reševanju smo imeli srečo, da nam je eno od polj – namreč tretje – tako pošteno pomagalo. Takšne naloge je mogoče postaviti tudi tako, da jih ne moremo reševati hitreje, kot da poskusimo vse možne kombinacije, tako kot smo naredili v drugi rešitvi, s tabelo. Kot izkušeni bobri dobro vedo, je teh vrstic 16. In vsak dodaten ventil podvoji število vrstic – ter s tem število različnih kombinacij. Problem, ki si ga reševal, je zato lahko tudi zelo težak, saj število kombinacij, ki jih je potrebno preveriti, zelo hitro narašča s številom ventilov.

Širjenje skrivnosti

državno, 6. – 7. razred



Na vsakem zelenem kvadratku na sliki živi bober. Sivi kvadrati so nezasedeni.

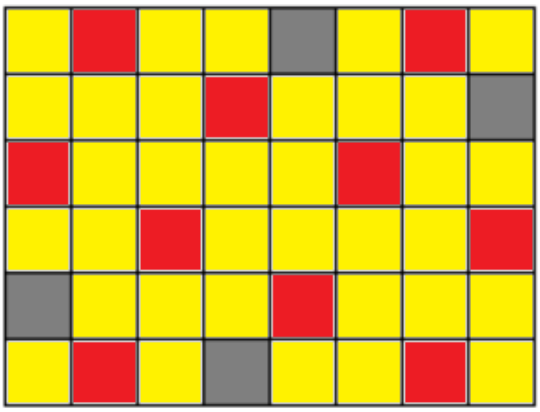
Bobri ne morejo zadržati skrivnosti zase. Vse, kar danes poveš enemu bobru, bodo jutri vedeli vsi njegovi sosedi. (Vsak bober ima največ štiri sosede; diagonalnih polj ne štejemo za sosednja.)

Imaš novico, za katero želiš, da bi jo jutri vedeli vsi bobri. Da ne boš porabil preveč časa z govorjenjem, je ne boš povedal vsem, temveč le določenim. Izbral jih boš tako, da bodo takrat, ko jo le-ti povejo sosedom, novico poznali vsi.

Najmanj koliko bobrom moraš povedati novico?

Rešitev

Novico povemo bobrom, katerih domovi so na sliki označeni z rdečo barvo.



Računalniško ozadje

Naloga je primer problema, ki mu rečemo “problem pokritja”. Se ti je zdela težka? Saj tudi je! Za večino problemov pokritij ne poznamo učinkovitih postopkov reševanja. Medtem ko bi nalogo, kakršno si reševal tule, računalnik rešil v trenutku, si lahko brez težav izmislimo takšne, ki jim ni kos noben računalnik. Večina računalnikarjev celo verjame – čeprav tega ni še nihče uspel dokazati – da si za te probleme nikoli ne bomo uspeli izmisliti hitrih postopkov reševanja.

Avtomobilske registracije

državno, 6. – 9. razred

Številke registrskih tablic v Bobroviji so sestavljene iz šestih števk. Mehanik Tim jih zlaga na police, oštevilčene od 0 do 18. Na vsaki polici je lahko le ena registrska tablica. Pospravlja jih po sistemu, ki mu omogoča, da hitro poišče iskano tablico:

- Sešteje prvi števki in postavi tablico na to polico – če je še prosta. Tako je registracijo 357900 (Ford Mondeo) postavil na polico 8, ker je 3 + 5 = 8.
- Če je polica, na katero bi moral postaviti tablico, že zasedena, postavi registracijo na prvo prosto polico z večjo številko.

Na police mora poleg 357900 zložiti še naslednje registracije:

- 115678 VW Golf
- 011234 Toyota Camry
- 561478 VW Jetta
- 652147 Ford Fiesta
- 093366 Kia Rio
- 209874 Ford Focus
- 113598 Toyota Corolla

Zlagal jih bo natančno v tem, napisanem vrstnem redu. Kako bodo videti police, ko konča delo?

Rešitev

Police bodo videti takole:

0	
1	011234 Toyota Camry
2	115678 VW Golf
3	209874 Ford Focus
4	113598 Toyota Corolla
5	
6	
7	
8	357900 Ford Mondeo
9	093366 Kia Rio

10	
11	561478 VW Jetta
12	652147 Ford Fiesta
13	
14	
15	
16	
17	
18	

Računalniško ozadje

Računalniki shranjujejo velike količine podatkov, ki jih morajo biti zmožni tudi hitro poiskati. Takšnim tabelam pravimo *zgoščevalne tabele*: vse, kar shranjujemo v njih, je na mestu, ki ga določi *zgoščevalna funkcija* – v našem primeru je to predpis "seštej prvi števki". Zgoščevalna funkcija nam torej pove, kam spraviti neko reč in kje jo potem iskati.

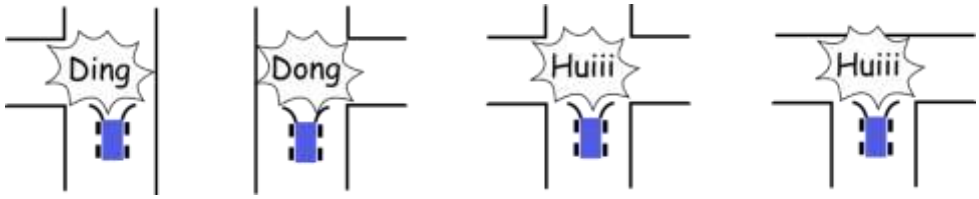
Ime je dobila po tem, ker "zgošča" – v našem primeru predpisuje, pod katerim številom med 0 in 18 shraniti različna števila med 000000 in 999999. Zaradi "zgoščanja" prihaja do "trkov", ko bi morali postaviti dve stvari na isto mesto. V tem primeru moramo z drugim pravilom poiskati drugo prosto mesto. V našem primeru je tudi drugo pravilo preprosto: izberi naslednje prosto mesto.

V pravih programih uporabljamo bolj zapletene zgoščevalne funkcije in bolj premetena pravila za določanje naslednjega elementa. Bistvo pa je enako, kot si ga spoznal v tej nalogi.

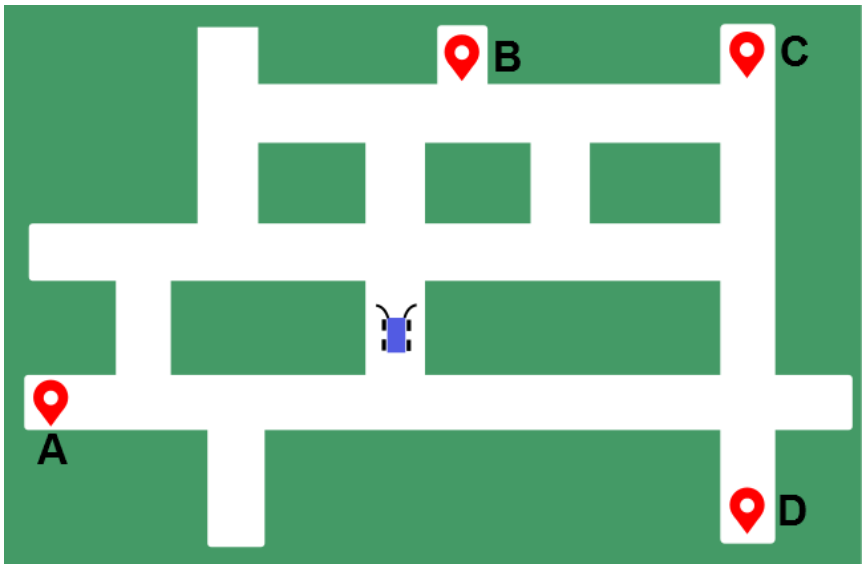
Slepi potnik

državno, 6. – 9. razred

Avtomobil za slepe ima senzorje, ki zaznajo križišča. Vrsto križišča – je mogoče zaviti levo, desno ali na obe strani – sporoči z zvokom.



Ko ga je Ana nekega dne vozila, je bilo slišati zvoke: Huiii Ding Huiii Dong. Vožnjo je začel, kjer je narisano, obrnjen navzgor. Kje jo je končal?



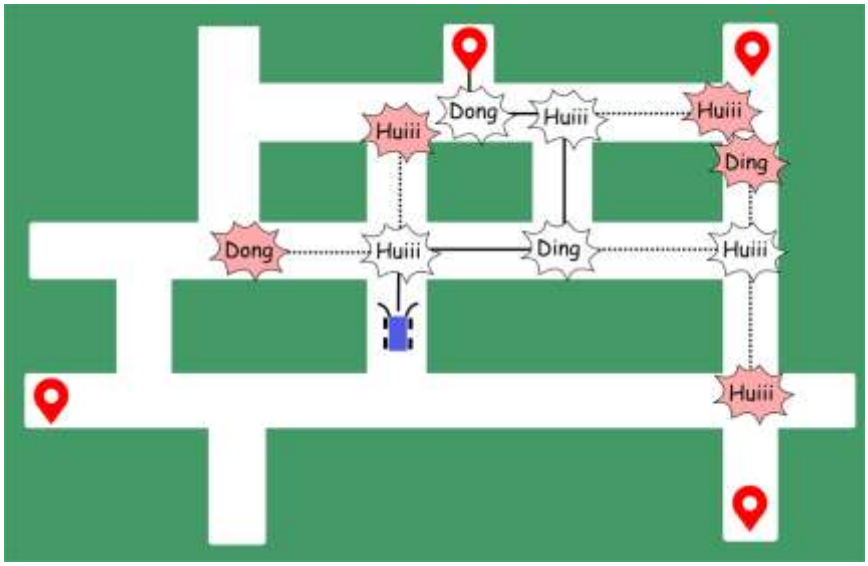
Rešitev

Da je bil prvi zvok Huiii, je očitno. Ker je bil drugi Ding, lahko sklepamo, da je na prvem križišču zavil desno (če bi levo, bi bil naslednji zvok Dong). Na naslednjem križišču je zavil levo, saj sta lahko le tako naslednja zvoka Huiii in Dong. Končal je na mestu, označenem s črko B.

Računalniško ozadje

Programerji pogosto naredijo kakšno napako. Včasih jo iščejo tako, da v program vstavijo različne izpise. Iz njih potem sklepajo, kaj je počel program in kaj je šlo narobe.

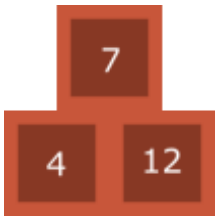
Podobno detektivsko zgodbo si reševal tule, saj zvoki niso usmerjali avtomobila, temveč si moral iz njih sklepati, kje je vozil.



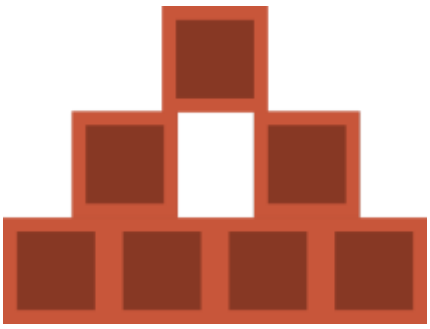
Zlaganje škatel

državno, 6. – 9. razred

Imamo oštevilčene škatle. Zlagamo jih v piramide: vsaka škatla stoji na dveh drugih tako, da ima škatla na levi manjšo številko od zgornje, škatla na desni pa večjo. Piramido iz treh škatel s številkami 4, 7 in 12 kaže slika na desni.

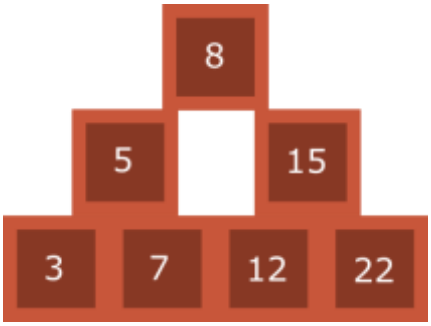


Kako pa bi bila videti piramida iz škatel s številkami 3, 5, 7, 8, 12, 15, 22? Katera številka bo na katerem mestu?



Rešitev

Da zadostimo pravilu, moramo številke preprosto napisati z leve proti desni.



Računalniško ozadje

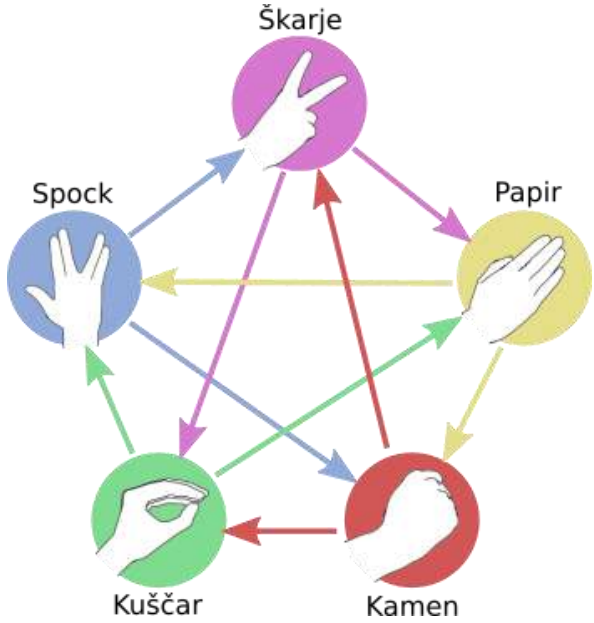
Predstavljajmo si takšno piramido iz 1023 škatel. Kako bi našli škatlo z neko določeno številko? Začeli bi pri vrhu. Če je iskana številka manjša od številke na vrhu, gremo levo, sicer desno. To ponavljamo, dokler ne naletimo na iskano številko. Koliko korakov bi zahtevalo to iskanje? Največ toliko, kolikor je visoka piramida. Zgoraj je ena škatla, pod njo dve, spodaj štiri, osem, šestnajst... 1 + 2 + 4 + 8 + 16 + 32 + 64 + 128 + 256 + 512 = 1023; sešteti smo morali deset nadstropij. Če bi podvojili število škatel, bi za to potrebovali le eno nadstropje več.

Ker je iskanje po takšnih “piramidah” tako učinkovito, jih pogosto uporabljamo tudi pri shranjevanju podatkov v računalniku, le da jih tam ne rečemo “piramida” temveč “urejeno drevo”.

Papir, škarje, kamen

6. – 9. razred, 1. – 4. letnik

Papir, škarje, kamen je igra za dva igralca. Igralca v vsaki potezi izbereta eno orožje – papir, škarje ali kamen – in ga istočasno pokažeta z roko. Kamen skrha škarje, škarje prerežejo papir in papir ovije kamen. Zmagovalec dobi točko. Če oba igralca pokažeta isto orožje, točke ne dobi nihče.



Sheldon in Rajesh sta v igro dodala kuščarja in Spocka. Kdo premaga koga, kaže skica. Tako, na primer, kamen premaga (zmečka) kuščarja.

V neki igri je Sheldon najprej pokazal Spocka, nato kuščarja, Spocka, kamen in Spocka. Rajesh je pokazal Spocka, papir in trikrat škarje. Kakšen je končni rezultat?

Sheldon

Spock kuščar Spock kamen Spock

Rajesh

Spock papir škarje škarje škarje

Rešitev

4 : 0 za Sheldona. V prvem krogu je rezultat neodločen. Nato kuščar premaga papir, Spock škarje, kamen škarje in Spock škarje.

Računalniško ozadje

Pravila igre – kdo premaga koga - so prikazana v obliki, ki jo v računalništvu (in matematiki) pogosto uporabljamo. Imenujemo jo *graf*.

Razkrivanje gesel

državno, 6. – 9. razred

Spletna mesta ne shranjujejo naših gesel, temveč iz njih izračunajo neko številko in shranijo le-to. Ko se poskušamo drugič prijaviti, moramo vnesti geslo. Spletni strežnik na novo izračuna številko in če se ujema s shranjeno, je geslo pravilno.

Neko podjetje skriva gesla tako, da vsako črko zamenja z eno števko, skladno s spodnjo tabelo.

A	B	C	Č	D	E	F	G	H	I	J	K	L	M	N	O	P	R	S	Š	T	U	V	Z	Ž
1	0	0	2	0	0	0	0	2	0	2	1	0	2	2	1	2	1	2	2	1	1	0	1	0

Nekdo si je za geslo izbral besedo PLAVANJE. Spletna stran si je to zapomnila kot 20101220 (P=2, L=0, A=1, V=0, A=1, N=2, J=2, E=0).

Lahko iz tako shranjene številke uganemo geslo? Morda. Vdrli smo v spletno mesto in ugotovili, da je geslo, ki ga uporablja direktor podjetja, shranjeno kot 21021120220. Vemo, da rad čolnari in rešuje uganke, zato si je gotovo izbral eno od naslednjih gesel: SKRIVNOSTNO, PREMETENOST, PRISTANIŠČE, ČOLNARJENJE. Lahko poveš, katerega? Ali pa tega ne moremo zagotovo ugotoviti?

Rešitev

```
SKRIVNOSTNO = 21100212121
PREMETENOST = 21020102121
PRISTANIŠČE = 21021120220
ČOLNARJENJE = 21021120220
```

Kaj je njegovo geslo, ne moremo izvedeti: lahko je PRISTANIŠČE ali ČOLNARJENJE.

Mar to pomeni, da ne moremo izvedeti njegovega gesla? Ah, seveda ga lahko: pač poskusimo oba.

Računalniško ozadje

Spletne strani v resnici ne poznajo naših gesel, temveč shranijo nekaj, kar izračunajo iz njih, tako kot v tej nalogi. To je potrebno zato, da nepridipravi, ki bi vdrli na stran, ne bi mogli izvedeti našega gesla.

Da ta sistem deluje, mora biti izračun gesel bolj zamotan, kot ta tule. Poleg tega si za geslo ne smemo izbirati smiselnih besed – vsaj ne ene same – sicer lahko nepridiprav naredi natančno isto, kot smo naredili v tej nalogi, namreč poišče besedo, ki se preračuna v našo številko. Najsibo izračun še tako zapleten, napiše lahko program, ki vzame vse besede iz slovarja in jih preskusi eno za drugo.

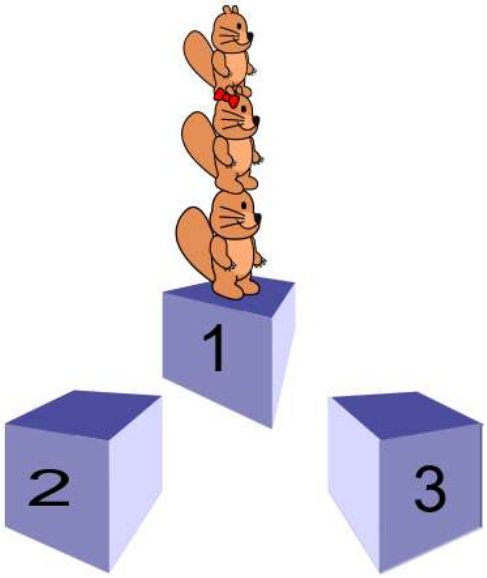
Akrobati

državno, 6. – 9. razred

Akrobatska družina bobrov (oče, mama, otrok) vadi novo akrobacijo. Imajo tri ploščadi. Na začetku vsi stojijo na prvi ploščadi, na ramenih drugega.

Skačejo tako, da naenkrat skoči le en bober, namreč tisti na vrhu. Skačejo le po ploščadih; skočijo lahko na prazno ploščad ali na ramena tistega, ki je že na ploščadi. Pri tem nihče ne sme skočiti na manjšega od sebe: otrok sme skočiti na mamo ali očeta, mama pa le na očeta. Oče ne sme skočiti na nikogar.

Kolikšno je minimalno število skokov, ki so potrebni, da vsa družina stoji na tretji ploščadi?

