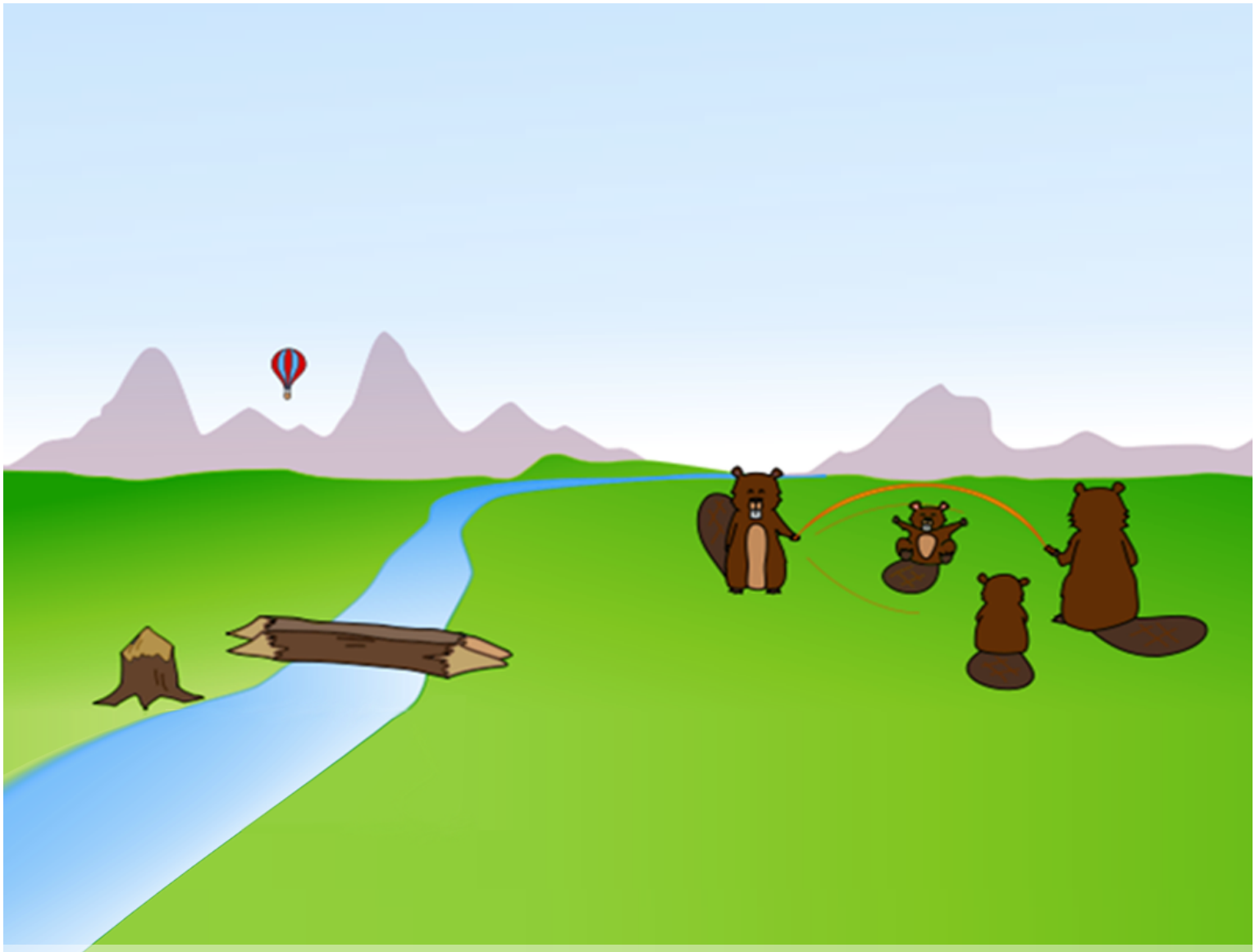


Bober 2015/16

Naloge in rešitve šolskega tekmovanja

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)

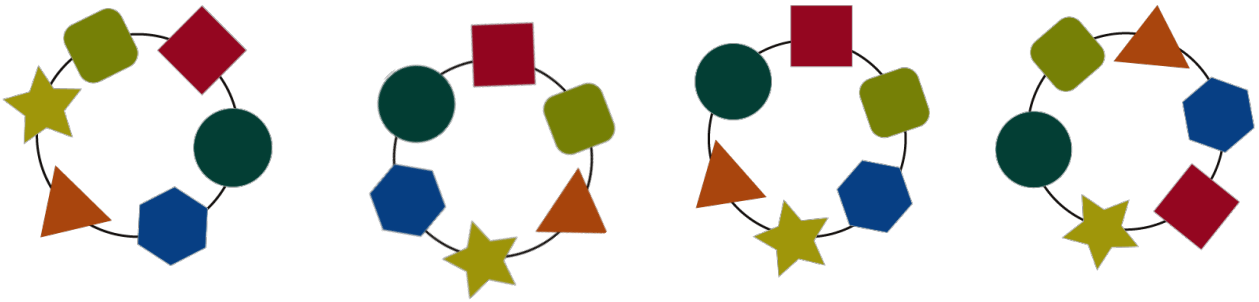
Kazalo nalog

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.