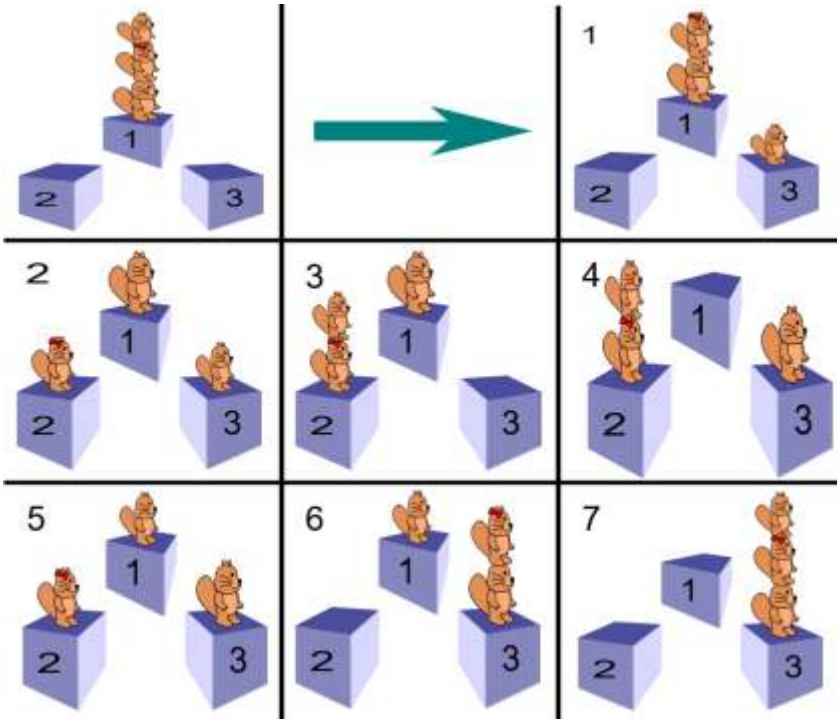


Rešitev

Sedem.

Celotna družina mora biti na tretji ploščadi. Najprej mora biti na tretji ploščadi oče. To pomeni, da mora mama biti na drugi ploščadi. Da mama skoči na drugo ploščad, mora otrok skočiti na tretjo ploščad. Tako dobimo naslednje zaporedje skokov:

1. Otrok skoči {1 na 3}
2. Mama skoči {1 na 2}
3. Otrok skoči {3 na 2}
4. Oče skoči {1 na 3}
5. Otrok skoči {2 na 1}
6. Mama skoči {2 na 3}
7. Otrok skoči {1 na 3}



Računalniško ozadje

V Indiji stoji starodavni tempelj. V njem so trije drogovi, na katerih je 64 različno velikih obročkov. Menihi jih neutrudno prestavljajo po enakih pravilih, kot skače akrobatska družina: večjega obroča ne smejo nikoli prestaviti na manjšega. Ko je bil svet še mlad, so bili vsi obroči na prvem drogu. Menihi jih prestavljajo in prestavljajo ... Ko bodo vsi obroči na zadnjem drogu, bo konec sveta.

Nalogo v takšni obliki poznamo pod imenom Hanojski stolpi. Ogledaš si jo lahko na Wikipediji: Hanojski stolpi.

Bo sveta kaj kmalu konec? Pravzaprav – kako, da ga še ni?

Če imamo n obročev, za prestavljanje potrebujemo 2^n-1 potez. Poglejmo dokaz; kdor ne razume, pa naj ga brez skrbi preskoči in potem bere naprej.

1. Če bi imeli le en obroč, bi ga premaknili na tretjo ploščad v eni potezi.
2. Recimo, že da vemo, da za premik n-1 obročev potrebujemo 2^{n-1}-1 potez. Če dodamo še en obroč na dno, je potrebno premakniti n-1 obročev na drugo ploščad, za kar potrebujemo 2^{n-1}-1 potez. Nato premaknemo dodatni obroč na tretjo ploščad (1 poteza) in nato n-1 obročev na tretjo ploščad (2^{n-1}-1). Skupaj je to torej 2^{n-1}-1 + 1 + 2^{n-1}-1 = 2·(2^{n-1}-1)+1 = 2·2^{n-1}-1 = 2^n-1.

Če ne razumemo računa, nič hudega. Pomembno je, da razumemo posledice. Vsak dodaten obroč podvoji čas reševanja. Dva dodatna obroča ga početverita. Trije ga poosmerijo. Če dodamo deset obročev, bomo nalogo reševali tisočkrat dlje. Z dvajsetimi dodatnimi obroči je potez milijonkrat več. S tridesetimi milijardokrat.

64 obročev zahteva 18446744073709551615 potez. Če so menihi hitri in za vsako potezo potrebujejo samo eno sekundo, bodo za svoje delo potrebovali 584 milijard let. Vesolje je trenutno staro manj kot 14 milijard. Razlogi za skrb so torej odveč.

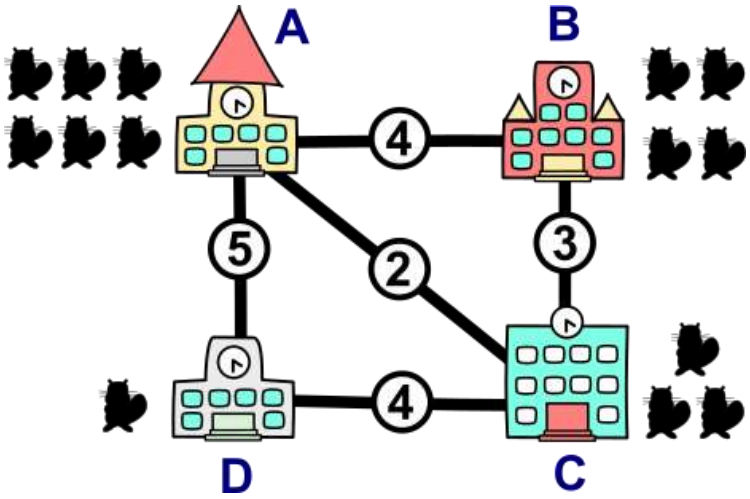
Kakšno zvezo ima to z računalništvom? Tudi ko opazujemo računalniške programe, nas zanima, koliko časa se bodo izvajali. Predvsem nas zanima, kako se bo podaljšal čas, če povečamo velikost problema. Nekateri postopki so takšni, da dvakrat več podatkov zahteva dvakrat dalj časa. Pri nekaterih dvakrat več podatkov zahteva le en korak več. Spet pri drugih je čas sorazmeren kvadratu velikosti podatkov: dvakrat več podatkov pomeni štirikrat daljši čas, petkrat več podatkov petindvajsetkrat daljšega.

Prestavljanje hanojskih stolpov zahteva čas, ki se z vsakim dodatnim podatkom podvoji. Če bi bilo obročev 65 namesto 64, bi menihi namesto 584 milijard let potrebovali dvakrat toliko, 1168 milijard let. Za takšne postopke učeno rečemo, da zahtevajo »eksponentni čas«. Po domače: takšni postopki so uporabni le za zelo majhne količine podatkov, saj hitro postanejo prepočasni.

Srečanje

državno, 6. – 9. razred

Srečati se mora šest predstavnikov šole A, štirje predstavniki šole B, trije predstavniki šole C in en predstavnik šole D. Dobili se bodo v eni od šol. Med šolami vozijo avtobusi, katerih cena za eno osebo je napisana na povezavah na sliki. Upoštevaj tudi prestopanja: pot iz šole B v D stane sedem bevrov (3 bevre do šole C in še 4 do šole D).



Na kateri šoli naj se dobijo, da bo skupna cena vseh vozovnic za vse predstavnike čim manjša?

Rešitev

Poračunajmo, koliko stanejo vozovnice, če se dobijo na posamezni šoli.

A: $6 \times 0 + 4 \times 4 + 3 \times 2 + 1 \times 5 = 27$ bevrov
B: $6 \times 4 + 4 \times 0 + 3 \times 3 + 1 \times 7 = 40$ bevrov
C: $6 \times 2 + 4 \times 3 + 3 \times 0 + 1 \times 4 = 28$ bevrov
D: $6 \times 5 + 4 \times 7 + 3 \times 4 + 1 \times 0 = 70$ bevrov

Dobili se bodo na šoli A. Poti do C so sicer cenejše tako za one s šole B kot one z D, vendar je na A več potnikov.

Računalniško ozadje

Takšnim nalogam pravimo optimizacijske naloge: od nas zahtevajo, da poiščemo najcenejšo (ali po kakem drugem kriteriju najugodnejšo) rešitev v okviru danih omejitev. Različnih vrst optimizacijskih nalog je veliko in algoritmi za njihovo reševanje predstavljajo pomemben del računalništva.

Gledališke luči

državno, 6. – 9. razred

Barve luči se ne mešajo tako kot tempere pri likovnem pouku. Osnovne barve so rdeča, zelena in modra; če sestavimo rdečo in zeleno, dobimo rumeno. Če sestavimo vse dobimo belo; če nobene, je črna tema. Vse možne kombinacije kaže tabela.

rdeča					■	■	■	■
zelena			■	■			■	■
modra		■		■		■		■
kombinacija	■	■	■	■	■	■	■	
	črna	**modra**	**zelena**	**svetlo modra**	**rdeča**	**vijolična**	**rumena**	**bela**

Oder Mestnega gledališča osvetljujejo tri luči: rdeča, zelena in modra. Prižigajo in ugašajo se takole.

Rdeča je eno minuto ugasnjena, eno prižgana, eno ugasnjena, eno prižgana ...

čas	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**
		■		■		■		■		■		■		■		■		■

Zelena je dve minuti ugasnjena, nato dve minuti prižgana, dve ugasnjena, dve prižgana ... in tako naprej do konca predstave.

čas	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**
			■	■			■	■			■	■			■	■		

Modra je štiri minute ugasnjena, štiri prižgana, štiri ugasnjena ...

čas	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**
					■	■	■	■					■	■	■	■		

Predstava se začne ob 20.00 in je eno minuto v temi. Ob 20.03, na primer, se prižgeta rdeča in zelena luč, zato je oder osvetljen rumeno.

Kako je osvetljen oder 1 uro in 15 minut kasneje, ob 21.15?

Rešitev

21.15 je 75. minuta.

Rdeča luč se prižge ob lihih minutah, torej bo ob začetku minute 75 prižgana.

Kdaj je prižgana zelena, lahko ugotovimo tako, da izračunamo ostanek po deljenju s 4: če je enak 2 ali 3, je prižgana. $75 = 18 \times 4 + 3$. Zelena luč bo prižgana.

Za modro izračunamo ostanek po deljenju z 8. Če je enak 4, 5, 6 ali 7, je prižgana. $75 = 9 \times 8 + 3$. Modra luč je ugasnjena.

Ker svetita rdeča in zelena luč, je oder osvetljen rumeno.

Do rešitve lahko pridemo tudi drugače: vzorec se ponavlja vsakih 8 minut. Torej je v 75 minuti enako, kot je bilo v 67. minuti – in enako, kot je bilo v 59., 51., 43., 35., 27., 19., 11. ali 3.

Gre še hitreje: namesto da odštevamo po osem minut, lahko preprosto izračunamo ostanek po deljenju z 8.

Kako pa bi rešil nalogo, če bi se modra luč prižigala in ugašala na tri minute?

Računalniško ozadje

Številke 1, 2 in 4 niso izbrane naključno. Kako štejemo v dvojiškem zapisu? 00000, 00001, 00010, 00011, 00100, 00101, 00110, 00111, 01000, 01001, 01010, 01011, 01100, 01101, 01110, 01111, 10000, 10001 in tako naprej. Opazuj skrajno desno številko. Stalno se spreminja – 0, 1, 0, 1 ... Predzadnja številka je dvakrat 0, dvakrat 1, dvakrat 0, dvakrat 1. Tretja z desne je štirikrat 0, štirikrat 1 ... Luči v gledališču se torej prižigajo in ugašajo tako, kot da bi štele. 75 zapišemo kot 1001011. Zanimajo nas le zadnje tri številke, 0, 1, 1, ki ustrezajo modri, zeleni in rdeči. Modra je ugasnjena (0), zelena prižgana (1) in rdeča prižgana (1).

Zanimivo je tudi ozadje zgodbe: barve. Pri likovnem pouku si se učil(a) o mešanju osnovnih barv – rumene, rdeče in zelene. Takšnemu mešanju rečemo *odštevalno mešanje*. Bela svetloba je sestavljena iz vseh barv. Papir je bel, ker odbija vse barve. Rumena barva vsebuje barvila, ki vpijajo vse barve, razen takšne mešanice barv, ki jo oko zazna kot rumeno. Rumena barva torej prepreči papirju, da bi odbijal določene barve. Če rumeno pomešamo z rdečo, bo papir vpijal še nekatere druge barve; tisto, kar se bo od njega še odbijalo, bo oko zaznalo kot oranžno.

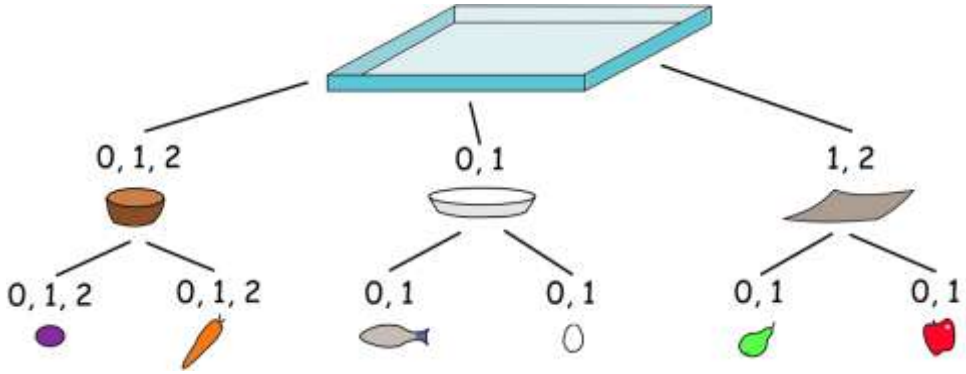
Računalniški zasloni, projektorji in pisane luči pa barve *dodajajo*. Tema je črna. Rdeča luč sveti v barvi, ki jo oko zazna kot rdečo. Če dodamo zeleno luč, dobimo mešanico barv, ki jo oko zazna kot rumeno. Od tega, koliko rdeče in zelene – ter ali dodamo še malo modre, je odvisen odtenek rumene, ki ga bomo dobili.

Osnovne barve pri tem *seštevalnem mešanju* so rdeča, zelena in modra. Te barve namreč zaznavajo celice v našem očesu. Več o teh zanimivih stvareh ti lahko pove učiteljica za biologijo.

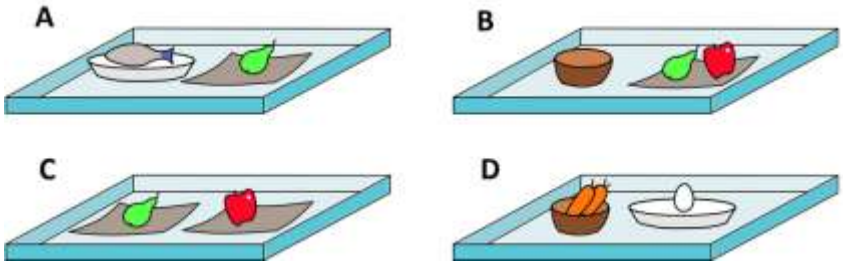
Šolsko kosilo

državno, 6. – 9. razred

Bobri gredo na šolsko kosilo. Vzeti morajo pladenj. Nanj morajo dati 0, 1 ali 2 skledi, 0 ali 1 krožnik in 1 ali 2 prtička. V skledo morajo dati 0, 1 ali 2 slivi in 0, 1 ali 2 korenčka. Na krožnik dajo 0 ali 1 ribo in 0 ali 1 jajce, na prtiček pa 0 ali 1 hruško in 0 ali 1 jabolko.



Štirje bobri so si izbrali kosila na spodnjih slikah. Katero od njih ni skladno s pravili?



Rešitev

Pladenj D nima prtička.

Računalniško ozadje

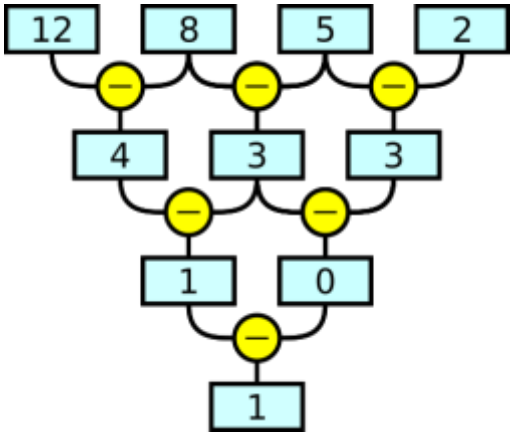
Kateri opis ti je prišel pri reševanju naloge bolj prav: slika ali besedilo? Slika, ne? Takšni sliki pravimo drevo (čeprav je obrnjeno na glavo). Tudi, kadar ga v resnici niti ne narišemo, si jih pogosto predstavljamo vsaj v mislih, saj so pregledna. Prav nam pridejo vsakič, ko razmišljamo o čem, kar je znotraj nečesa, kar je znotraj nečesa ...

Tudi pri programiranju so drevesa zelo uporabna za sistematično, učinkovito in pregledno shranjevanje podatkov.

Piramida

državno, 6. – 9. razred, 1. – 2. letnik

Če v nek stroj vpišemo števila 12, 8, 5 in 2, pokaže tako sliko.



Pri kateri od spodnjih kombinacij bo v zadnji, spodnji vrstici izpisa številka 0?

A. 13 9 7 6
B. 16 9 4 1
C. 13 8 4 2
D. 5 5 5 1

Rešitev

Pravilna rešitev je B.

Stroj v vsaki vrstici izpiše razlike zaporednih elementov prve. V drugi bo torej 7 5 3, v tretji 2 2 in v četrti 0.

So to edine štiri številke, ki dajo rezultat 0? Ne, prav gotovo ne: če vpišemo 1 1 1 1, bomo že v drugi vrstici dobili 0 0 0 in potem same ničle. Vedno, kadar vpišemo štiri enake številke, bodo ničle že v drugi vrstici. A to je preveč preprosto. Lahko poiščeš še kakšno drugo, manj dolgočasno četverko? (Namig: opaziš kaj posebnega v zaporedju?)

Računalniško ozadje

Računalniški čipi so sestavljeni iz različnih podenot. Ena od njih so “odštevalniki”. Stroj iz naloge je torej preprosto vezje.

Matematiki bi vedeli o tem vezju povedati še kaj več: vezje računa drugi odvod. Kogar zanima, lahko izve kaj več na http://mathworld.wolfram.com/FiniteDifference.html.

Dohiti me

6. – 9. razred, 1. – 4. letnik

Alja ima shranjenih 10 hlodov, Bernarda pa 1 hlod.

Naslednji dan se bosta podirali nova drevesa. Oba sta začeli glodati drevesa v gozdu natanko ob istem času. Vsako drevo, ki ga naglodata in podreta, shranita vsaka na svojem kupu.

Alja potrebuje eno uro, da podre eno drevo.

Bernarda je bolj spretna in dela vedno hitreje. V prvi uri podre eno drevo, v drugi uri podre dve drevesi, v tretji uri tri, v četrti štiri in tako naprej.

Čez koliko ur bo Bernarda prehitela Aljo?

Rešitev

Bernarda potrebuje najmanj 5 ur, da dohiti Aljo v številu podrtih dreves na kupu.

Zapišimo stanja kupov obeh bobrovk v tabelo:

Čas	Aljin kup dreves	Bernardin kup dreves
začetek	10	1
po 1 uri	11	2
po 2 urah	12	4
po 3 urah	13	7
po 4 urah	14	11
po 5 urah	15	16

Po 5 urah bo torej imela Bernarda na svojem kupu 16 dreves, Alja pa le 15.

Računalniško ozadje

Vprašanje v nalogi se nanaša na področje analize hitrosti algoritmov. Alja podira drevesa z linearno hitrostjo, kar zapišemo z $O(n)$. To pomeni, da bo skupno število podrtih dreves v n urah sorazmerno z n. Za Bernardo pa rečemo, da podira drevesa s kvadratno naraščajočo hitrostjo, kar zapišemo z $O(n^2)$. To pomeni, da bo skupno število Bernardinih podrtih dreves po n urah sorazmerno z n^2.

Skladišče želoda

državno, 6. – 9. razred, 1. – 4. letnik

Veverice nosijo želod v skupno skladišče. Število shranjenih želodov je zapisano na tabli pred skladiščem. Ko veverica prinese želod, stori naslednje:

- odloži želod v skladišču;
- prebere številko na tabli;
- gre domov in v miru razmisli, koliko je ta številka + 1;
- vrne se k skladišču, pobriše številko na tabli in napiše, kar je naračunala doma.