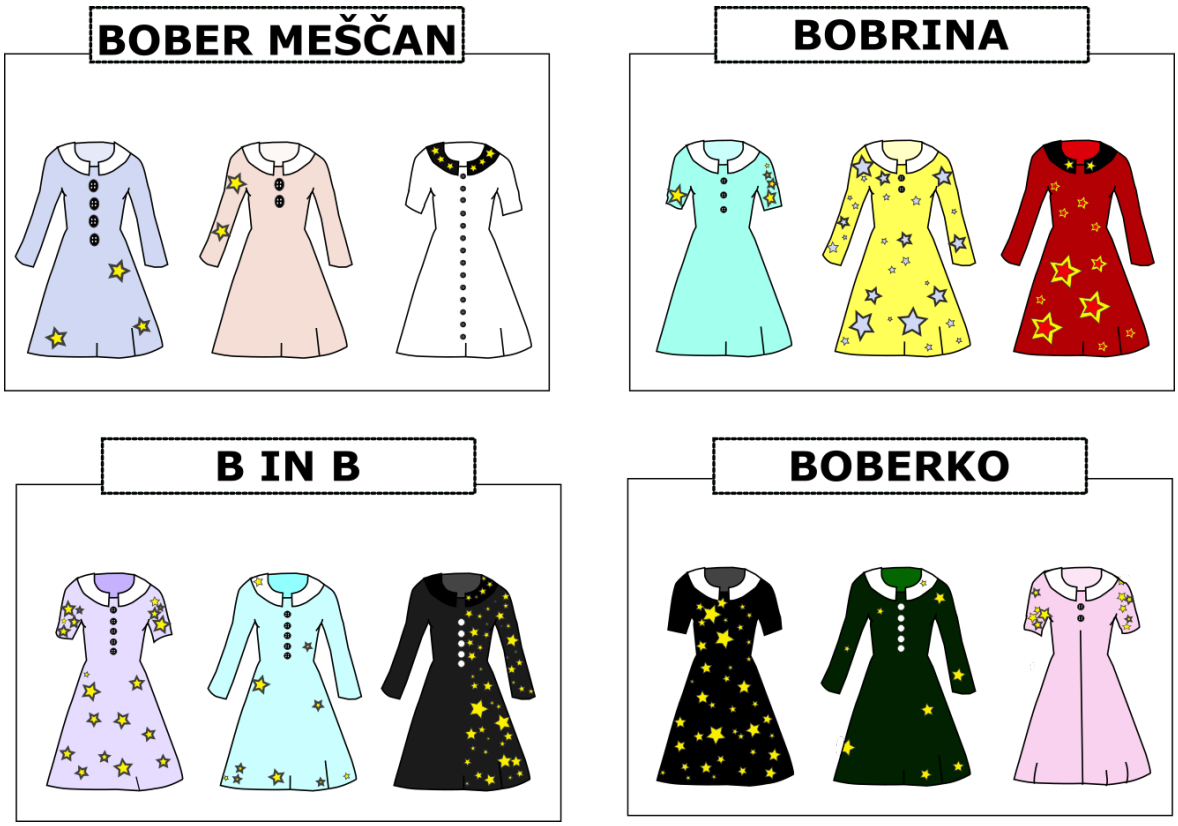
Sanjska obleka

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 DODAK PRTIČEK, SAJ VIDIMO, DA JE NAD KROŽNIČKOM.
- KROŽNIK JE POSTAVIL NA PRTIČEK,
- NOŽ JE NA KROŽNIKU.

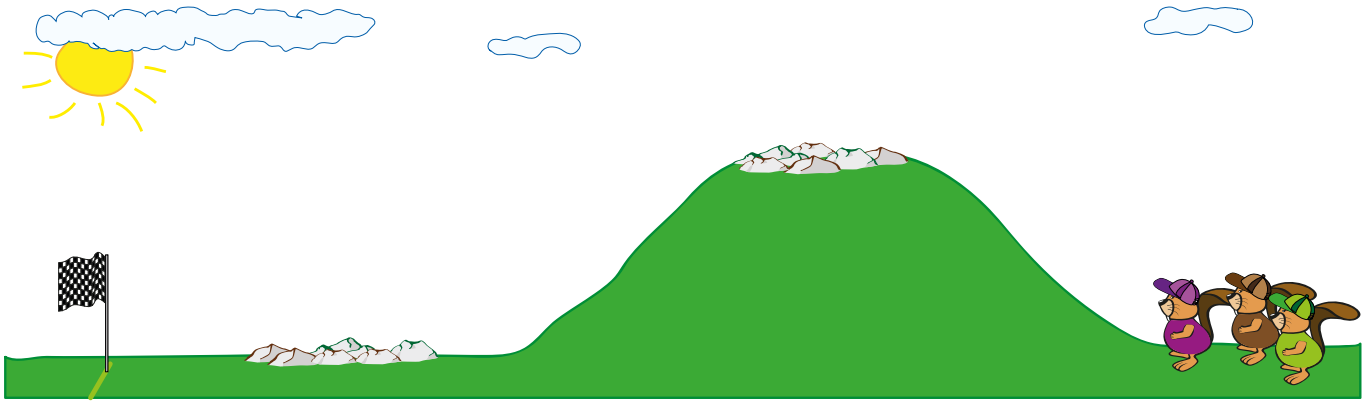
Računalniško ozadje

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.

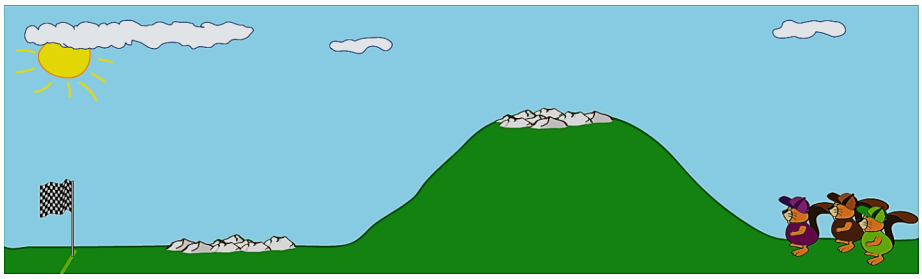
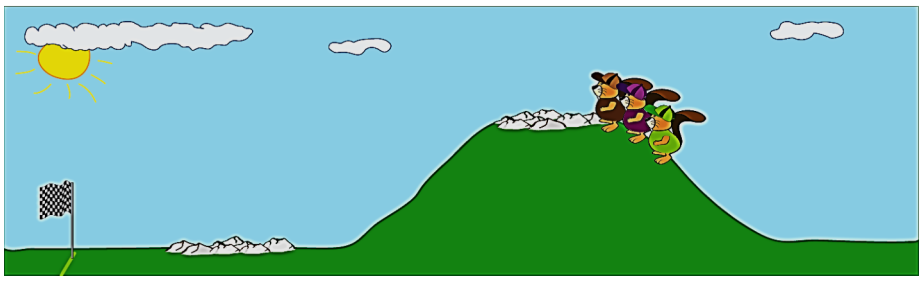
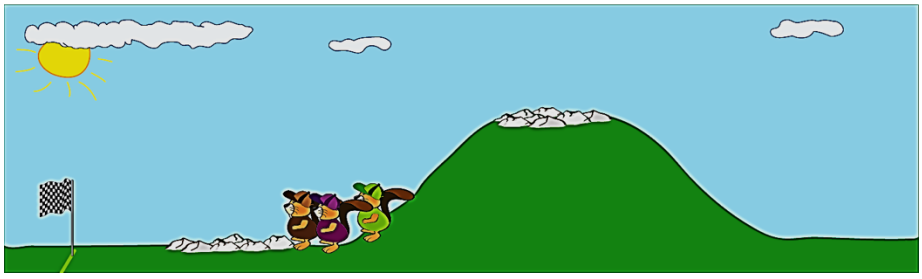
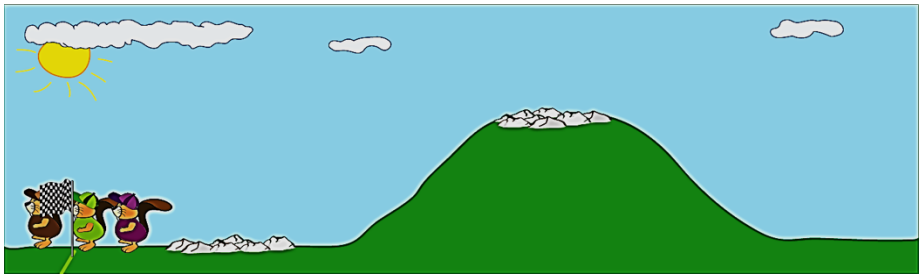
ROZIKA BO PREHITELA ENEGA TEKAČA PRED SEBOJ MED TEKOM NAVZDOL.

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

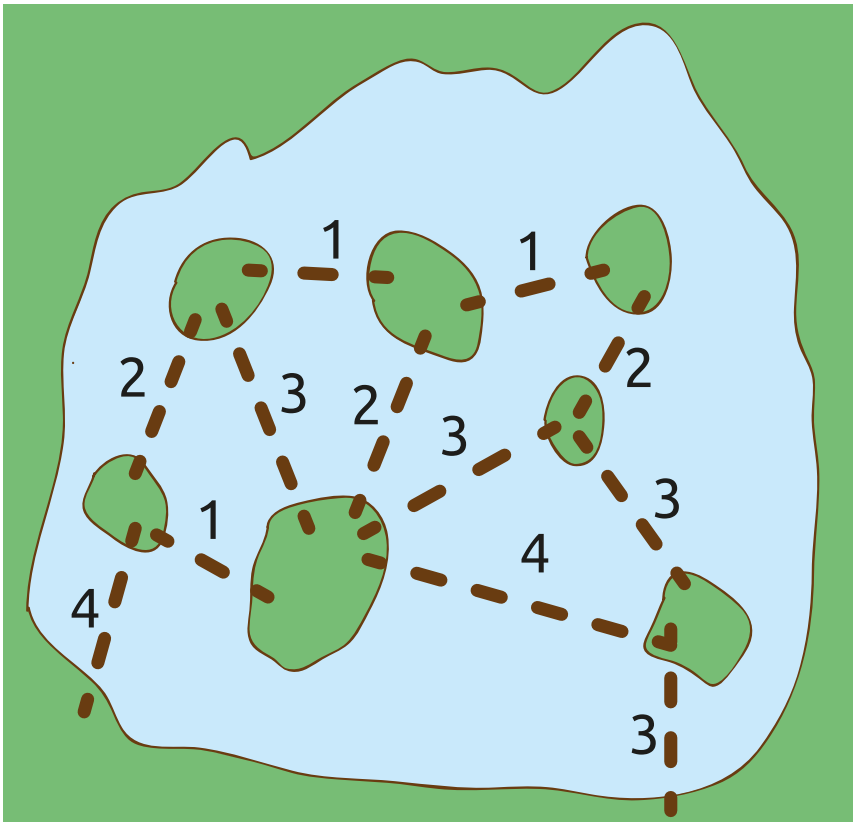
Programerji morajo pogosto pregledovati, kako deluje njihov program – posebej, če ne deluje pravilno. Takrat morajo slediti njegovemu delovanju vrstico za vrstico.