V začetku je skladišče prazno in na tabli je številka 0. Nato prihajajo veverice v tem vrstnem redu:

R1 R2 W1 R3 R4 W2 W3 R5 W4 R6 W5 R7 R8 R9 R10 W6 W7 W8 W9 W10

Najprej prva veverica odloži želod in prebere številko (R1). Nato druga veverica odloži želod in prebere številko (R2). Nato se prva veverica vrne in zapiše številko (W1). Nato tretja veverica odloži želod in bere (R3). Nato četrta veverica odloži želod in bere (R4). Nato se druga veverica vrne in piše (W2). Nato se tretja vrne in piše (W3). Nato peta odloži želod in bere (R5) ...

Katera številka bo napisana na koncu, ko deseta veverica zapiše, kar je naračunala?

Rešitev

Pod vsako število zapišimo, kaj veverica prebere oziroma zapiše:

R1	R2	W1	R3	R4	W2	W3	R5	W4	R6	W5	R7	R8	R9	R10	W6	W7	W8	W9	W10
0	0	1	1	1	1	2	2	2	2	3	3	3	3	3	3	4	4	4	4

Pod vsak R napišemo, kar je v tistem trenutku na tabli, to je, zadnjo rdečo številko. Pod W-je pišemo številko, ki je za 1 večja kot pri pripadajočem R; tako smo pod W6 napisali 3, ker je pod R6 napisana 2.

Veverice torej mislijo, da imajo v skladišču štiri želode, v resnici pa jih je 10.

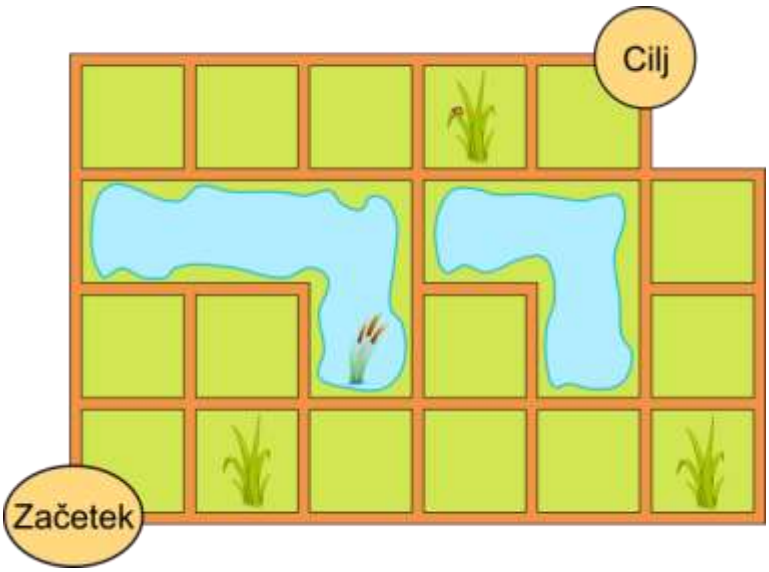
Računalniško ozadje

Današnji računalniki imajo več procesorjev ali jeder, pa tudi veliko število enot znotraj vsakega procesorja; lahko si jih predstavljaš kot ljudi, ki imajo več glav in lahko razmišljajo o več rečeh naenkrat. Do težav pa pride, kadar več enot bere in piše na isto mesto v pomnilniku. Če računalniki ne bi imeli posebnih mehanizmov, ki pazijo na to, bi prihajalo do natančno takšnih napak, kot jih delajo veverice v tej nalogi.

Obrni!

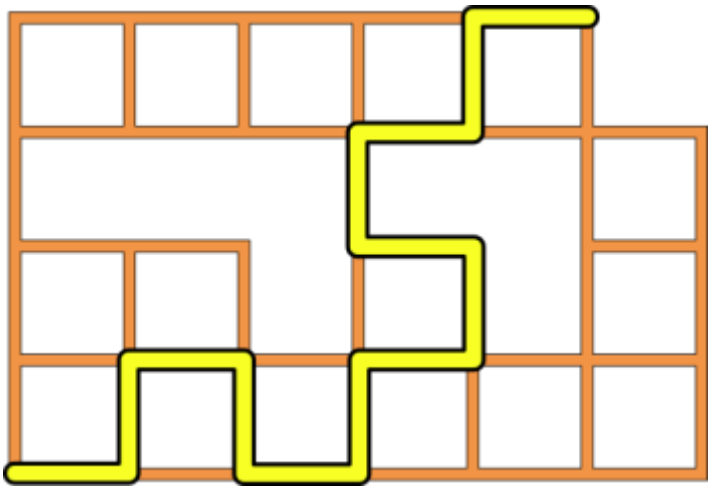
Muhasti kralj zahteva, da kočijaž zavije na vsakem križišču (ne glede na to, ali gre za križišče treh ali štirih cest). Kočijaž pa ga želi peljati po najkrajši poti – seveda upoštevajoč kraljevo zahtevo po zavijanju.

Kralja peljejo od spodnjega levega do zgornjega desnega oglišča zemljevida. Vsak odsek poti je dolg en kilometer. Koliko kilometrov bo dolga pot?

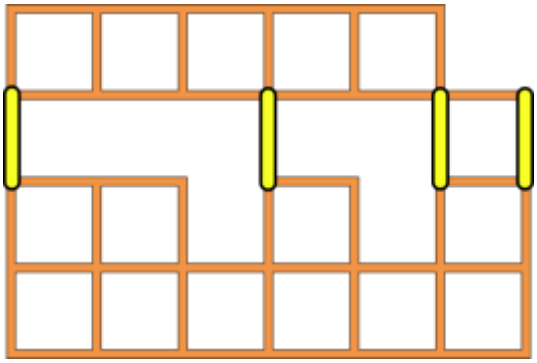


Rešitev

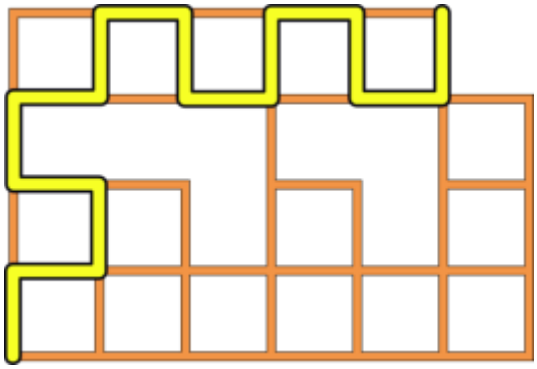
Trinajst kilometrov.

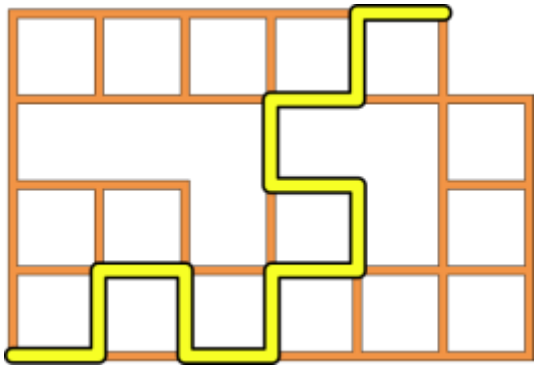


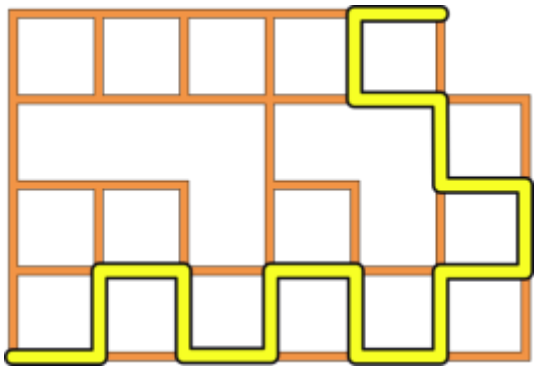
Pri reševanju precej pomaga, če opazimo, da bo moral kočijaž nujno po eni od štirih poti, ki so označene na spodnji sliki.

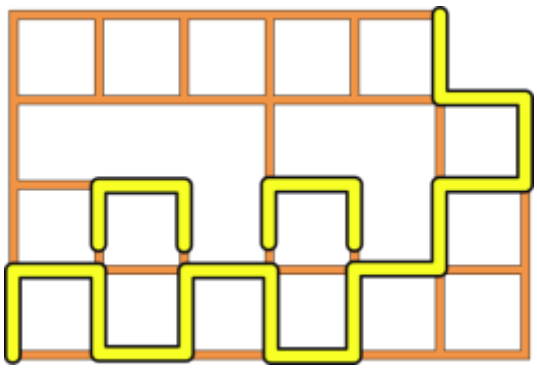


Za vsako od njih je lahko poiskati najkrajšo pod od začetne točke do te povezave in od te povezave do cilja. Najkrajša med njimi je druga.









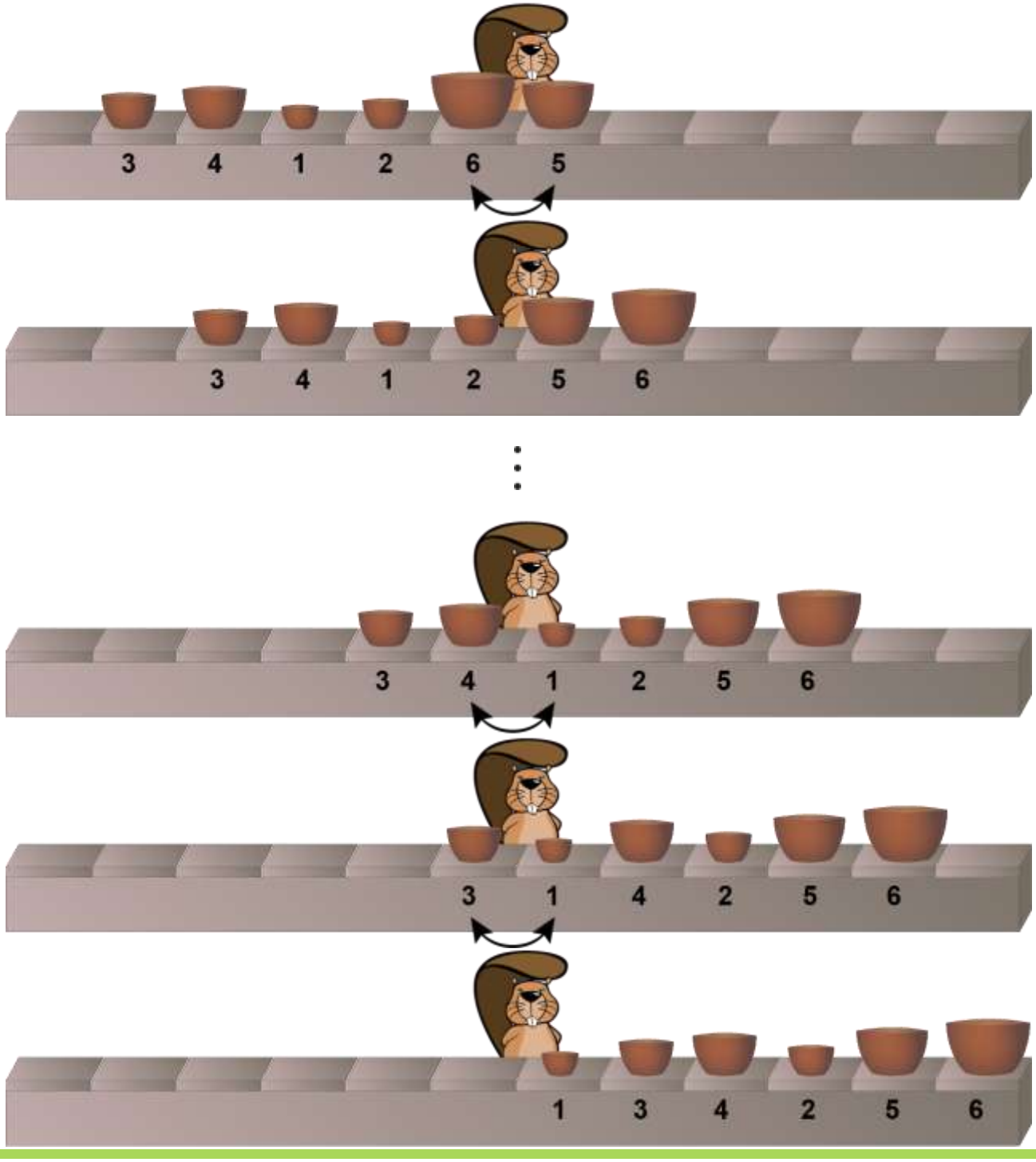
Računalniško ozadje

Število različnih poti je ogromno. Koliko jih je v posamezni takšni zgodbi, je odvisno od tega, kakšne neumnosti si izmišlja kralj, običajno pa se število poti z vsakim dodatnim križiščem približno podvoji. Pri iskanju najkrajše poti si zato niti z najhitrejšim računalnikom ne moremo privoščiti pregledati vseh možnosti. Enega od postopkov za reševanje te naloge smo spoznali ob nalogi Novi jez. Trik, ki smo ga uporabili tule, pa je drugačen: imenuje se deli in vladaj. Ideja postopkov, narejenih po tem načelu, je, da problem razdelimo na podobne (ali celo enake), vendar manjše probleme. Medtem ko bi bilo pregledovanje vseh smiselnih poti od začetka do konca brezupno, lahko hitro vidimo vse možne poti od začetka do ene od štirih izbranih poti in od te poti do konca. S tem problem razdelimo na dva podproblema in jih uženemo ločeno. Razdelimo in vladamo – kot stari Rimljani.

Tovarna skled

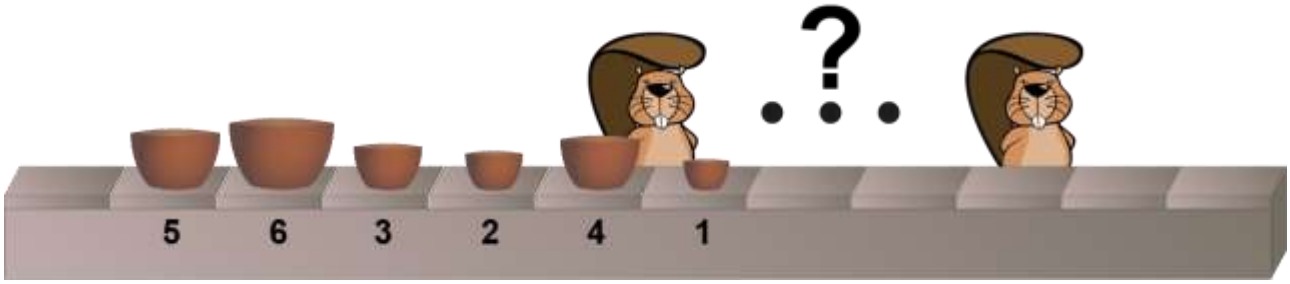
državno, 6. – 9. razred, 3. – 4. letnik

Tovarna izdeluje pakete šestih skled različnih velikosti. Tekoči trak jih pomika z leve proti desni. Stroj za izdelavo skled jih na trak zlaga pomešano. Pred pakiranjem jih morajo delavci, ki stojijo ob traku, urediti po velikosti (glej sliko). Vsak delavec spremlja sklede, ki gredo mimo njega. Ko vidi dve zaporedni skledi, ki ju je potrebno zamenjati, ju zamenja.



Delavci vedno menjajo le sosednji skledi. Isto skledo lahko zamenjajo večkrat (kot skledo velikosti 1 na gornjih slikah), nikoli pa se ne vračajo k skledam, ki so šle že mimo njih.

Koliko delavcev mora stati ob tekočem traku, da uredijo sklede na spodnji sliki?



Rešitev

Štirje.

- Začetni razpored skled je 5 6 3 2 4 1.
- Ko pridejo mimo prvega delavca, je razpored 1 5 6 3 2 4 (opravil je pet zamenjav, vedno s posodo 1).
- Ko pridejo mimo drugega, je razpored 1 2 5 6 3 4 (ta je vedno prestavljal skledo 2).
- Ko pridejo mimo tretjega, je razpored 1 2 3 5 6 4 (vedno prestavlja skledo 3).
- Ko pridejo mimo četrtega, je razpored 1 2 3 4 5 6 (prestavlja skledo 4).

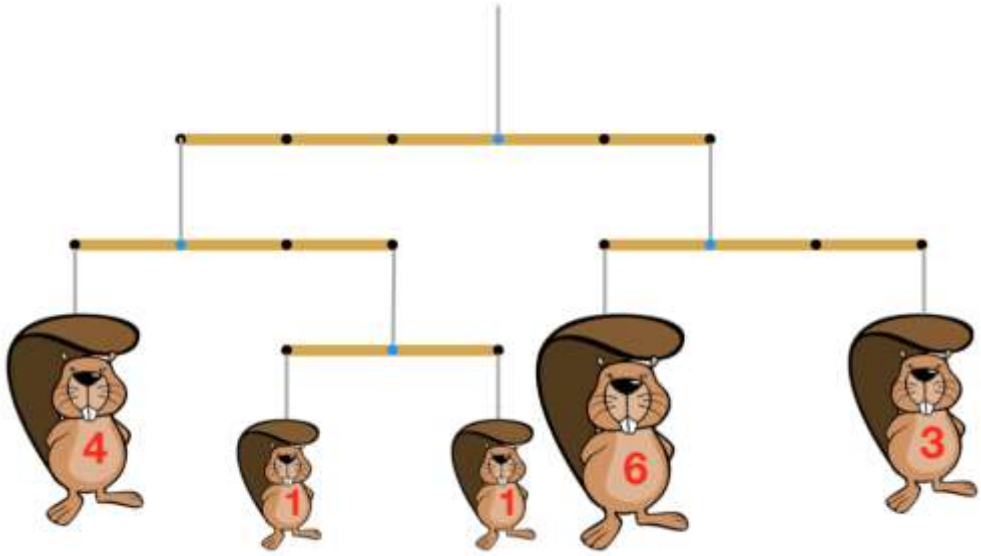
Računalniško ozadje

Delo s podatki je običajno veliko lažje, če so urejeni. Zato poznamo precej postopkov urejanja, primernih za različne situacije. Urejanju, kakršnega si spoznaval(a) v tej nalogi, pravimo urejanje z mehurčki. Postopek običajno deluje tako, da gre večkrat prek zaporedja in v vsakem prehodu zamenjuje le sosednje elemente. Zaporedje je urejeno, ko se prvič zgodi, da v celotnem prehodu ne zamenjamo ničesar več.

Urejanje z mehurčki je preprosto, žal pa ni prav učinkovito. Za urejanje 1000 elementov je v najslabšem primeru potrebnih pol milijona zamenjav. Boljši algoritmi jih potrebujejo le okrog deset tisoč.

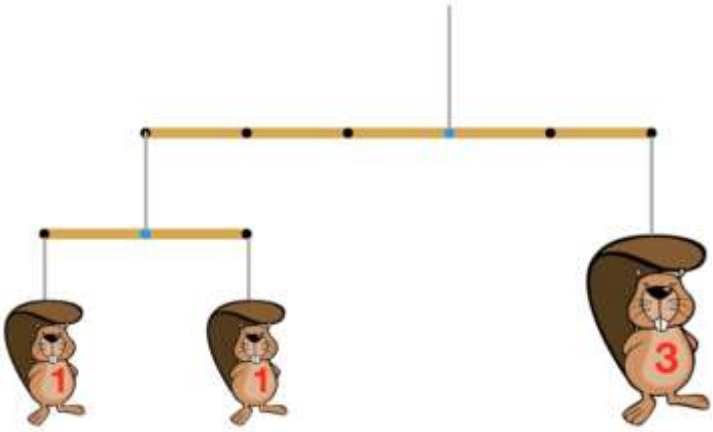
Obeski

8. razred – 9. razred, 1. – 4. letnik



Obeske na zgornji sliki opišemo z zaporedjem (-3 (-1 4) (2 (-1 1) (1 1))) (2 (-1 6) (2 3)).

Kako bi opisali obesek na spodnji sliki?



Rešitev

(-3 (-1 1) (1 1)) (2 3)

Poglejmo najprej primer.

Najprej imamo (-3 (-1 4) (2 (-1 1) (1 1))) (2 (-1 6) (2 3)): gornja palica štrli za tri dolžine levo in dve desno.

Tisto, kar sledi -3, namreč (-1 4) (2 (-1 1) (1 1)), opisuje, kaj visi na levi strani. Spet zapišimo jasneje: (-1 4) (2 (-1 1) (1 1)). Leva palica je obešena tako, da je en del levo od vrvice in dva desno.

Številki -1 sledi 4, saj je tam bober s številko 4. Številki 2 sledi (-1 1) (1 1), saj je en del palice levo in en del desno od vrvice, na vsaki strani pa visi bober s številko 1.