Naloga Čez drn in strm 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.

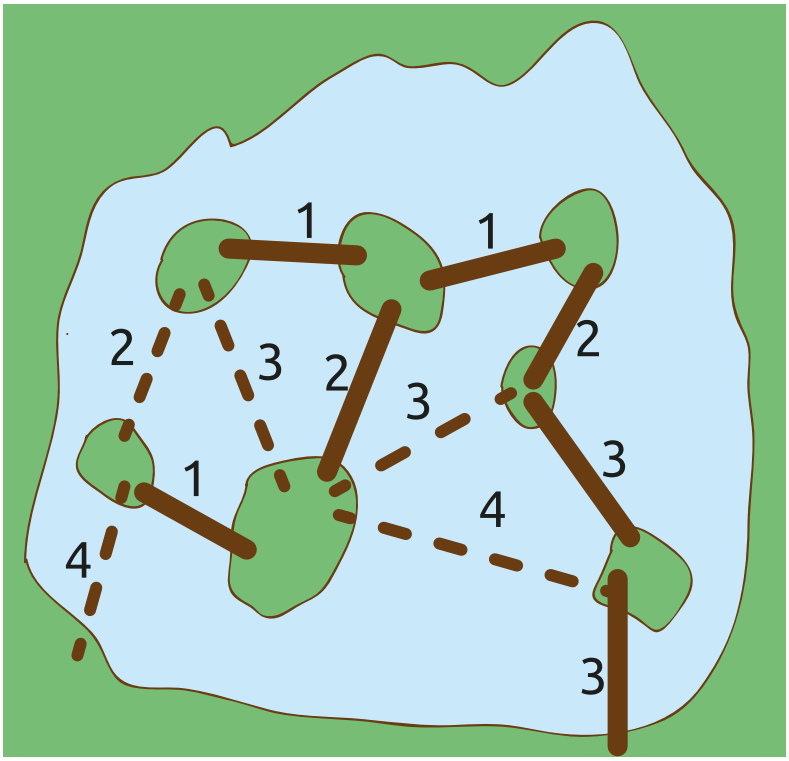


Najmanj koliko hlodov potrebujejo za izgradnjo mostov?

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

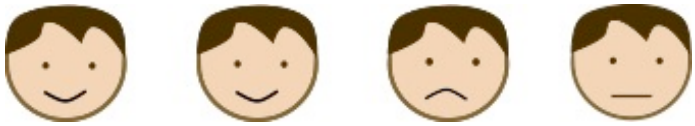
V nalogi rešujemo znan problem, ki se v različnih preoblekah pojavlja na najrazličnejših področjih matematike, računalništva in drugje. Imenujemo ga problem *iskanja minimalnega vpetega drevesa v grafu*. Za reševanje problema obstaja več postopkov. Ta, ki ga opisujemo zgoraj, se imenuje Primov algoritem.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

Obrazi z očali

3. – 5. razred

Vsak izraz na obrazu se ujema z eno obliko očal.

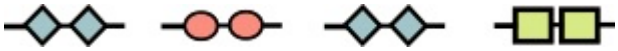
Tule imamo štiri obraze.



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

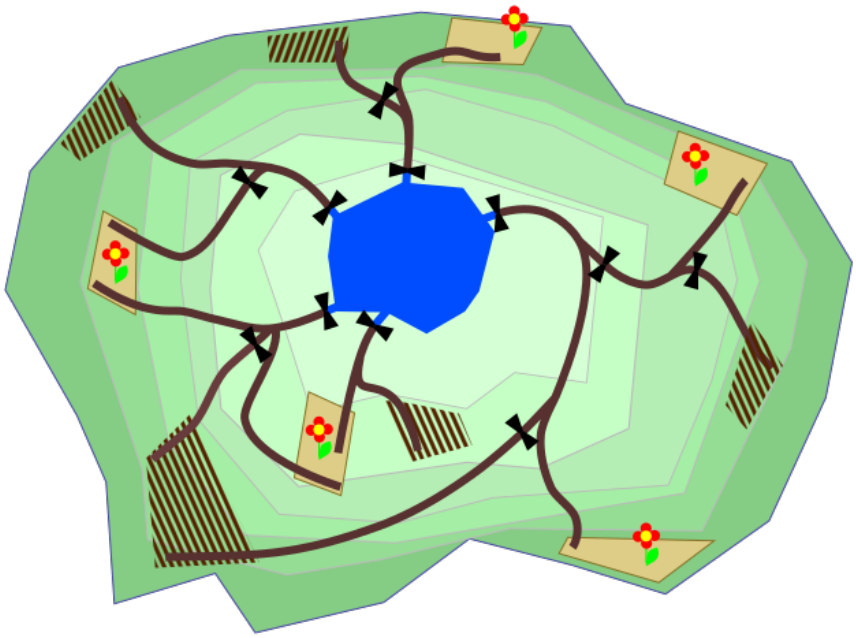
Številu razlik med dvema zaporedjema pravimo Hammingova razdalja. Uporabljamo jo, recimo, v genetiki, kjer s preštevanjem razlik v genskem zapisu določamo, kako različna sta dva gena.