Zdaj pa še desni del: (-1 6) (2 3): en del levo in dva desno, na prvem visi bober s številko 6 in na drugem bober s številko 3.

Upoštevajoč primer rešitev sestavimo takole:

Na prvo palico sta pritrjena ena palica in en obesek. Palica je pritrjena 3 mesta levo od pritrdilne vrvice, obesek, to je bober s številko 3, pa 2 mesti desno od pritrdilne vrvice. Ta del zapišemo kot (-3 ?) (2 3).

Zdaj pogledamo še kaj visi na mestu -3, kjer smo zaenkrat napisali ?. Tu imamo palico, na kateri sta pritrjena 2 obeska, in sicer na mestu -1 bober s številko 1 in na mestu 1 prav tako bober s številko 1. Ta del obeska zapišemo z (-1 1) (1 1).

S tem delom nadomestimo prej napisan ? in celoten obesek predstavimo z zapisom (-3 (-1 1) (1 1)) (2 3).

Računalniško ozadje

Podatki so pogosto predstavljeni tako, da imamo nekaj znotraj nečesa znotraj nečesa znotraj nečesa ... Spomni se le na direktorije/imenike na disku. Po drugi strani moramo takšne reči pogosto zapisati "linearno", z zaporedji znakov ali števil. Tule si spoznal enega od načinov, ki ga uporabljamo.

Preveč oklepajev, praviš? Eden prvih in najpomembnejših programskih jezikov, Lisp, je bil videti natančno tako: sami oklepaji. Motivacija za takšen opis okraskov prihaja prav iz njega.

Lizike (1)

Ena lizika stane 12 bevrov.
Paket z dvema lizikama stane 20 bevrov.
Paket s štirimi lizikami stane 32 bevrov.
Paket osmih lizik stane 48 bevrov.
Škatla, v kateri je šestnajst lizik, stane 64 bevrov.

Kolikšen je najmanjši znesek, ki ga moramo plačati, če želimo kupiti 21 lizik? Na voljo je poljubno število paketov vsake velikosti. Seveda lahko kupimo tudi več lizik, kot jih potrebujemo, in višek razdelimo bobrom.

Rešitev

Za 21 lizik potrebujemo 108 bevrov.

Najprej preverimo, koliko stane ena lizika v posameznem paketu.

Ena lizika stane 12 bevrov.
Ena lizika v paketu dveh stane 20/2 = 10 bevrov.
Ena lizika v paketu štirih stane 32/4 = 8 bevrov.
Ena lizika v paketu osmih stane 48/8 = 6 bevrov.
Ena lizika v paketu šestnajstih pa stane 64/16 = 4 bevre.

Ugotovimo lahko, da so lizike v večjih paketih cenejše kot v manjših, torej gotovo ne bomo kupili dveh paketov enake velikosti (razen, morda, tistega s 16 lizikami), saj bi se v tem primeru bolj splačalo kupiti en večji paket.

Bomo morda kupili več lizik, kot jih potrebujemo? Preverimo. Kupimo lahko

2×16: 128 bevrov
$1 \times 16, 1 \times 8$: 112 bevrov
$1 \times 16, 1 \times 4, 1 \times 2$: 116 bevrov
$1 \times 16, 1 \times 4, 1 \times 1$: 108 bevrov

Drugih možnosti nam ni potrebno preverjati, saj bi v vseh drugih primerih večkrat kupili enak paket, kar pa se, vemo, ne splača.

Računalniško ozadje

Morda se ti zdi, da gre za nalogo iz pretvarjanja v dvojiški zapis? Ne. Ozadja problema je drugačno, številke so le slučajno postavljene tako, kot so. Kaj se v resnici skriva za to nalogo, pa boš izvedel(a) v razlagi naslednje.

Lizike (2)

Ena lizika stane 12 bevrov.
Paket z dvema lizikama stane 20 bevrov.
Paket s štirimi lizikami stane 44 bevrov.
Paket osmih lizik stane 72 bevrov.
Škatla, v kateri je šestnajst lizik, stane 150 bevrov.

Kolikšen je najmanjši znesek, ki ga moramo plačati, če želimo kupiti 21 lizik? Seveda lahko kupimo tudi več lizik, kot jih potrebujemo, in višek razdelimo bobrom.

Rešitev

Za 21 lizik potrebujemo 196 bevrov.

Poglejmo, kakšne so cene ene lizike v različnih paketih:
Ena lizika stane 12 bevrov.
Ena lizika v paketu dveh stane 20/2 = 10 bevrov.
Ena lizika v paketu štirih stane 44/4 = 11 bevrov.
Ena lizika v paketu osmih stane 72/8 = 9 bevrov.
Ena lizika v paketu šestnajstih pa stane 150/16 = manj kot 10, a več kot 9 bevrov (natanko 9,375).

Ugotovimo lahko, da se nam paketov po štiri in šestnajst lizik ne splača kupovati, saj je ceneje kupiti dva paketa po dve liziki kot en paket s štirimi lizikami ter dva paketa po osem lizik kot en paket s šestnajstimi.

Trije paketi po osem lizik bi stali $3\times72 = 216$ bevrov.
Dva paketa po osem in trije paketi po dve liziki bi stali $2\times72 + 3\times20 = 204$ bevre.
Dva paketa po osem in dva paketa po dve in ena lizika bi stali $2\times72 + 2\times20 + 1\times12 = 196$ bevrov.

Nima smisla, da bi vzeli več kot eno posamezno liziko, saj so te najdražje. Podobno velja tudi za paket z osmimi lizikami, ki je bolj ugoden kot štirje paketi z dvema lizikama.

Računalniško ozadje

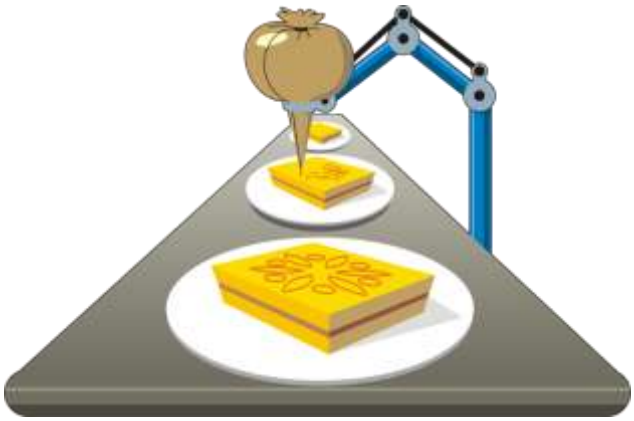
Naloga je primer problema, ki ga v računalništvu imenujemo *problem nahrbtnika*. Ta spada v družino problemov, pri katerih je naloga napolniti nahrbtnik s predmeti različnih velikosti ali cen. Obstaja več različic: včasih lahko v nahrbtnik spravimo le en primerek vsakega predmeta, včasih lahko predmete razrežemo, drugič spet ne, včasih so velikosti predmetov cela števila ...

Vendar pa računalničarji ne rešujejo tega problema zato, da bi dejansko napolnili svoje nahrbtnike, ampak zato, da bi optimizirali porabo pomnilnika, zagotovili varno komunikacijo ali rešili druge, na videz nepovezane probleme.

Dekoracija torte

8. - 9. razred, 1. – 4. letnik

V pekarni tort imajo vse avtomatizirano: pečene torte se peljejo po tekočem traku mimo robota z vrečko za dekoriranje, ki na torto nariše različne oblike.



Robot pozna naslednje štiri ukaze: `list`, `krog`, `obrni`, `ponovi`.

1. Ukaz `list` nariše:

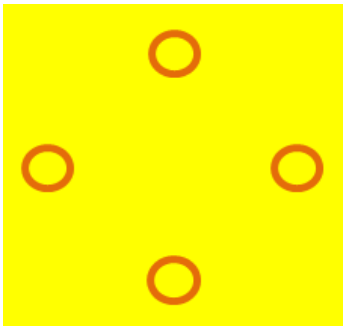
2. Ukaz `krog` nariše:

3. Ukaz `obrni k`: obrne torto za k stopinj v smeri urinega kazalca.
4. Ukaz `ponovi n [ ... ]`: n-krat ponovi ukaze, ki se nahajajo med oklepaji.

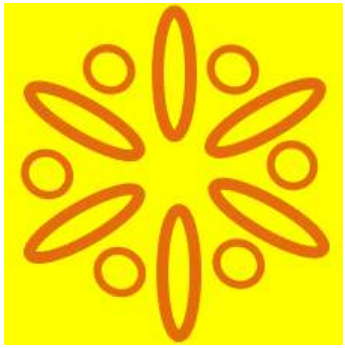
Primer: če robotu podamo ukaze

```
ponovi 4
  [ krog
    obrni 90 ]
```

bo robot na torto narisal naslednjo dekoracijo:



Katero od navedenih zaporedij ukazov robotu **ne nariše** dekoracije na sliki?



A.
```
ponovi 6
  [ obrni 30
    krog
    obrni 30
    list ]
```

B.
```
ponovi 6
  [ list
    obrni 60 ]
obrni 330
ponovi 6
  [ krog
    obrni 300 ]
```

C.
```
ponovi 6
  [ list
    obrni 60 ]
ponovi 6
  [ krog
    obrni 60 ]
```

D.
```
ponovi 3
  [ obrni 120
    ponovi 2
      [ list
        obrni 30
        krog
        obrni 150 ]
  ]
```

Rešitev

Pravilni odgovor je C.

Vsa zaporedja ukazov, razen ukazov pod C, narišejo podan vzorec, vendar vsako zaporedje ukazov nariše liste in kroge v drugem vrstnem redu. Ukazi pod C pa narišejo kroge na liste, ne pa med njih.

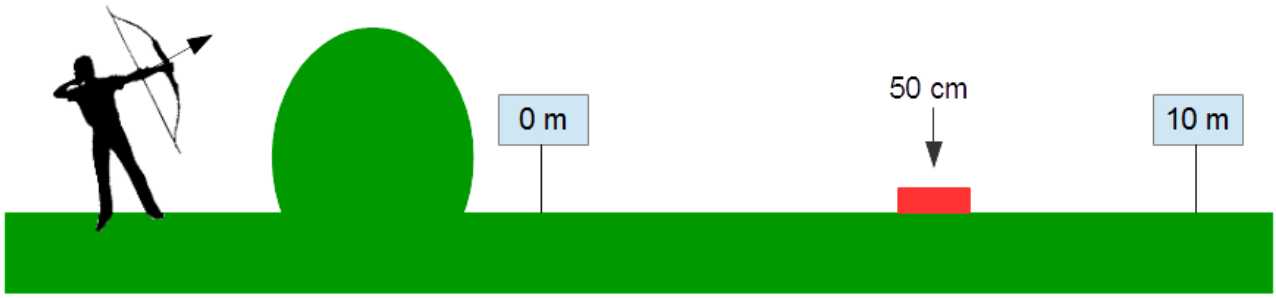
Računalniško ozadje

Problem v nalogi nam predstavi programiranje. Množica ukazov v nalogi je sicer zelo enostaven programski jezik, ki pozna funkcije z argumenti in zanko.

Iskanje tarče

državno, 8. – 9. razred, 1. – 4. letnik

Arnold je zelo natančen lokostrelec in lahko ustreli puščico natančno tako daleč, kot želi. Žal pa ne vidi tarče: o njej ve le, da je nekje med oznakama za 0 in 10 metrov. Po vsakem strelu mu prijatelj Marko pove le, ali je puščica priletela na tla pred tarčo ali za tarčo – ali pa jo je zadela.



Tarča je široka 50 cm. Koliko strelov potrebuje Arnold, da bo tarčo zagotovo zadel?

Rešitev

Potreboval bo največ pet strelov.

Najprej ustreli na razdaljo 5 metrov. Če je zadel, je to to. Sicer pa bo lahko iz tega, kar pove Marko, izvedel, da je tarča bodisi med 0 in 5 bodisi med 5 in 10 metri. Področje, na katerem je lahko tarča, je s tem razpolovil z 10 na 5 metrov. Drugo puščico izstreli na 2,5 ali 7,5 metra; s tem strelom bo ponovno razpolovil območje, na katerem se lahko nahaja tarča; zdaj ve, na katerem 2,5-metrskem področju je. S tretjo puščico to področje razpolovi na 1,25 metrov in s četrto na 0,6125 metra. Če ustreli peto puščico na sredo tega področja, bo tarča zadeta.

Računalniško ozadje

Postopek, ki ga uporablja Arnold, se imenuje *dvojiško iskanje* ali *bisekcija*: v vsakem koraku iskanja se število reči, po katerih iščemo ali velikost področja, kjer je rešitev (v tem primeru tarča) razpolovi. Ali, obratno, z eno puščico več lahko preiščemo dvakrat večje področje. S šestimi puščicami Arnold preveri 20 m, s sedmimi 40, z osmimi 80, z devetimi 160 in z desetimi že skoraj tretjino kilometra. Koliko puščic bi potreboval, če bi bila tarča nekje med Zemljo in Luno (in bi Arnold lahko ustrelil tako daleč)? Trideset.

Kadar z računalniki rešujemo probleme, ki se jih je mogoče lotiti z bisekcijo, lahko v zelo kratkem času rešimo zelo velike probleme. Poiskati, recimo, priimek v urejenem telefonskem imeniku ali prastaro elektronsko pošto v ogromnem predalniku, je preprosto in hitro prav zato, ker se ga lotevamo na podoben način, kot se Arnold loti iskanja tarče.

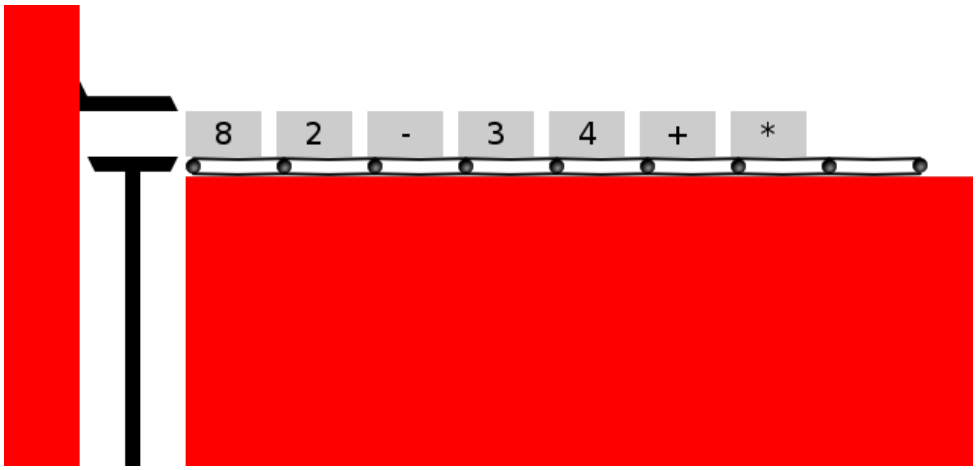
Računski stroj

državno, 8. – 9. razred, 1. – 4. letnik

Nek računski stroj deluje tako, da mu podajamo številke in operacije. Ko dobi številko, jo shrani na vrh kupa števil. Ko dobi operacijo (seštevanje, odštevanje, množenje, deljenje), vzame zgornji števili s kupa, izračuna operacijo (sešteje, odšteje, množi, deli) in shrani rezultat nazaj na kup namesto teh dveh števil.

Za uporabo tega stroja je potrebno račune zapisati nekoliko drugače. Na primer,

- 2 + 3 izračunamo kot 2 3 +,
- 10 – 2 izračunamo kot 10 2 –,
- 5 × 2 + 3 izračunamo kot 5 2 * 3 +,
- 5 + 2 × 3 izračunamo kot 5 2 3 * +,
- (8 – 2) × (3 + 4) izračunamo kot 8 2 – 3 4 + *.



Kako bi izračunali 4 × (8 + 3) – 2?

Rešitev

4 8 3 + * 2 –. Napišemo lahko tudi 4 3 8 + * 2 –, pa tudi 8 3 + 4 * 2 – in 3 8 + 4 * 2 –.

Računalniško ozadje

Takšen zapis izrazov je za ljudi neroden in nepregleden, računalniški programi pa veliko lažje delajo z izrazi v takšni obliki. Starejši kalkulatorji so zato uporabljali takšen zapis, še danes pa se uporablja v tiskalnikih.

Takšnemu zapisu izrazov rečemo obratni Poljski zapis (po Poljaku Jan Lukasiewiczu) ali tudi postskriptni zapis. Če na kakem tiskalniku kdaj opaziš besedo postscript, boš poslej vedel(a), zakaj: ker se računalnik in tiskalnik pogovarjata v jeziku postscript, ki je dobil ime po takšnem zapisu izrazov.

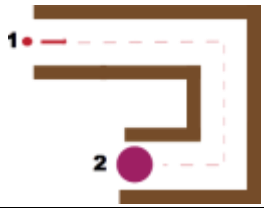
Pot iz labirinta

državno, 8. – 9. razred, 1. – 4. letnik

Ana, Berta, Cilka in Dani imajo različne strategije iskanja poti iz labirinta.

Ana Če ne stojim pred zidom, grem naravnost.

Če stojim pred zidom, se obrnem na desno.



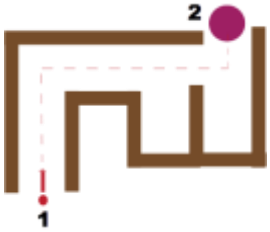
Berta Če ni zidu na levi, se obrnem levo in grem naprej.

Sicer: če stojim pred zidom, se obrnem desno.

Sicer: če ne stojim pred zidom, grem naprej.

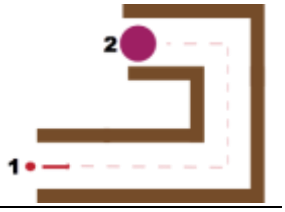
Cilka Če ne stojim pred zidom, grem naprej.

Če stojim pred zidom, se obrnem desno, grem naprej do naslednjega zidu, kjer se obrnem levo in grem naprej.

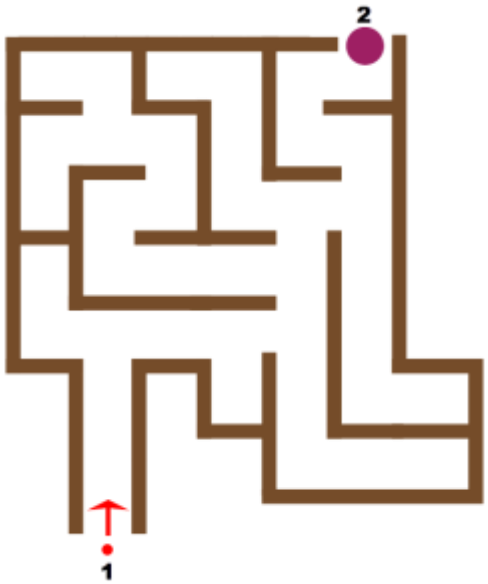


Dani Če ne stojim pred zidom, grem naravnost.

Če stojim pred zidom, se obrnem na levo.

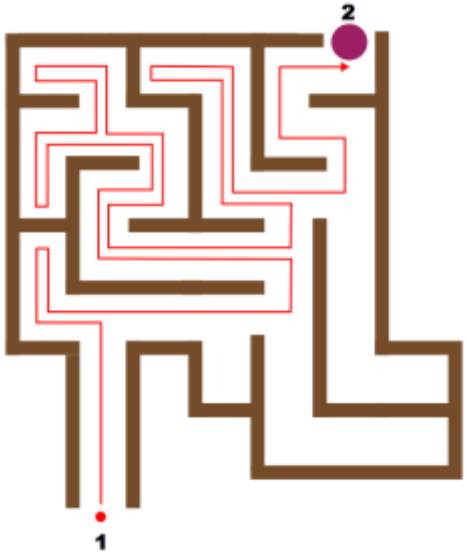


Katera od njih bo našla izhod iz spodnjega labirinta?

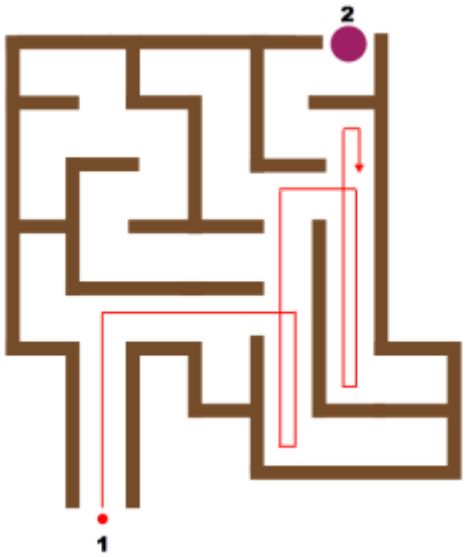


Rešitev

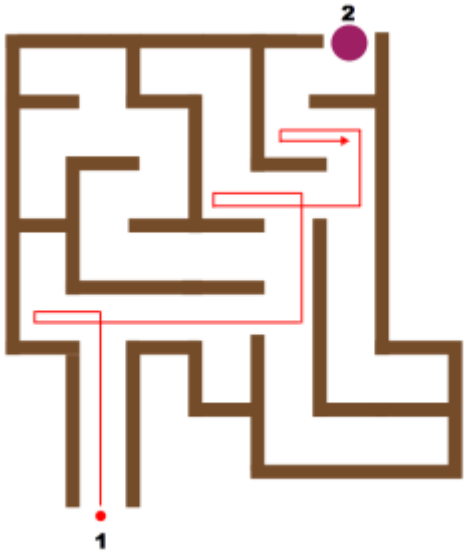
Berta. Njen postopek deluje v vsakem labirintu – razen, če jo postavimo nekam znotraj labirinta, v katerem je mogoče hoditi v krogu.



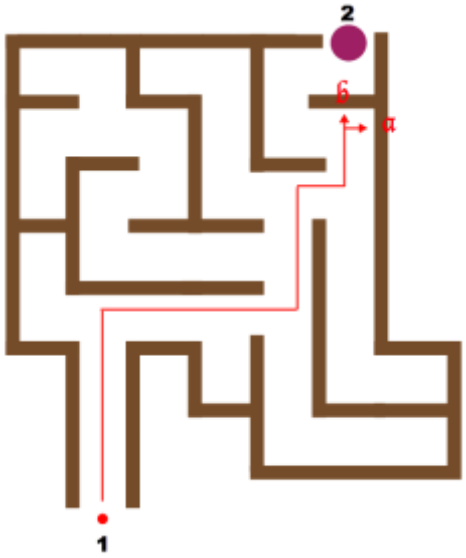
Ani ne bo uspelo. Ko pride že skoraj do izhoda, začne nenadoma krožiti po istem hodniku.



Dani tudi ne – zgodi se ji podobno kot Ani, le en hodnik kasneje.



Cilki se zgodi še nekaj nerodnejšega: tik pred koncem se zatakne v kotu in se obrača levo in desno in levo in desno ...



Računalniško ozadje

Iskanje splošnih postopkov za reševanje podobnih problemov je pomemben del računalniške znanosti. Želimo si postopkov, ki bi hitro in zanesljivo rešili vse možne oblike dane vrste problemov – v tem primeru postopek, ki zna rešiti vsak labirint.

Skrivna sporočila

1. – 4. letnik

Šifriranje, pri katerem vsak znak zamaknemo za, recimo, dva znaka naprej, pozna že vsak otrok. Tule je boljši trik. Črke, kot navadno, oštevilčimo. Dodali bomo še presledek, kot 26. črko.

A	B	C	Č	D	E	F	G	H	I	J	K	L	M	N	O	P	R	S	Š	T	U	V	Z	Ž	
									1	1	1	1	1	1	1	1	1	1	2	2	2	2	2	2	2
1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6

Če je premik lih, se premaknemo v levo, če je sod v desno. Če nas premik pripelje izven abecede, nadaljujemo z drugega konca.

Trik šifriranja pa bo v tem, da bomo vsakič uporabljali drug premik. Prvi premik je dogovorjen. Vsak naslednji premik pa je enak zaporedni številki pravkar šifrirane črke.

Recimo, da želimo šifrirati besedo LOM z začetnim premikom 5.

- L premaknemo za 5 levo, v G; naslednji premik bo 13, ker je to zaporedna številka L-ja.
- O premaknemo za 13 levo, v C; naslednji premik bo 16, saj je to zaporedna številka O-ja.
- M premaknemo za 16 desno; pri tem gremo prek desnega roba in pridemo v Č.

Besedo LOM torej zašifrifamo v GCČ.

Preverimo, ali razumeš. Kako zašifriramo besedo KIP, če kodiranje začnemo s premikom 2?

Rešitev

- K premaknemo za 2 desno, dobimo M; naslednji premik bo 12.
- I premaknemo za 12 desno, dobimo U; naslednji premik bo 10.
- P premaknemo za 10 desno, dobimo A.

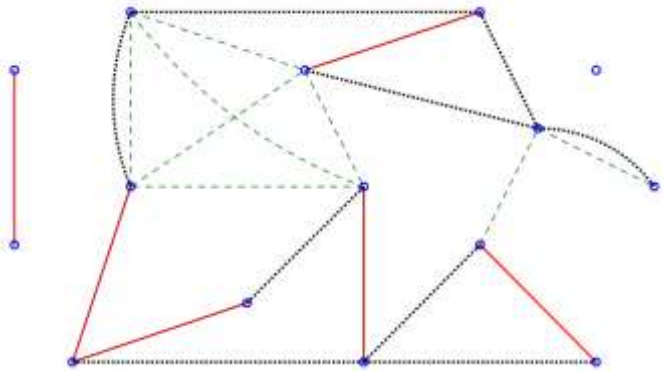
Rešitev je torej MUA.

Računalniško ozadje

Takšne načine šifriranja so si izmišljali pred stotimi leti. Za današnje potrebe so neuporabna, saj jih je prelahko razdreti. Veš, kako bi se lotil branja tako zašifriranih sporočil, če ne bi poznal ključa?

Znanci na busu

1. – 4. letnik



Peter, Jure in Vid so na neki posebno dolgočasni vožnji z avtobusom vprašali nekatere potnike, ali se poznajo in na kakšen način. Rezultat so narisali.

Vsaka točka predstavlja potnika.

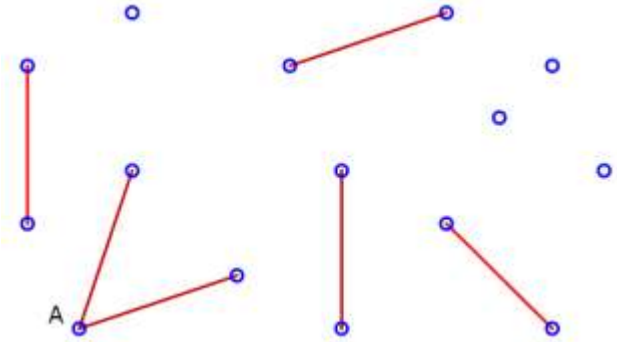
- Peter je povezal dvojčke z neprekinjeno rdečo črto.
- Jure je povezal prijatelje s pikčasto črno črto.
- Vid je povezal sošolce s prekinjeno zeleno črto.

Ugotovimo lahko dvoje. Prvič, res jim je bilo dolgčas. Drugič, le eden je narisal pravilno sliko. Kdo?

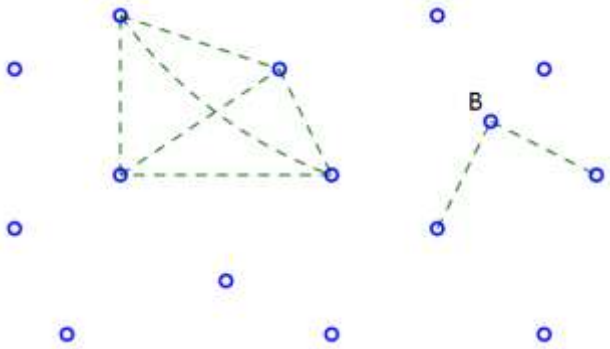
Rešitev

Jure.

Petrove rdeče povezave kažejo, da je A dvojček z dvema osebama. To ne more biti res.



Vidove povezave kažejo, da je B sošolec dvema, ki med sabo nista sošolca. Če predpostavimo, da Vid ne hodi v dve šoli (kar je čisto smiselno predpostaviti), je to nemogoče.



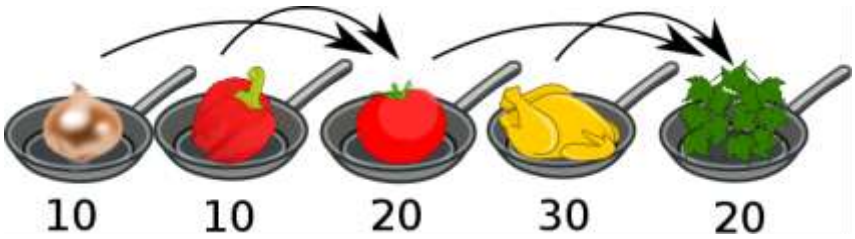
Računalniško ozadje

Naloga se ukvarja z relacijami, ki pridejo prav na vseh mogočih področjih računalništva. Relacije imajo lahko določene lastnosti in te imajo seveda učena imena. Tako, recimo, za relacijo "biti sošolec" pravimo, da je tranzitivna ali, po domače, "sošolec mojega sošolca je sošolec". Relacija "biti prijatelj" te lastnosti, kot dobro vemo, nima.

Čahohbili

1. – 4. letnik

Kuharski mojster Sergej zelo rad kuha, najraje pa pripravlja gruzijsko nacionalno jed *čahohbili*.



Za pripravo čahohbilija na vrtu uporablja en sam kuhalnik. Postopek priprave je takšen:

1.	Skuhaj čebulo.	10 minut
2.	Skuhaj papriko.	10 minut
3.	Združi kuhano čebulo in kuhano papriko, dodaj paradižnik ter kuhaj.	20 minut
4.	Skuhaj piščanca.	30 minut
5.	Združi vse skuhano pod koraki 3 in 4, dodaj začimbe ter kuhaj vse skupaj.	20 minut

Za pripravo čahohbilija potrebuje Sergej skupaj 90 minut.

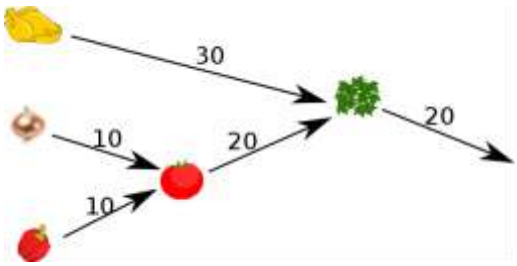
Kadar Sergej kuha v domači kuhinji, uporablja več kuhalnikov hkrati, zato je jed pripravljena hitreje. Katera od navedenih trditev **ni pravilna**?

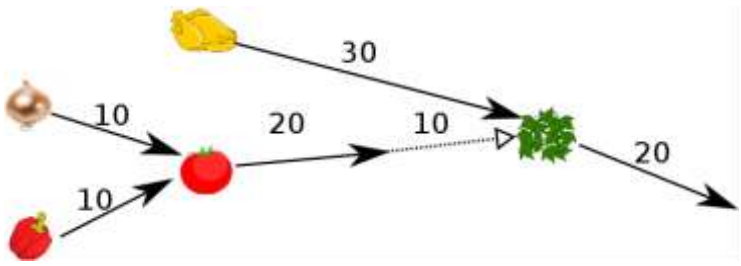
A. Sergej lahko skrajša čas kuhanja za 10 minut, če uporabi 2 kuhalnika.
B. Sergej lahko skrajša čas kuhanja za 30 minut, če uporabi 2 kuhalnika.
C. Sergej lahko skrajša čas kuhanja za 40 minut, če uporabi 3 kuhalnike.
D. Sergej lahko skrajša čas kuhanja za 50 minut, če uporabi 4 kuhalnike.

Rešitev

Pravilni odgovor je D.

Leva slika prikazuje, kako lahko skrajšamo čas kuhanja za 40 minut (odgovor C). Ker uporabimo tri kuhalnike, lahko sočasno pripravljamo na enem piščanca (30 minut), na drugih dveh pa najprej čebulo in papriko (po 10 minut) ter nato oboje skupaj 20 minut. Po tridesetih minutah združimo vse skuhano, dodamo začimbe in kuhamo še 20 minut. Skupen čas kuhanja je tako 50 minut, kar je 40 minut manj, kot če uporabimo le en kuhalnik.





Desna slika pa prikazuje, kako skrajšamo čas kuhanja za 30 minut (za odgovora A in B). Z uporabo dveh kuhalnikov lahko sočasno kuhamo čebulo in papriko (po 10 minut), nato pa na enem kuhalniku združeno čebulo in papriko (20 minut), na drugem kuhalniku pa že začnemo s kuhanjem piščanca (30 minut). Na koncu vse skuhano združimo, dodamo začimbe in kuhamo še 20 minut. Skupni čas kuhanja je tako 60 minut (10 + 30 + 20), kar je 30 minut manj, kot če uporabimo le en kuhalnik.

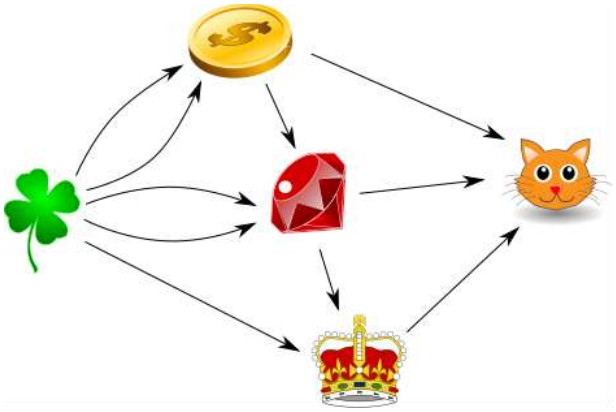
Računalniško ozadje

Kuhalniki v nalogi so računalniški viri, kot so na primer procesorji. Če imamo le en vir, moramo naše delo opraviti zaporedoma. Če pa imamo na voljo več virov, lahko včasih naloge izvajamo vzporedno (sočasno).

Skrajšanje časa je podobno strukturiranju programske kode tako, da se izvaja kar najhitreje glede na razpoložljivo število procesorjev. Vzporedno računanje je pomembno raziskovalno področje v računalništvu.

Alkimist

1. – 4. letnik



Alkimist je umetnik preoblikovanja snovi. Lahko spremeni en predmet v drugega:

- dve deteljici v kovanec;
- kovanec in dve deteljici v rubin;
- rubin in deteljico v krono;
- kovanec, rubin in krono pa v mačko.

Vsi predmeti, ki jih uporabi, se pri preoblikovanju spremenijo v nov predmet in izginejo.

Koliko deteljic potrebuje, da ustvari mačko? Pet, deset, enajst ali dvanajst?

Rešitev

Pravilni odgovor je 11.

Preoblikovanja so naslednja:
kovanec = 2 deteljici
rubin = 2 deteljici + 1 kovanec = 4 deteljice
krona = 1 rubin + 1 deteljica = 4 deteljice + 1 deteljica = 5 deteljic
mačka = 1 kovanec + 1 rubin + 1 krona = 2 deteljici + 4 deteljice + 5 deteljic = 11 deteljic

Odgovor 5 dobijo tisti, ki slike ne preberejo pravilno in ne opazijo, da potrebujemo 2 deteljici, da lahko ustvarimo kovanec ali rubin. Ali pa tisti, ki ne opazijo, da potrebujemo kovanec za ustvarjanje rubina ali rubin za ustvarjanje krone.

Odgovor 10 pa dobijo tisti, ki zmotno menijo, da je potrebno število deteljic enako številu povezav.

Če ste dobili odgovor 12 (D), ste se ušteli na kak drug, izviren način.

Računalniško ozadje

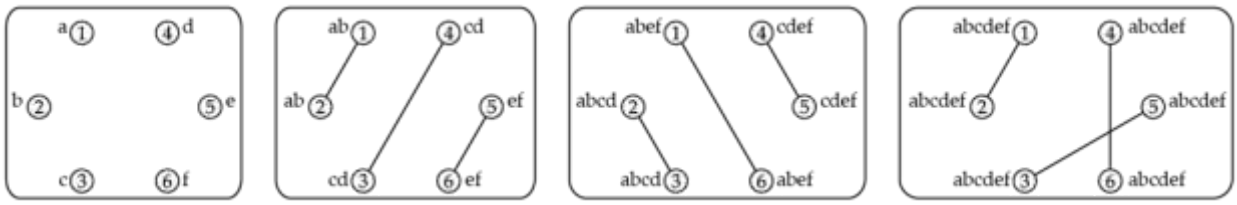
Naloga prikazuje, kako lahko uporabimo graf za prikaz odvisnosti med stvarmi. Graf je podatkovna struktura, ki se veliko uporablja v računalništvu za prikaz razmerij. Graf nam tudi pomaga pri vizualizaciji naloge, kar je veliko lažje razumeti kot pri branju tekstovnega opisa povezav.

Vohuni

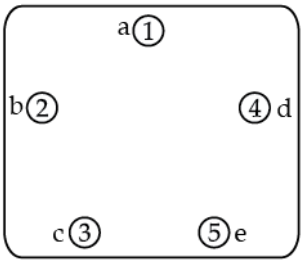
1. – 4. letnik

Vsak petek si šest vohunov izmenja vse informacije, ki so jih zbrali med tednom. Zaradi varnosti se vsak vohun vedno dobi le z enim drugim vohunom, saj ne želi, da bi ga videli skupaj z več vohuni. Tako morajo vohuni organizirati več zaporednih srečanj, na katerih se sestanejo v parih ter izmenjajo vse informacije, ki so jih prejeli do tega srečanja.

Skupina šestih vohunov potrebuje le tri zaporedna srečanja, da si izmenjajo vse skrivnosti. Pred prvim srečanjem vsak vohun pozna po en del skrivnosti (vohun 1 ve '*a*', vohun 2 pozna '*b*' in tako naprej). Na prvem srečanju se dobita vohuna 1 in 2, izmenjata informacije in tako oba poznata skrivnost '*ab*'. Spodnji diagrami prikazujejo, kateri vohuni se srečajo na vsakem od zaporednih srečanj (vohuna, ki se srečata, sta povezana s črto). Poleg tega so pri vohunih pripisane tudi skrivnosti, ki jih v danem trenutku poznajo. Po treh srečanjih vsi vohuni poznajo vse skrivnosti.



Po odmevnem mednarodnem incidentu se eden od vohunov ni več udeleževal srečanj. Kakšno je minimalno število zaporednih srečanj preostalih petih vohunov, da si lahko izmenjajo vse informacije?



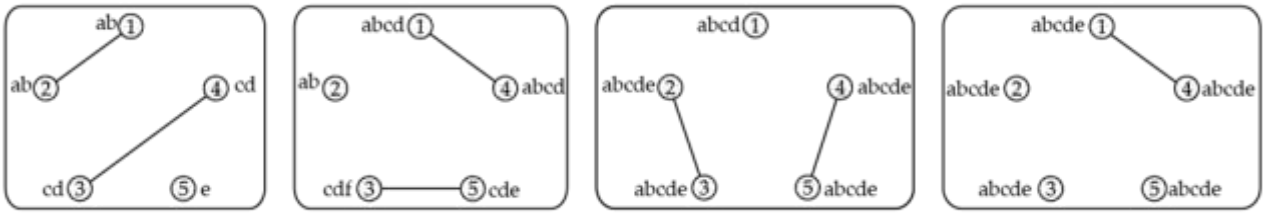
Rešitev

Pravilni odgovor je 4 srečanja.

Odgovor je verjetno nekoliko nepričakovan, saj bi na prvi pogled bolj očiten odgovor 3 (ali manj), saj imamo manj vohunov. Še bolj nenavaden pa je ta pravilni odgovor ob upoštevanju dejstva, da bi štirje vohuni očitno lahko izmenjali vse informacije na le dveh srečanjih.

Vendar nas neuspešni poskusi za rešitev naloge v treh ali manj srečanjih kmalu pripeljejo do vzroka tega problema: ker je število vohunov liho, je na vsakem srečanju eden od njih neaktiven (nima se s kom srečati). Recimo, da vohun 5 ne sodeluje na prvem srečanju, vendar sodeluje na drugem srečanju. Po drugem srečanju tako le dva vohuna poznata njegovo skrivnost 'e'. Na tretjem srečanju se ta dva vohuna srečata z dvema drugima vohunoma, torej skrivnost 'e' poznajo le štirje vohuni. Zato potrebujemo še četrto srečanje, da skrivnost 'e' izve še zadnji vohun.

Tako smo pokazali, da potrebujemo *najmanj* 4 srečanja. Da pokažemo tudi, da je so 4 srečanja dejansko dovolj, smo pripravili še diagram srečanj.