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?



Grede, ki jih je potrebno zaliti Grede, ki jih ne smemo zaliti

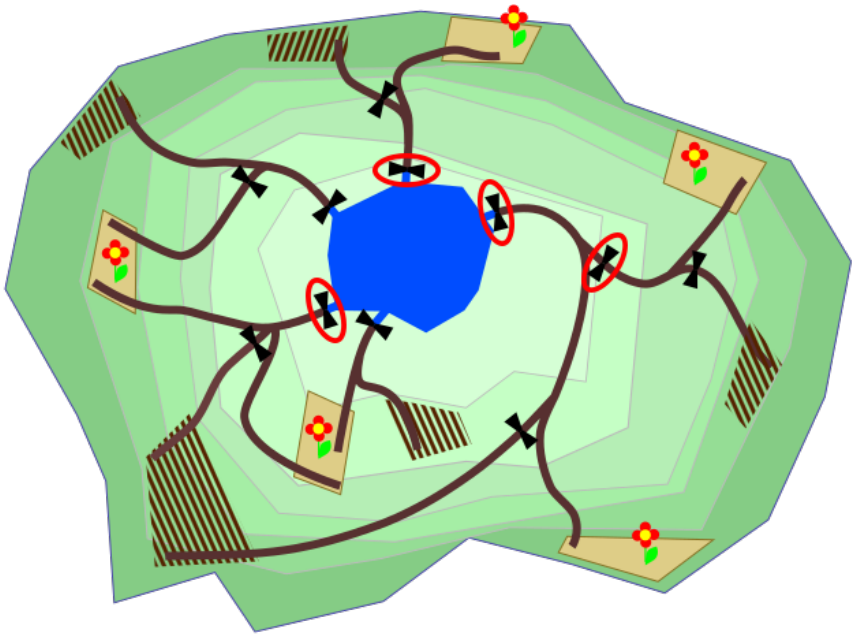
Rešitev

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.



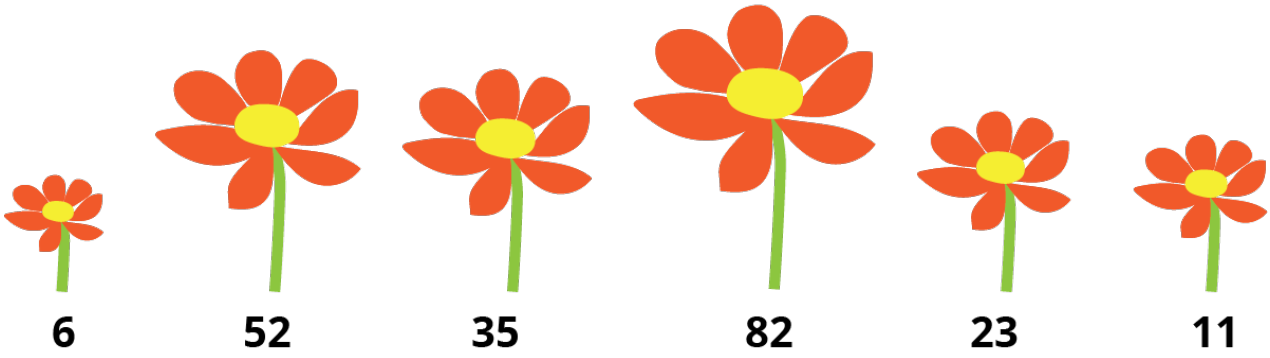
Grede, ki jih je potrebno zaliti Grede, ki jih ne smemo zaliti

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

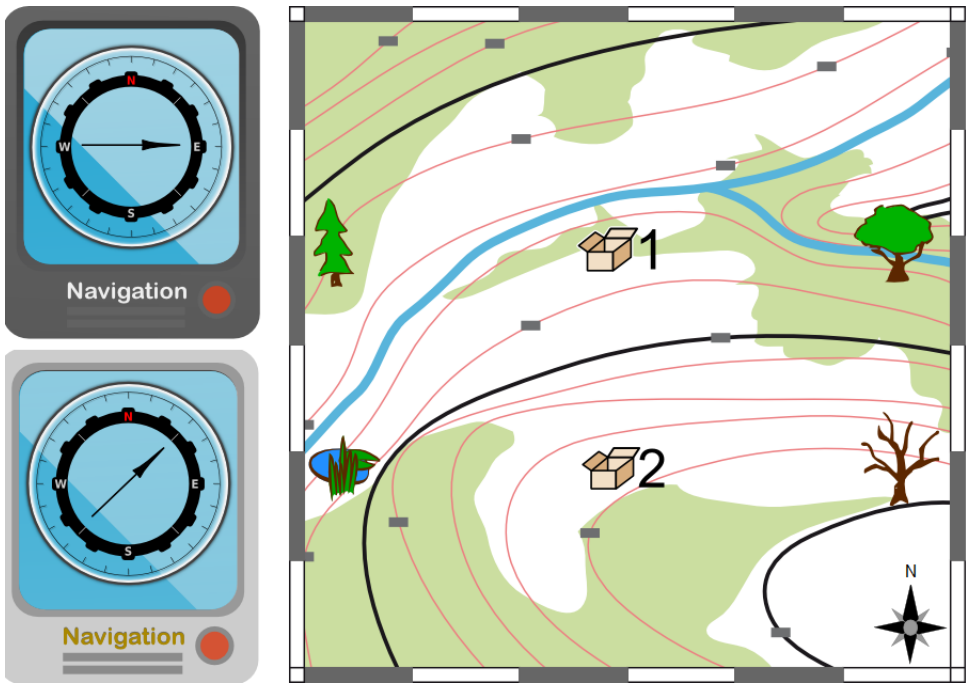
Lahko opišeš, na kakšen način se odloča čebela? Ni težko: čebela gre vedno na cvet, na katerem je največ nektarja.