Računalniško ozadje

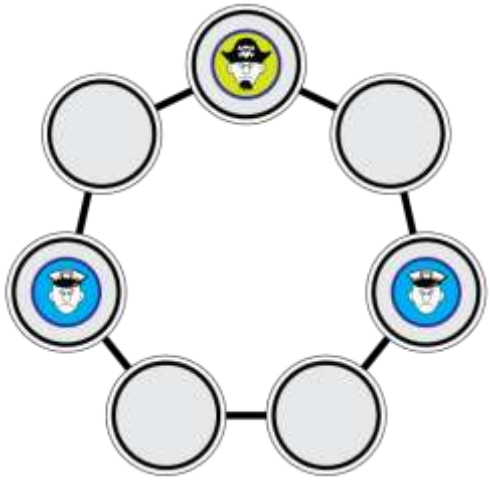
Kadar si računalniki izmenjujejo informacije, to pogosto delajo v parih. Tako lahko nastane problem, kako naj si v najkrajšem času izmenjajo informacije vsi računalniki v mreži. Torej morajo računalničarji rešiti podoben problem, kot so ga imeli vohuni. Problem je poznan tudi kot problem opravljivcev (angleško *gossip problem*). Poskusite ga rešiti za različno število vohunov in morda boste ugotovili zanimivo pravilo.

Rešitev problema je bila prvič ugotovljena (in tudi opisano splošno pravilo za reševanje) leta 1975. Ta problem in tudi več njemu podobnih problemov so povezani z različnimi področji računalništva, ki se ukvarjajo z izmenjavo podatkov, komunikacijskimi omrežji in kriptografijo.

Pirati

1. – 4. letnik

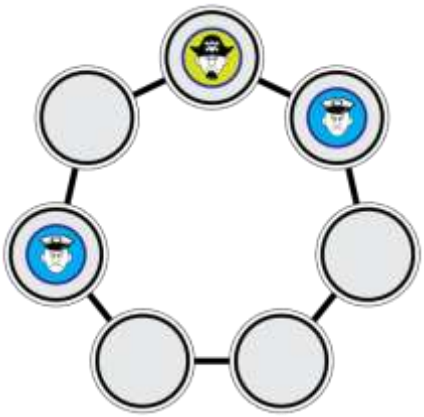
Jana in Janko igrata namizno igro *Lovec na pirate*. Ob vsaki potezi se eden od policistov (a ne oba) premakne na sosednje polje. V naslednji potezi se premakne pirat, ki pa je hitrejši od policistov in se vedno premakne za dve mesti. Policist se lahko premakne le na prosto mesto (ne more se premakniti na mesto, ki ga zaseda njegov kolega policist ali pirat). Igra se konča, ko je pirat prisiljen, da se premakne na mesto, ki ga zaseda policist. Situacijo prikazuje spodnja slika, le da je trenutno na potezi policist. Za zmago mora torej policist spraviti pirata v pozicijo, ki jo prikazuje slika, ko je na potezi pirat.

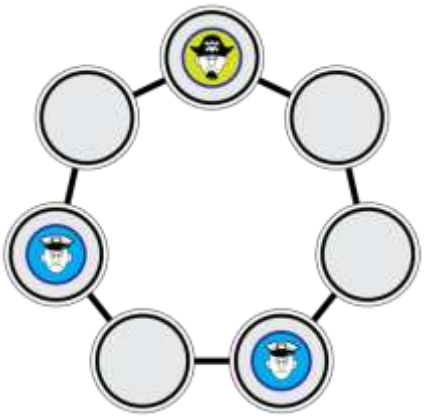


Jana, ki igra pirata, je precej spretna pri izogibanju policistom. Pomagajte Janku, ki igra policista, da bo odigral popolno igro in zmagal. Koliko potez mora narediti, preden bo lahko ujel pirata?

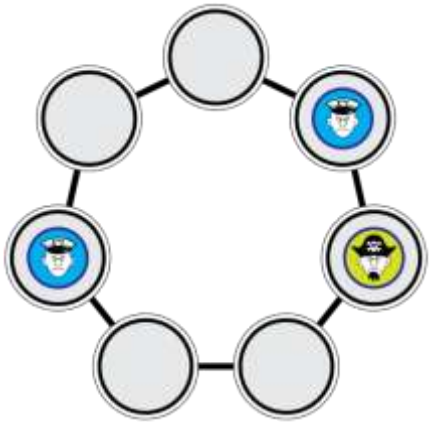
Rešitev

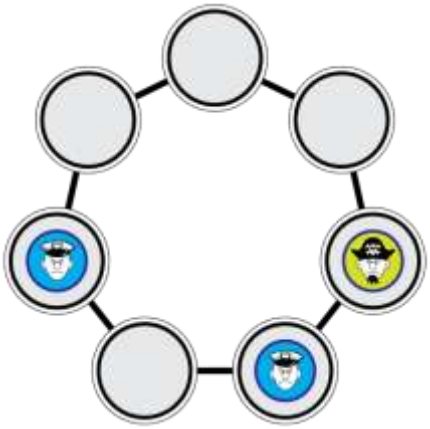
Če Jana igra dovolj spretno, Janko ne more zmagati. Predpostavimo, da bi mu jo uspelo prisiliti v položaj, ki ga prikazuje slika, in bi bila tako prisiljena v premik na polje, ki ga zaseda policist, s čimer izgubi igro. Kakšna je bila postavitev pred Jankovo zadnjo potezo? Janko je lahko premaknil levega ali desnega policista za eno polje gor ali dol. Ker je igralna deska simetrična, bomo predpostavili, da se je premaknil desni policist (situacija je enaka, če se premakne levi policist). Torej smo imeli pred zadnjo potezo eno od naslednjih situacij:





Pojdimo še eno potezo nazaj. Pirat je na svoje mesto lahko prišel le z desne strani, saj je na levi strani policist. Torej je bila situacija pred potezo pirata naslednja:





Potemtakem je lahko situacija na sliki, ki vodi v prijetje pirata, lahko nastala le iz zgornjih dveh pozicij (ali njunih zrcaljenih različic). Vendar pa je Jana resnično dobra igralka te igre in če bi se znašla v eni izmed zgoraj teh dveh situacij, bi pirata pač premaknila levo in ne gor.

Računalniško ozadje

Programi, ki igrajo namizne igre, računajo možne "poti" skozi igro. Navadno začnejo v trenutnem stanju igre (za razliko smo v naši nalogi pregledovali stanja od končnega stanja nazaj) in preračunajo vse možne poteze, ki jih lahko naredita oba igralca. Ocenijo vse poteze, ki jih lahko naredi računalnik, pri tem pa predpostavijo, da bo nasprotnik naredil najboljšo možno potezo. V bolj kompleksnih igrah, kot je na primer šah, računalnik analizira poteze le do neke globine (nekje do 15 potez) in nato približno oceni kvaliteto pozicije.

Robota

1. – 4. letnik

Janez je izdelal dva robota za risanje, ki poznata ukaza:

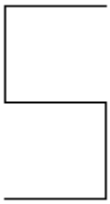
- **naprej** – robot gre en korak naprej,
- **obrat** – robot se obrne za 90 stopinj.

Robota je postavil na tla in vsakemu od njiju poslal tri ukaze. Pri tem je opazil, da se eden od robotov pri ukazu **obrat** obrne za 90 stopinj v desno, drugi pa se pri njem obrne za 90 stopinj v levo.

Nato je Janez želel narisati nekaj slik s pomočjo obeh robotov. Postavil ju je na tla na dve različni mesti, vendar je obema vedno poslal iste ukaze. Robota ne moreta biti sočasno na istem mestu.

Katere od spodnjih slik **nista mogla narisati** robota?

A.



B.

C.

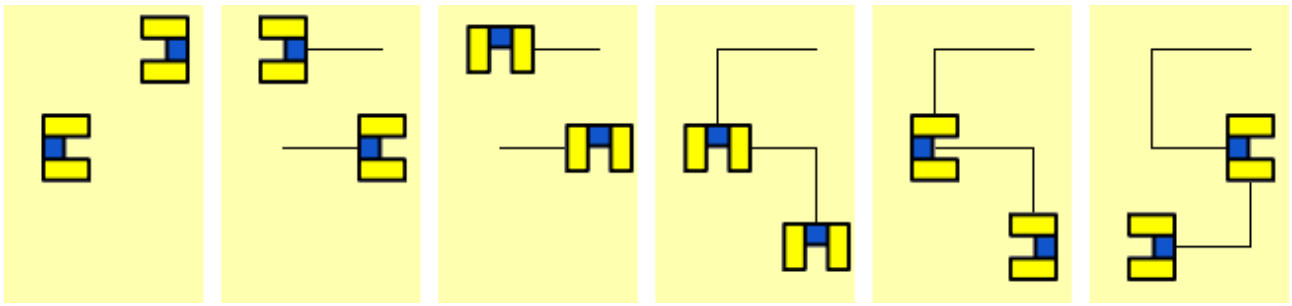
D.

Rešitev

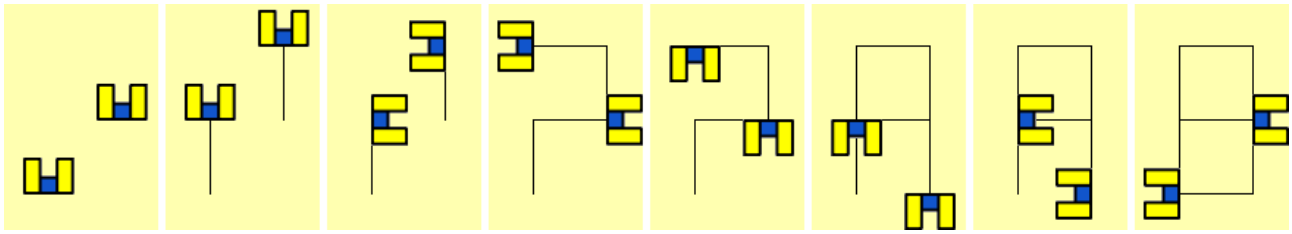
Pravilni odgovor je D. Robota nista mogla narisati črke L.

Najprej poiščimo način, kako bi robota lahko narisala znake pod rešitvami A, B in C.

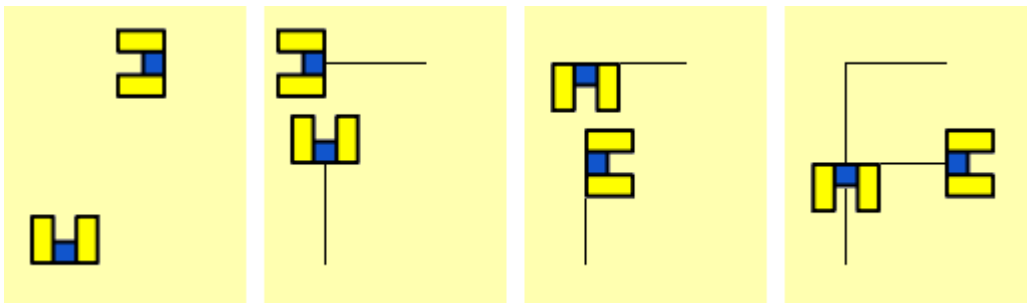
A: Prva slika prikazuje začetni poziciji in orientaciji obeh robotov. Vsakemu robotu pošljemo zaporedje naslednjih ukazov: naprej, obrat, naprej, obrat, naprej.



B: Prva slika prikazuje začetni poziciji in orientaciji obeh robotov. Vsakemu robotu pošljemo zaporedje naslednjih ukazov: naprej, obrat, naprej, obrat, naprej, obrat, naprej.



C: Prva slika prikazuje začetni poziciji in orientaciji obeh robotov. Vsakemu robotu pošljemo zaporedje naslednjih ukazov: naprej, obrat, naprej.



Sedaj pa pokažimo še, da robota ne moreta narisati črke L. Najprej lahko ugotovimo, da leve črte pri črki L ne more narisati en sam robot, saj drugi robot ne more narediti dveh zaporednih premikov naprej (torej se mora obrniti po prvem premiku naprej). Torej morata robota narediti vsaj en obrat – ali za izris preostanka leve črte ali za izris spodnje črte. Vendar pa, ko se izriše katerikoli del leve črte, robot ne more narediti nobenega drugega obrata kot le za 180 stopinj, saj ne sme risati levo ali desno. Črke L torej ne moremo izrisati, saj oba robota ne moreta narediti obrata.

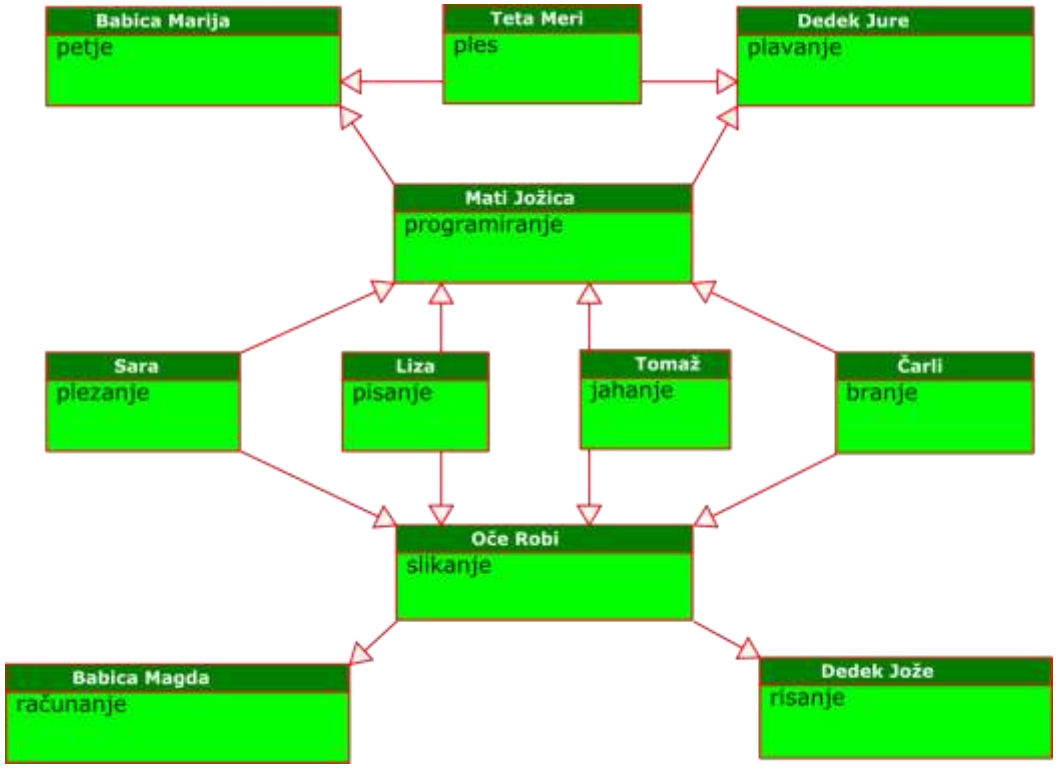
Računalniško ozadje

Navodila robotom za risanje so primer *algoritma*. Okoliščine v nalogi so nekoliko nenavadne, saj ukaze uporabljata dva robota, ki se razlikujeta v razumevanju posameznega ukaza. Zmožnost razumevanja posledic podanih ukazov tudi v bolj nenavadnih okoliščinah je pomemben del *algoritmičnega razmišljanja*.

Rodbinski talenti

1. – 4. letnik

Člani neke rodbine so talentirani. Vsak ima svoj poseben talent, nekaj pa jih podeduje: hči podeduje vse mamine talente, sin pa vse očetove. Diagram kaže rodbino; pri vsakem članu je zapisan tudi njegov posebni talent.



Kot vidimo, je mama Jožica po babici Mariji podedovala talent za petje in ima poleg tega tudi talent za programiranje. Liza po mami podeduje talent za petje in programiranje, poleg teh dveh pa ima še talent za pisanje. Lizini talenti so torej pisanje, programiranje in petje.

Katera trditev je pravilna?

A. Sarini talenti so branje, programiranje in petje.
B. Tomaž podeduje talent za računanje po babici Magdi.
C. Teta Meri ima talent za plesanje in plavanje.
D. Tomaževi talenti so jahanje, risanje in slikanje.

Rešitev

Trditev D je pravilna, saj je Tom podedoval talent za risanje od svojega dedka in za slikanje od očeta. Trditev A ni pravilna, ker Sara po bratu Čarliju ne more dedovati talenta za branje. Trditev B ni pravilna, saj Tomaž ne deduje po babici. Trditev C ni pravilna, ker teta Mari ne deduje plavanja po svojem očetu.

Računalniško ozadje

Dedovanje je pomemben koncept objektno orientiranega programiranja. Dedovanje nam omogoča logično povezovanje in razširjanje podatkovnih struktur in njihovih algoritmov.

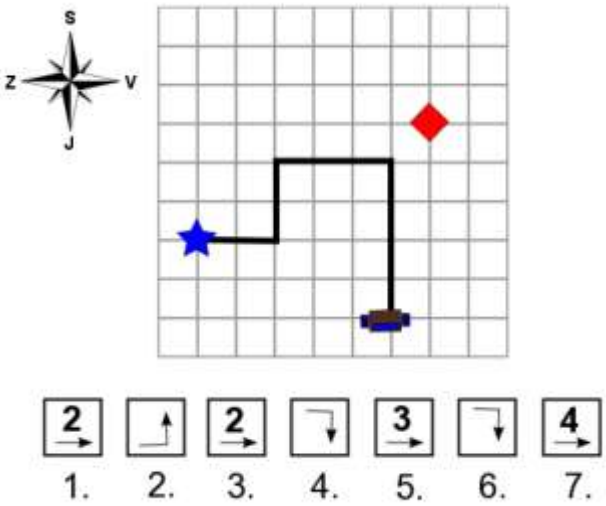
Robota nadziramo s tremi tipkami.

Robot se obrne levo

Robot se obrne desno

Robot se za X enot premakne v smeri, v katero je obrnjen

Robot je v začetku pri modri zvezdi, obrnjen proti vzhodu. Janez želi robota premakniti do rdečega diamanta, zato je pritisnil sedem tipk v zaporedju, prikazanem na sliki. Na žalost je po pomoti pritisnil dve tipki preveč, zato robot ni dosegel cilja.



Kateri dve tipki je pritisnil pomotoma?

A. prvega in drugega
B. prvega in četrtega
C. tretjega in četrtega
D. drugega in šestega

Rešitev

Pravilni odgovor je C

Robot se mora premakniti za 3 enote proti severu in 6 enot proti vzhodu. V celotnem zaporedju ukazov je proti severu obrnjen samo po drugem ukazu in preden se ponovno obrne. Zato mora biti četrti ukaz napačen. Prav tako mora biti napačen tretji ukaz, saj bi se sicer premaknil 5 enot proti severu, kasneje pa se nikoli ne obrne proti jugu, da bi izničil odvečen premik.

Če preverimo, kaj se zgodi, ko uporabimo ukaze označene z 1, 2, 5, 6 in 7, vidimo, da to zaporedje dejansko pripelje robota od modre zvezde do rdečega diamanta.

Računalniško ozadje

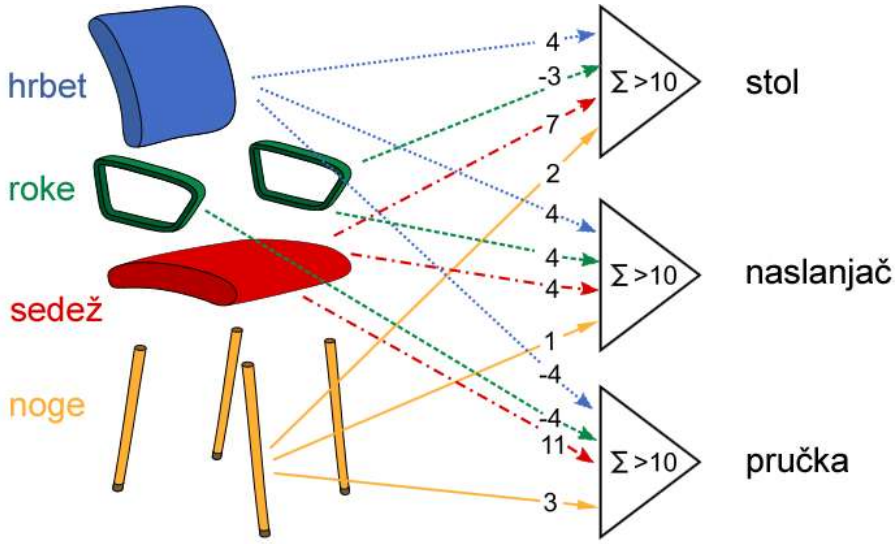
Računalnik je stroj, ki ga programiramo podobno, kot upravljamo robota, le da imamo na voljo veliko več različnih ukazov. Iz zaporedja ukazov sestavljamo programe. Programi vsebujejo od samo nekaj, pa do več tisoč ukazov.