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.

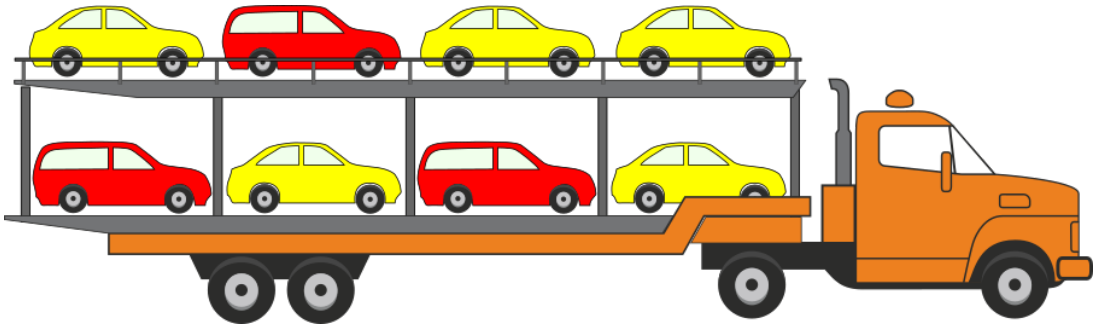
Prevoz avtomobilov

4. – 9. razred

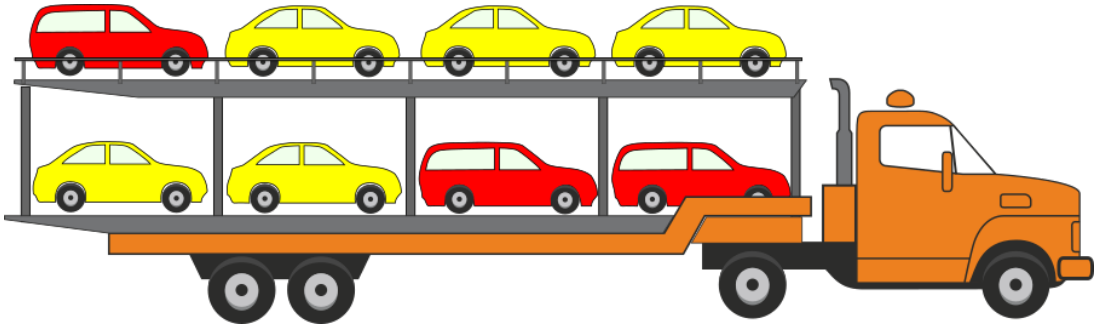
V tovarni avtomobilov so pravkar začeli proizvodnjo rdečih (temnejših) in rumenih (svetlejših) avtomobilov. Rdeči pride s proizvodne linije vsakih sedem minut, rumeni pa vsakih pet minut. Nakladajo jih na tovornjak – najprej zgoraj, nato spodaj.

Kako bo izgledal prvi tovornjak, ki ga bodo napolnili?

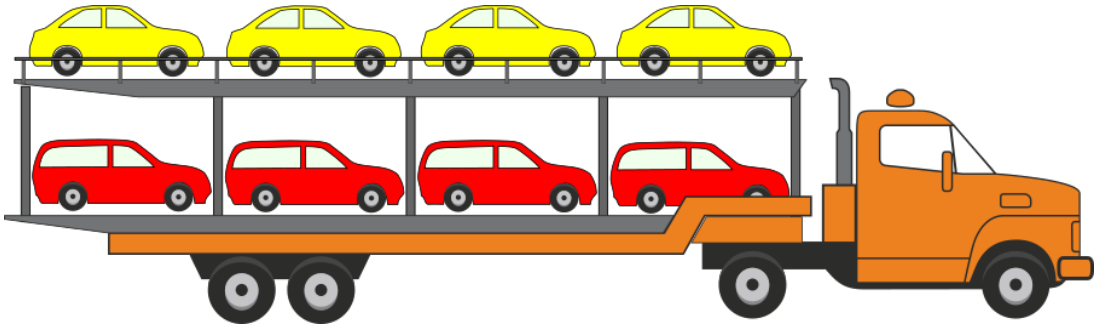
A.