Pri pisanju programov pogosto naredimo napako, ki ji v računalniškem žargonu pravimo “hrošč”, procesu iskanja in odpravljanja programskih napak pa razhroščevanje. Naloga postavi reševalca v vlogo programerja, ki išče napako v programu.

Stol ali naslanjač?

1. – 4. letnik

Bobrovski raziskovalni center za umetnost lenobe je razvil razpoznavni sistem za počivalnike, ki temelji na treh »nevronih«. Ti preučijo dele, ki jih ima posamezen predmet, ter seštevajo ustrezne točke, če ima predmet naslanjalo za hrbet, naslanjalo za roke, sedež ali noge (število točk za posamezne dele za vsako vrsto počivalnika je razvidno s spodnje slike).



Nevroni razpoznavajo *stole*, *naslanjače* (z naslanjali za roke) in *pručke* (brez naslonila), kadar pri enem izmed nevronov vsota preseže vrednost 10 in je pri drugih dveh nevronih ta vsota manjša ali enaka 10.

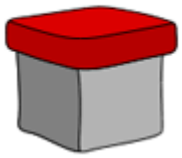
Primer: predmet na desni sliki ima sedišče, noge in naslanjalo za hrbet, nima pa naslanjal za roke, kar da skupaj 13 točk na prvem nevronu, 9 na drugem in 10 na tretjem. Torej bo predmet prepoznan kot *stol*.

Katerega od naslednjih predmetov razpoznavni sistem **ne bo prepoznal**?

A. (hrbet, roke, sedišče)

B. (hrbet, sedišče)

C. (sedišče)



D. (roke, sedišče, noge)

Rešitev

Pravilni odgovor je D.

Predmet pod A ima naslednje lastnosti: hrbet, roke, sedišče in brez nog. Uteži za posamezne vrste počivalnikov so naslednje: stol --> 4-3+7 = 8; naslanjač --> 4+4+4 = 12; pručka --> -4-4+11 = 3. Torej bo predmet A prepoznan kot *naslanjač*.

Predmet pod B ima naslednje lastnosti: hrbet, sedišče, brez rok in brez nog. Uteži za posamezne vrste počivalnikov so naslednje: stol --> 4+7 = 11; naslanjač --> 4+4 = 8; pručka --> -4+11 = 7. Torej bo predmet B prepoznan kot *stol*.

Predmet pod C ima naslednje lastnosti: sedišče, brez hrbta, brez rok in brez nog. Uteži za posamezne vrste počivalnikov so naslednje: stol --> 7; naslanjač --> 4; pručka --> 11. Torej bo predmet C prepoznan kot *pručka*.

Predmet pod D pa ima naslednje lastnosti: sedišče, roke, noge, brez hrbta. Uteži za posamezne vrste počivalnikov so naslednje: stol --> -3+7+2 = 6; naslanjač --> 4+4+1 = 9; pručka --> -4+11+3 = 10. Nobena od alternativ ni dobra za ta tip pohištva, zato predmet D ne bo prepoznan.

Računalniško ozadje

Prepoznavanje predmetov, ki spadajo v neko skupino (razred, množica stvari), ni enostavna naloga. Pomislimo samo na to, kaj je na primer vrtnica. Lastnosti, ki pomagajo ločiti vrtnico od bobra, nam navadno niso v pomoč pri ločevanju vrtnice od tulipana. Vendar pa je prepoznavanje veliko bolj enostavno, kadar imamo vnaprej določene skupine predmetov, ki jih prepoznavamo. Tak poenostavljen problem se imenuje *klasifikacija*. Razpoznavni sistem v nalogi je načrtovan tako, da vsak predmet razvrsti v eno izmed štirih skupin: stoli, naslanjači, pručke in ostalo. *Nevroni* so enostavne komponente, ki izračunajo vsoto in se *aktivirajo*, če je rezultat večji od določenega praga (pravzaprav je ta enostaven model presenetljivo podoben temu, kar biologi dejansko vedo o delovanju nevronov v možganih). Številke, ki jih nevroni seštevajo, imenujemo *uteži*, saj določajo pomembnost vsake od lastnosti vhodnih podatkov za nalogo klasifikacije. Tako je na primer sedišče zelo pomembna lastnost pručk, noge pa so skoraj nepomembne za naslanjače.

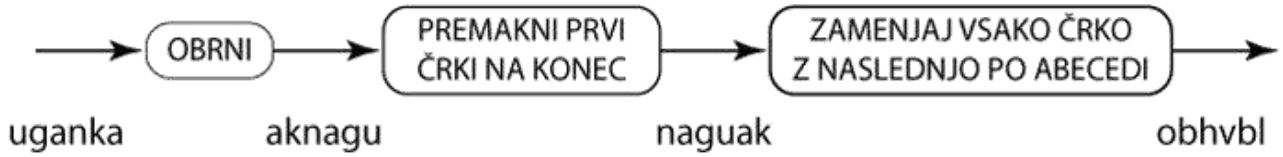
V splošnem je nevron jedrnat način izražanja sicer kompleksnega pravila. Ker so vsi vhodni podatki binarne (resnično/neresnično oziroma 1/0) lastnosti, lahko zapišemo to pravilo v obliki tabele, v kateri predstavimo vse možnosti. Tako bi za nevron, ki prepoznava stole, lahko zapisali:

	hrbet	*roke*	*sedišče*	*noge*	*vsota*	*klasifikacija*
uteži	**4**	**-3**	**7**	**2**		
	0	0	0	0	0	ostalo
	0	0	0	1	2	ostalo
	0	0	1	0	7	ostalo
	0	0	1	1	9	ostalo
	0	1	0	0	-3	ostalo
	0	1	0	1	-1	ostalo
	0	1	1	0	4	ostalo
	0	1	1	1	6	ostalo
	1	0	0	0	4	ostalo
	1	0	0	1	6	ostalo
	1	0	1	0	11	stol
	1	0	1	1	13	stol
	1	1	0	0	1	ostalo
	1	1	0	1	3	ostalo
	1	1	1	0	8	ostalo
	1	1	1	1	10	ostalo

Iz tabele je razvidno, da nevron za stole prepoznava predmete kot stole le v primeru, če imajo naslonjalo za hrbet in sedišče ter nimajo naslanjal za roke, lahko pa imajo tudi noge (ni pa nujno).

Ne boš prebral!

Peter in Petra si pošiljata sporočila, ki jih skrijeta po postopku na sliki.

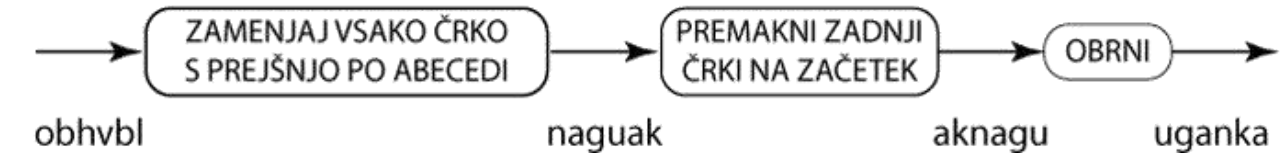


Primer vidimo pod sliko: beseda **uganka** se spremeni v **obhvbl**.

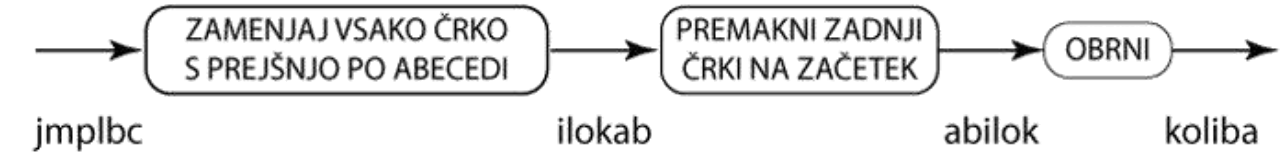
Peter je Petri poslal sporočilo: **jmplbc**. Katero besedo ji sporoča?

Rešitev

Besedo preberemo tako, da opravimo obratne korake v obratnem vrstnem redu:



Kaj nastane iz besede jmplbc?



Računalniško ozadje

Tudi nekateri resnični postopki šifriranja morajo ob dešifriranju ponoviti vse operacije v obratnem vrstnem redu. Takšni postopki so običajno “simetrični”: obe strani uporabljata enak (le morda obratni) postopek in enak ključ.

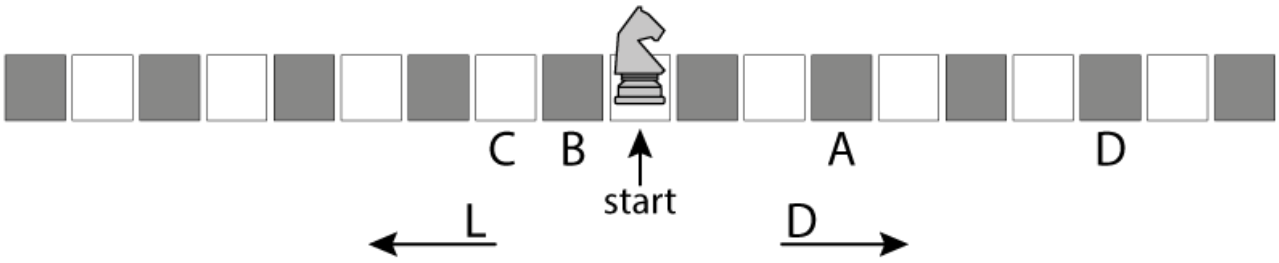
Postopek iz te naloge je seveda veliko preveč preprost, da bi bil praktično uporaben.

Konj

državno, 1. – 4. letnik

Enodimenzionalni šah je dolgočasna igra za enega igralca. Edina figura je šahovski konj, ki izmenično skače desno in levo. Igra traja pet potez. Konj mora natančno enkrat skočiti za pet polj, enkrat za štiri, enkrat za tri, za dve in za eno. Vrstni red razdalj je lahko poljuben. Tako lahko v neki igri konj najprej skoči levo za 3, nato desno 4, levo 1, desno 5, levo 2.

Na katerem izmed spodnjih polj ne more končati igre?



Rešitev

Označimo premike desno s pozitivnimi števili, levo z negativnimi.

- Na polju A bo končal z zaporedjem + 5 – 4 + 3 – 2 + 1 = 3.
- Na polju B bo končal z zaporedjem + 1 – 5 + 2 – 3 + 4 = -1.
- Na polju D bo končal z zaporedjem + 5 – 3 + 4 – 1 + 2 = 7.
- Na polju C ne more končati. Seštevali in odštevali bomo števila 1, 2, 3, 4, 5. Ker so tri od teh števil liha, bo rezultat vedno lih. Konj bo zato vedno končal na temnem polju.

Računalniško ozadje

Pri reševanju naloge se je splačalo poteze nadomestiti s številkami. S tem smo algoritmični problem prevedli na matematičnega – in ko to storimo, moremo uporabiti vse številne trike, ki smo se jih naučili pri matematiki.

Problem je zanimiv tudi, ker ga je potrebno le malo spremeniti in postane (verjetno) nerešljiv: če dovolimo konju, da se večkrat zapored premakne v isto smer, niti z najhitrejšimi računalniki ne moremo odgovoriti na vprašanje, ali lahko konj konča na določenem polju – čim postane število potez v igri nekoliko večje. To vprašanje sodi med tako imenovane NP-polne probleme, za katere še nihče ni našel hitrega postopka reševanja, niti ni še nihče dokazal, da ne obstaja.

Pomembni kosi

državno, 1. – 4. letnik

Informacije v človeškem genomu so zakodirane s štirimi komponentami: adenin (A), citozin (C), gvanin (G) in timin (T). Genski zapis tako predstavimo z zaporedjem znakov A, C, G in T, kot je na primer zaporedje CAGGAGGAT.

Taka zaporedja so lahko zelo dolga. Raziskovalce zanimajo predvsem bolj pomembni kosi. Pomembnost posameznega kosa lahko izračunamo na naslednji način:

pomembnost = dolžina kosa + število pojavitev kosa v zaporedju

Vzemimo zaporedje CAGGAGGAT. V tem zaporedju je pomembnost kosa AGGA 6: kos je dolg 4 znake in se pojavi 2-krat. Kos G je manj pomemben, saj je njegova vrednost le 5: v zaporedju se sicer pojavi 4 krat, a je dolg le 1 znak.

Kateri je najpomembnejši kos zaporedja **CATTGTTGTTGCATT**?

A. T
B. TT
C. TTG
D. GTTG
E. TTGTT
F. TTGTTG

Rešitev

Pravilni odgovor je A (kos T). Kos T ima vrednost 9 (dolžina 1 in frekvenca 8), v zaporedju pa ni nobenega drugega pomembnega kosa, ki bi imel vrednost večjo ali enako 9.

V danem zaporedju ima kos TTGTTG vrednost 8 (2 pojavitvi, dolžina 6). Kos TTGTT ima vrednost 7 (2 pojavitvi, dolžina 5). Kosi z vrednostjo 6, so CATT, GTTG, TGTT in TTGT (vsi imajo dolžino 4 in frekvenco 2), pa tudi TTG (dolžina 3 in frekvenca 3) ter TT (dolžina 2 in frekvenca 4).

Računalniško ozadje

Genetiki so odkrili, da celice izvajajo »program«, ki je shranjen v genih. Iz te perspektive je genetika v svojem bistvu pravzaprav kodiranje in procesiranje informacij. Že v začetku 90. let so začeli raziskovalci iz biologije in računalništva sodelovati, s čimer se je začelo razvijati novo področje: *bioinformatika*. Razvili so veliko novih načinov predstavitve in obdelave bioloških informacij s pomočjo računalnikov. To ni vplivalo le na biologijo, temveč tudi na farmacijo in medicino. Analiza (genetskega) zaporedja pa je še vedno eden od pomembnih vidikov bioinformatike.

Vreme

Janez namerava preživeti jutrišnji dan na plaži, vendar le, če bo med 13. in 19. uro vsaj tri ure sončno. V datoteki ima podatke o vremenski napovedi za vsako uro naslednjega dne. Datoteko sestavlja 24 vrstic, vsaka vrstica ustreza eni uri dneva, od 00:00 – 01:00 do 23:00 – 24:00, poleg ure pa je zapisana še napoved: *sončno*, *oblačno*, *deževno* ali *sneg*.

Janez lahko uporabi naslednje ukaze:

- ***SAMO b*** vrne le vrstice vhoda, ki vsebujejo besedo *b*;
- ***PRVIH n*** vrne le prvih *n* vrstic vhoda;
- ***ZADNJIH m*** vrne le zadnjih *m* vrstic vhoda;
- ***PREŠTEJ*** prešteje vrstice na vhodu in vrne njihovo število.

Če Janez uporabi | kot ločilo, lahko poljubno kombinira zgornje ukaze v zaporedje ukazov. Pri tem je izhod kateregakoli ukaza v zaporedju uporabljen kot vhod za naslednji ukaz. Vhod prvega ukaza je vedno datoteka z vremensko napovedjo.

Katero zaporedje ukazov mora uporabiti, da se odloči, ali gre na plažo ali ne?

A. PRVIH 19 | ZADNJIH 6 | SAMO sončno | PREŠTEJ
B. SAMO sončno | PRVIH 19 | ZADNJIH 6 | PREŠTEJ
C. PRVIH 20 | ZADNJIH 6 | SAMO sončno | PREŠTEJ
D. ZADNJIH 20 | PRVIH 6 | SAMO sončno | PREŠTEJ
E. PRVIH 19 | SAMO sončno | PRVIH 6 | PREŠTEJ

Rešitev

Pravilni odgovor je A, **PRVIH 19 | ZADNJIH 6 | SAMO sončno | PREŠTEJ**.

- **PRVIH 19** obdrži le vrstice, ki ustrezajo uram do 19.00.
- **ZADNJIH 6** obdrži zadnjih šest izmed teh vrstic, torej vrstice za ure od 13.00 do 19.00.
- **SAMO sončno** obdrži vrstice, ki napovedujejo sončno vreme
- **PREŠTEJ** jih prešteje.

Če bo številka na izhodu večja ali enaka 3, se bo torej bober Janez odločil za odhod na plažo.

B najprej izbere vse ure, za katere je napovedano sonce. Nato vzame prvih 19 in potem zadnjih 6 od teh vrstic. Rezultat bo število vseh sončnih ur, vendar največ 6.

Odgovor C je skoraj pravilen, vendar računa do 20.00.

V odgovoru D **ZADNJIH 20** obdrži ure od 4.00 do 24.00, **PRVIH 6** pa prvih 6 od teh. Izvedeli bomo število sončnih ur od štirih zjutraj do desetih dopoldan. Neuporabno.

V odgovoru E najprej izberemo vse ure do 19.00. Nato med njimi izberemo **SAMO sončne**. Če jih je več kot 6, obdržimo le **PRVIH 6** in jih **PREŠTEJ**emo. Rezultat je torej število sončnih ur pred sedmo zvečer, a največ šest.

Računalniško ozadje

Več računalniških nalog lahko učinkovito rešimo z zaporednim izvajanjem posameznih ukazov (ali korakov procesiranja), od katerih vsak iz podatkov svojega vhoda izloči ali spremeni del podatkov. Tak pristop nam omogoča, da se osredotočimo na posamezne enostavne naloge (tiste, ki jih izvedemo v posameznem koraku procesiranja) namesto reševanja enega samega, bolj kompleksnega, splošnega problema.

Recimo, da so podatki shranjeni v datoteki napoved.txt. Rešitev naloge lahko v računalnikih z Linuxom ali Mac Os X-om, ki imajo spodobno ukazno lupino in pripadajoče drobne programe, v resnici dobimo takole:

```
cat napoved.txt | head -n 19 | tail -n 6 | grep sončno | wc -l
```

cat napoved.txt prebere datoteko. Ukaza **head** in **tail** ustrezata našim **PRVIH** in **ZADNJIH**, le številko moramo podati z argumentom **–n**. Ukaz **grep** ustreza našemu **SAMO**, ukaz **wc –l** pa prešteje vrstice.

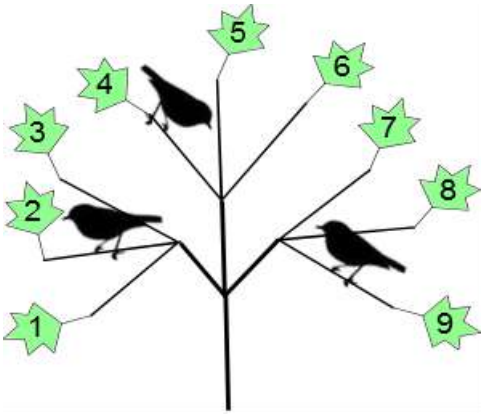
Pravega računalnikarja prepoznamo po tem, da stresa te ukaze iz rokava.

Ptiči

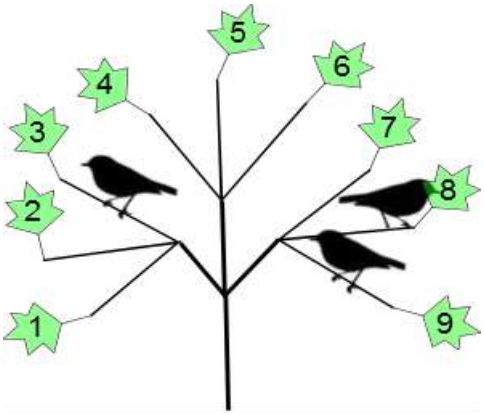
Imamo štiri drevesa z devetimi vejami. Na vsakem sedijo trije ptiči, kot kažejo slike. Vsake tri sekunde dva ptiča zletita na sosednjo vejo od tiste, na kateri sedita (vendar ne z veje 9 na 1 ali obratno).

Na enem izmed dreves so se uspeli vsi ptiči zbrati na prvi veji. Na katerem?