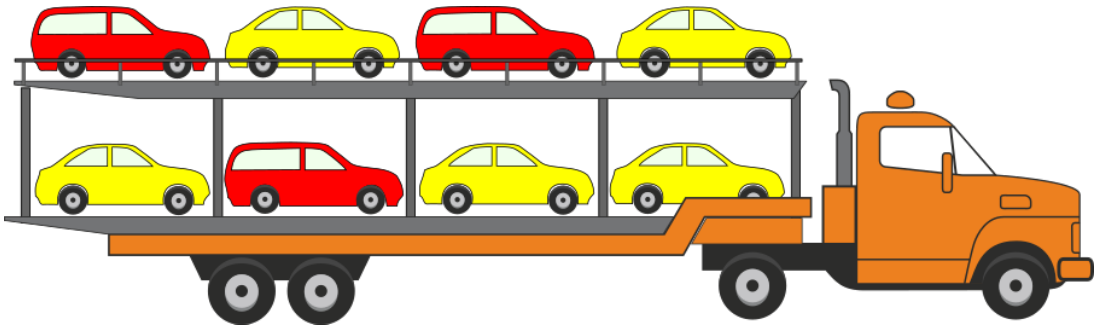
B.



C.



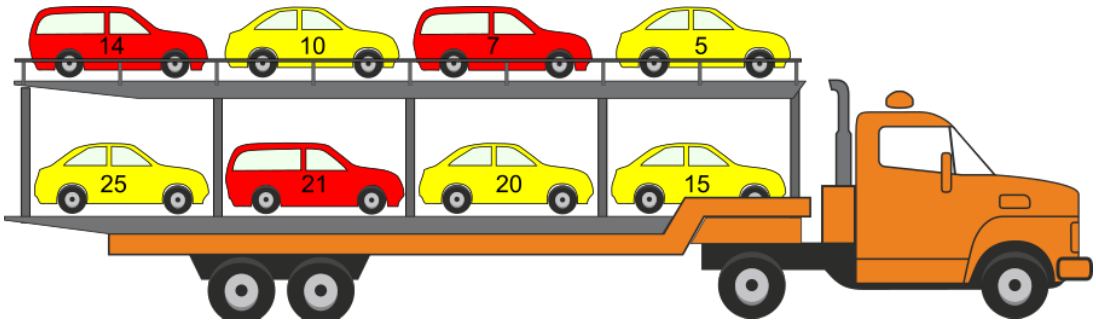
D.



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.



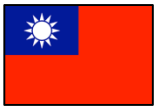
Računalniško ozadje

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.

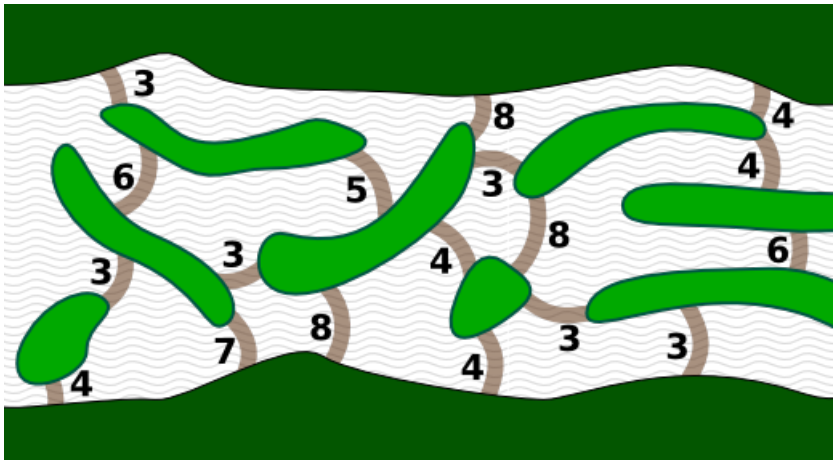
Tudi računalnikarji se morajo pogosto domisliti učinkovitih in uporabnih načinov zapisa podatkov.

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.

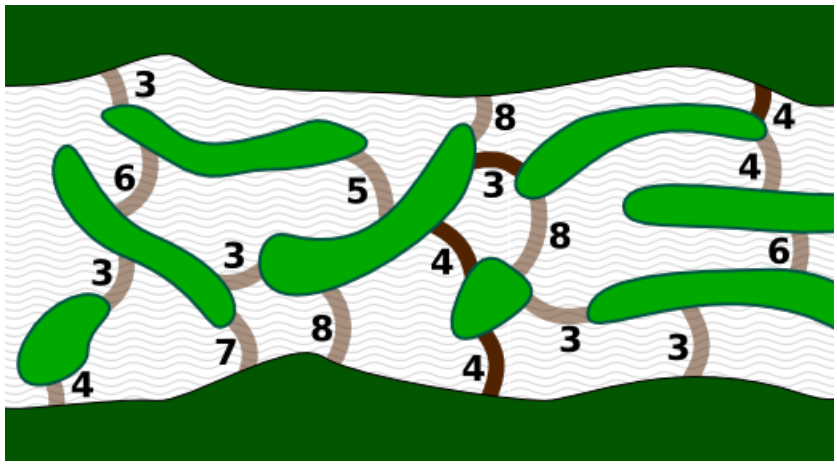
Skica kaže mesta, na katerih bi lahko postavili jezove, in koliko hlodov bi potrebovali za vsakega od njih. Najmanj koliko hlodov bodo potrebovali za zajezitev reke?