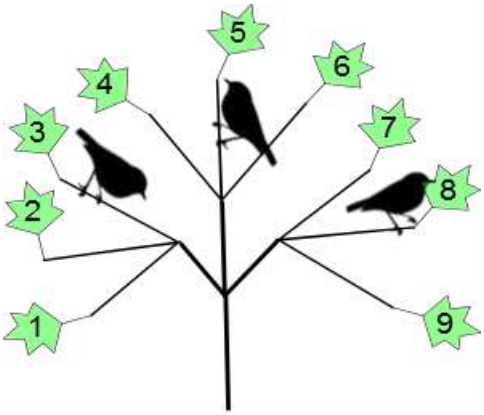
A.



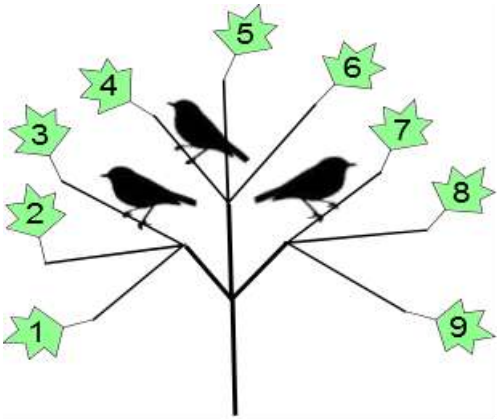
B.



C.



D.



Rešitev

Pravilni odgovor je A.

Opazujmo vsoto številk vej, na katerih sedijo ptiči. Ta je lahko soda ali liha. Ko gresta dva ptiča hkrati na sosednjo vejo, vsota ostane soda ali liha. Na drevesu A je vsota (2 + 4 + 9 = 15) liha in zato bo tudi vedno ostala liha. Vsote na ostalih drevesih so sode: 3 + 8 + 9 = 20, 3 + 5 + 8 = 16 in 3 + 4 + 7 = 14. Zato bodo vedno ostale sode.

Če se vsi ptiči zberejo na prvi veji, bo vsota 1 + 1 + 1 = 3, torej liha. Na drevesih B, C in D se to ne more zgoditi.

Vendar s tem še nismo dokazali, da se na drevesu A res lahko zberejo na isti veji. Morda tudi tam to iz kakega razloga ni mogoče?

S tem bomo opravili tako, da poiščemo rešitev. Ena od rešitev za drevo A je naslednja: (2, 4, 9) $\rightarrow$ (2, 5, 8) $\rightarrow$ (2, 6, 7) $\rightarrow$ (1, 6, 6) $\rightarrow$ (1, 5, 5) $\rightarrow$ (1, 4, 4) $\rightarrow$ (1, 3, 3) $\rightarrow$ (1, 2, 2) $\rightarrow$ (1, 1, 1).

Računalniško ozadje

V računalništvu je ena od ključnih metod za reševanje problemov, da poiščemo dele problema, ki se ne spremenijo, če uporabimo v problemu definirane veljavne korake. To imenujemo *invarianta*; pogosto jo uporabljamo na primer pri zankah. V našem primeru je invarianta parnost števila, ki ga lahko enostavno izračunamo: vsota indeksov vej, na katerih sedijo ptiči. To je veliko hitrejši način, kot če preverimo vse možne premike.

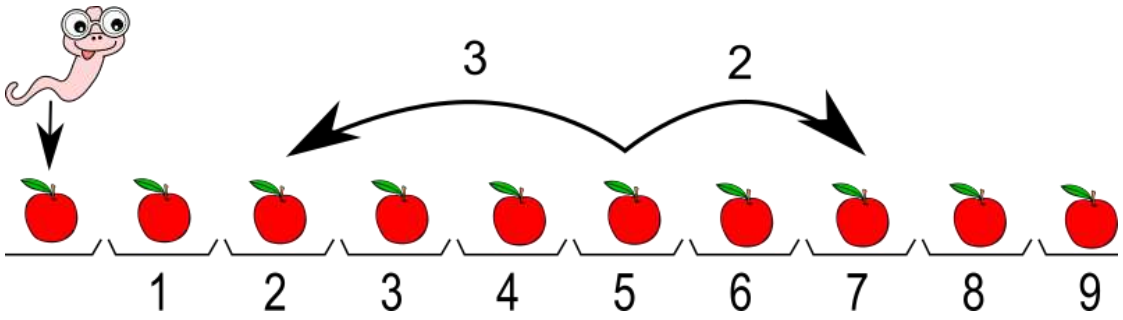
Koncept parnosti lahko uporabimo tudi v drugih primerih, kot je na primer preverjanje, ali se je prenos podatkov pravilno izvedel (na primer za branje črtnih kod ali za prenos podatkov preko interneta) ali pa zagotavljanje, da shranjeni podatki niso okvarjeni.

Črv

državno, 1. – 4. letnik

Ker imajo vse živali na tem tekmovanju imena, ga ima tudi črv. Tomaž.

Kot vsi jabolčni črvi tudi jabolčni črv Tomaž žre jabolka. Za razliko od ostalih pa Tomaž skače s krožnika na krožnik: vedno dva desno ali tri levo. Pri tem nikoli ne skoči na prazen krožnik in vedno poje jabolko na krožniku, na katerega je skočil – ker je lepo vzgojen in vedno poje vse, kar je na krožniku.



Danes bo začel na prvem izmed desetih krožnikov. Pojedel bo vsa jabolka. Katerega bo pojedel zadnjega?

Rešitev

Jabolko 8, to je, drugo jabolko z desne.

Tomaž lahko pobere vseh deset jabolk z naslednjim zaporedjem skokov: 2, 4, 1, 3, 5, 7, 9, 6, 8.

To je edino zaporedje skokov, ki omogoča Tomažu, da skoči na vsak krožnik in tako pobere vsa jabolka. Zakaj? Na začetku mora Tomaž skakati v desno na krožnika 2 in 4, saj bi v nasprotnem primeru (s skokom v levo) skočil levo od prvega krožnika. Nato mora skočiti na krožnik 1, saj lahko nanj pride le s krožnika 4, na katerega pa se kasneje ne bo mogel več vrniti. Nato bo štirikrat skočil desno, saj levo ne sme, ker tam ni (polnih) krožnikov. Tako obdela krožnike 1, 3, 5, 7 in 9. Z devet gre levo na 6 in nato desno na 8.

V nobenem koraku ni mogel skočiti drugače, saj so mu to preprečevala pravila. Le tedaj, ko je bil na krožniku 4, bi mu pravila dovolila skočiti desno, vendar smo tedaj videli, da mora nujno levo, sicer ne bo nikoli prišel do krožnika 1.

Računalniško ozadje

Eden od načinov, kako pridemo do rešitve tega problema, je, da upoštevamo vse možne razvrstitve krožnikov ter poiščemo tiste, ki vključujejo le veljavne skoke. Vsaka od možnih razvrstitev se imenuje *permutacija* in vseh skupaj je zelo veliko (362880), predvsem pa vedno hitreje narašča, ko dodajamo nova jabolka. Zato tak pristop, imenujemo ga tudi *iskanje z grobo silo* ali *izčrpno iskanje* (ali angleško *brute-force* oziroma *exhaustive search*), zahteva veliko časa.

Nekoliko drugačen pristop je, da gradimo rešitev za en krožnik naenkrat. Ko ugotovimo, da smo se zaplezali, se vrnemo nekaj korakov nazaj. Tak pristop imenujemo *vračanje* (ali angleško *backtracking*). Če lahko izločimo veliko permutacij že na začetnih korakih iskanja, bomo rešitev našli veliko hitreje. Tako bližnjico imenujemo *rezanje* (ali angleško *pruning*). O tem ti lahko še več kot tvoj učitelj računalništva pove tvoja mama, če igra Sudoku, saj tam počne natančno isto: postavlja številke, o katerih je prepričana, kadar ima več možnosti, pa se odloči za eno izmed njih in se nato vrača, če se odločitev izkaže za napačno.

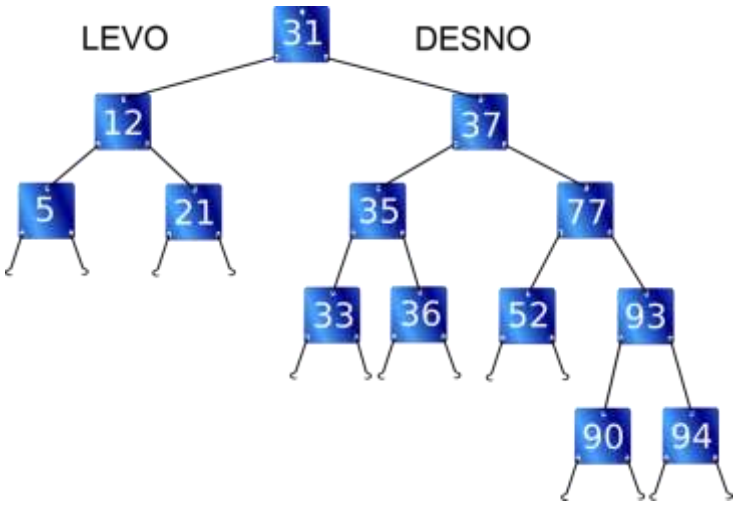
Opisan problem lahko obravnavamo kot graf. Krožniki so vozlišča, dve vozlišči pa sta povezani, če lahko Tomaž skoči z enega na drugega. Naša naloga je torej poiskati pot v grafu, ki vsako vozlišče obišče natanko enkrat. Tako pot imenujemo *Hamiltonova pot*. V splošnem je zelo težko poiskati tako pot skozi graf. Vendar pa je v našem primeru graf dovolj majhen in ima tudi določene posebne lastnosti.

Splošen problem iskanja Hamiltonove poti je *NP-poln problem*, kar pomeni, da spada med tiste zelo pomembne probleme, za katere še ne obstajajo učinkovite rešitve. Zanimivo je tudi to, da če bi nekdo našel učinkovito rešitev za enega od teh pomembnih NP-polnih problemov, bi imeli s tem tudi učinkovito rešitev za vse ostale.

Iskalno drevo

državno, 1. – 4. letnik

Iz oštevilčenih kartic smo sestavili strukturo, ki jo prikazuje slika.



Vsaka oštevilčena kartica je narejena tako, da ima na vrhu eno luknjico, spodaj pa dve žički, ki ju lahko pritrdimo na luknjice drugih kartic. Vsaka kartica ima izpisano neko celo število N.

- Če kartico s številko N povežemo z levo žičko na neko drugo kartico, mora biti številka na njej in tudi vse številke kartic, ki so povezane z njo, manjša od N.
- Če kartico s številko N povežemo z desno žičko na neko drugo kartico, mora biti številka na njej in tudi vse številke kartic, ki so povezane z njo, večja od N.

Nekatere oštevilčene kartice imajo proste žičke. Koliko kartic lahko še dodamo (neposredno) na te proste žičke, da razširimo strukturo po opisanih pravilih?

Rešitev

Imamo 14 prostih žičk, na katere lahko obesimo 11 kartic. Na žičke kartice s številko 36 ne moremo obesiti drugih kartic (sicer bi kršili zgornja pravila). Številka kartice na levi žički mora biti namreč večja od 35 in manjša od 36, celo število s tema lastnostma pa ne obstaja. Številka kartice na desni žički pa bi morala biti večja od 36 in manjša od 37 – tudi tako celo število ne obstaja. Podobno lahko tudi na kartico s številko 94 dodamo le kartico, povezano z desno žičko.

Računalniško ozadje

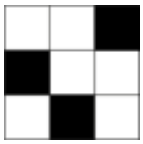
Struktura oštevilčenih kartic se imenuje *binarno iskalno drevo*. Na pogled je podobno pravemu drevesu, če ga obrnemo na glavo. Ime binarno (latinska beseda *bis* pomeni *dvakrat*) pa pove, da ima vsaka kartica natanko dve žički in tako lahko nanjo povežemo največ dve drugi številski kartici. Binarna iskalna drevesa se uporabljajo za shranjevanje različnih podatkov, njihova prednost pa je v tem, da lahko shranjene podatke v drevesu poiščemo zelo hitro.

HB koda

državno, 1. – 4. letnik

Bobri so za kodiranje števil razvili hitro bobrovo kodo (ali krajše HB kodo). Koda je grafična in sestoji iz kvadratkov. Vsak kvadratek ima določeno vrednost. Vrednosti kvadratkov zapolnimo po vrsticah od spodaj navzgor, od desne proti levi. Vrednost kvadratka spodaj desno je 1, vsak naslednji kvadratek pa ima dvakratno vrednost predhodnega kvadratka. Slika na desni prikazuje vrednosti prvih štirih kvadratkov za HB kodo velikosti 3x3.

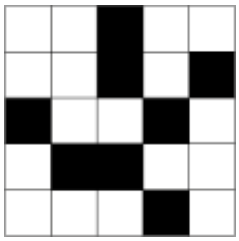
...	...	...
...	...	8
4	2	1



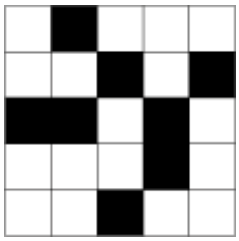
Število se s HB kodo zakodira tako, da se potemni nekatere kvadratke. Zakodirano število je vsota vrednosti vseh potemnjenih kvadratkov. Tako je, na primer, število, zakodirano z HB kodo 3x3, na levi sliki enako 2 + 32 + 64 = 98.

Katera od naslednjih rotacij HB kode velikosti 5x5 predstavlja največje zakodirano število?

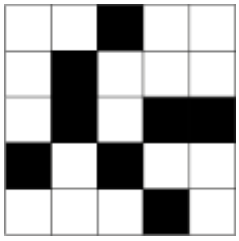
A.



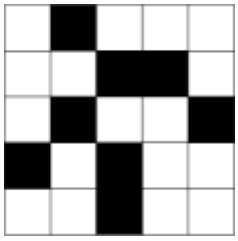
B.



C.



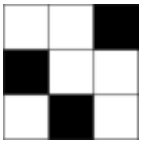
D.



Rešitev

Pravilni odgovor je D.

Poglejmo najprej zgornji primer z uporabo HB kode 3x3. Kodo:



lahko preuredimo v eno vrstico:

ali pa jo enostavno zapišemo z nizom ničel in enic kot 001100010. Vrednost, ki jo predstavlja ta HB koda, je enaka številu, ki ga dobimo s pretvorbo dvojiškega števila 001100010 v desetiško.

Sedaj pa si poglejmo še primerjavo dveh števil. Kako primerjamo 9-mestna desetiška števila? Katero število je večje, 934326958 ali 936715871? Odločimo se pri tretji števki z leve, ali ne? Prvi dve števki sta enaki, tretja pa se razlikuje ter tako določa, katero število je večje.

Podobno deluje tudi primerjava enako dolgih dvojiških števil: 010100111 je večje kot 010011010. Obe števili se začneta z 010, a se razlikujeta v četrtem bitu: drugo število je večje, ker ima četrti bit enak 1.

Vrnimo se k odgovorom na vprašanje naloge. Števili, ki ju predstavljata kodi pod A in C se začneta z 00; B in D pa se začneta z 01, torej sta večji. Če primerjamo še števili pod B in D, lahko vidimo, da je prva razlika na četrtem kvadratku v drugi vrstici: število D je torej večje, saj ima ta kvadratek potemnjen.

Računalniško ozadje

HB kode iz naloge so zelo podobne QR kodi. Namesto ene same številke lahko QR koda zakodira več številk. Hkrati pa je tudi nedvoumna: trije od štirih vogalov so označeni, tako da bo tudi v primeru, če med slikanjem kode obrnemo telefon na glavo, program vedel, katera rotacija je prava. Poleg tega je pomnožena tudi informacija v kodi, saj se tako poveča robustnost v primeru šuma v sliki.

Konjske dirke

državno, 3. – 4. letnik

Štefan ima 25 dirkalnih konjev. Vsi so zelo hitri, a Štefana zanima, kateri trije, po vrsti, so najhitrejši.

Ker nima štoparice, bo uporabil hipodrom, na katerem pa lahko sočasno tekmuje le pet konjev. Predpostavite lahko, da vsak konj vedno preteče progo v enakem času.

Katero je najmanjše število dirk, ki jih mora Štefan izvesti, da lahko določi, kateri konj je najhitrejši, kateri drugi najhitrejši in kateri tretji najhitrejši?

Rešitev

Potrebnih bo sedem dirk.

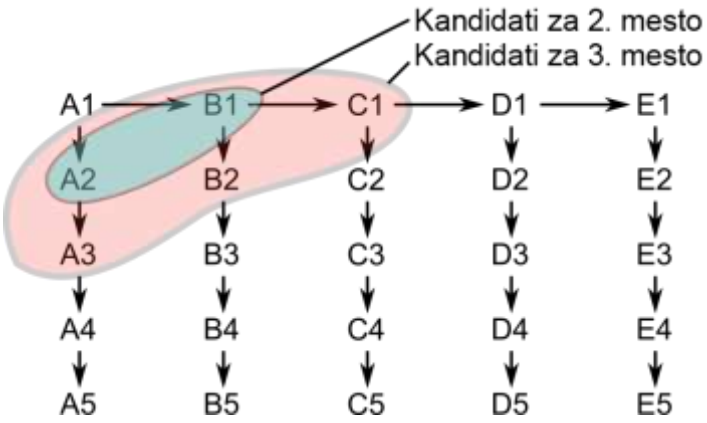
Najprej mora Štefan organizirati pet dirk, v katerih nastopi hkrati po pet konjev. Označimo konje, ki so nastopili v prvi dirki, z A1, A2, A3, A4 in A5, v zaporedju, kot so končali dirko (torej, konj A1 je bil na prvi dirki najhitrejši, A2 drugi in A5 najpočasnejši). Konje, ki so nastopili na drugi dirki, označimo z B1, B2, B3, B4 in B5. Podobno jih označimo tudi za preostale tri dirke.

V šesti dirki se nato pomerijo najhitrejši konji posameznih dirk (A1, B1, C1, D1 in E1), da določimo najhitrejšega konja med vsemi. Zaradi poenostavitve predpostavimo, da so konji to dirko končali v navedenem zaporedju, torej je A1 najhitrejši konj šeste dirke in E1 najpočasnejši konj šeste dirke (če to ne bi bilo res, bi opisana rešitev še vedno delovala, le črke bi morali spremeniti). Tako smo ugotovili, kateri konj je najhitrejši – to je A1.

Kateri konj bi lahko bil drugi najhitrejši? To je lahko ali konj B1 ali konj A2. Ker konja še nista tekmovala skupaj na dirki, ne vemo, kateri je hitrejši.

Kateri konj bi lahko bil tretji najhitrejši? Seveda, B1 ali pa A2. Poleg tega pa je tretji najhitrejši konj lahko tudi A3 (če bi bili najhitrejši trije konji A1, A2 in A3). Kandidat za tretjega najhitrejšega konja je tudi konj B2, če bi bili najhitrejši trije konji A1, B1 in B2. Nenazadnje pa je tretji najhitrejši konj lahko tudi C1, če bi bili najhitrejši trije konji A1, B1 in C1.

Zato mora Štefan organizirati še sedmo dirko, v kateri se pomerijo konji A2, A3, B1, B2 in C1, da lahko določi drugega in tretjega najhitrejšega konja.



Pogosta napaka pri določanju treh najhitrejših konjev je, da organiziramo pet dirk in zmagovalci posameznih dirk tekmujejo v šesti dirki za prvo, drugo in tretje mesto. V tem primeru so najhitrejši konji A1, B1 in C1, kar pa ni res v določenih primerih. Če na primer vsi trije najhitrejši konji slučajno sodelujejo v prvi dirki kvalifikacij (torej A1, A2 in A3), najhitrejši konj izloči drugega in tretjega najhitrejšega iz tekmovanja v šesti dirki.

Računalniško ozadje

Pri reševanju problema smo ustvarili del *mreže za razvrščanje*. Mreže za razvrščanje omogočajo hitro razvrščanje objektov z uporabo velikega števila *komparatorjev* – to so enote, ki primerjajo dva objekta naenkrat. Naša mreža v nalogi je bila nekoliko drugačna, saj smo lahko primerjali pet objektov (konjev) naenkrat, naš cilj pa ni bil razvrstiti cele črede, ampak poiskati le prve tri najhitrejše konje.

Mreže za razvrščanje so sicer odkrili že pred pol stoletja, a postajajo bolj zanimive šele v zadnjem času, ko imamo na voljo veliko strojne opreme za vzporedno računanje, kot je na primer tudi grafična kartica v tvojem računalniku, ki zmore sočasno narediti veliko stvari. Kljub temu pa še vedno ne obstaja algoritem za samodejno konstrukcijo optimalne mreže za sortiranje. Če si opisano nalogo rešil sam, boš morda ravno ti izumil tak algoritem!