Rešitev

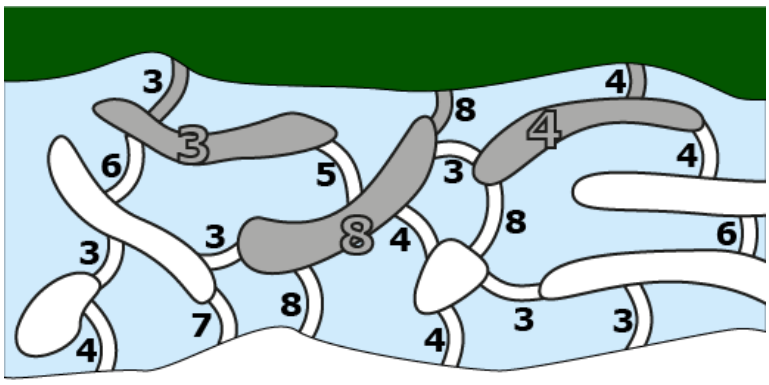
Potrebovali bodo 4 + 3 + 4 + 4 = 15 hlodov.

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.



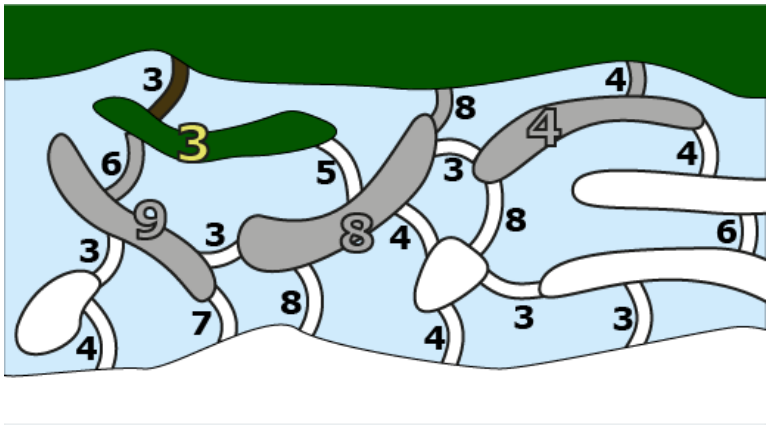
Nalogo si verjetno reševal s poskušanjem. Kako pa takšno nalogo rešimo sistematično?

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.)



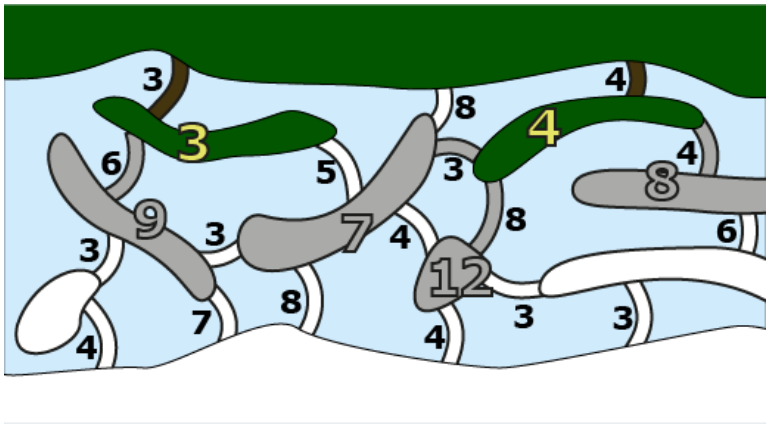
Vemo torej, da lahko dosežemo levi gornji otok s tremi hlodi. Z njega lahko dosežemo otok pod njim s skupno devetimi hlodi (tri do prvega in še šest naprej). Prav tako lahko pridemo do srednjega otoka z osmimi (5 + 3), a to že vemo.

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.



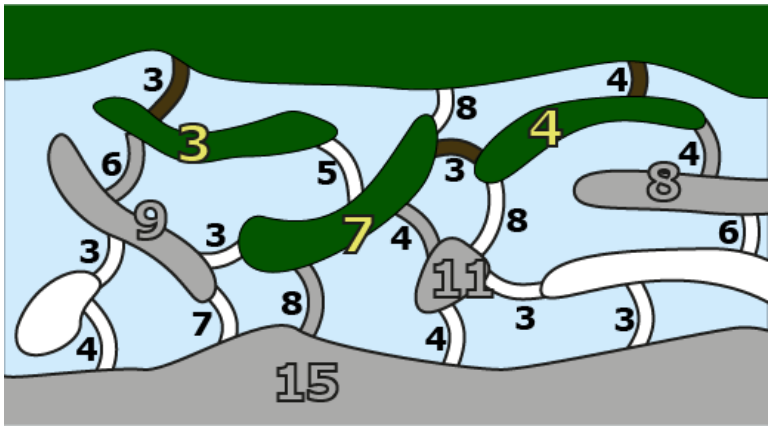
Za ostale sive otoke to ne velja: do njih bomo morda prišli po krajši poti. To pravzaprav vidimo takoj. Z gornjega desnega lahko dosežemo osrednji otok s skupno 4 + 3 = 7 hlodi! Popravimo osmico v sedmico. Poleg tega lahko z njega dosežemo dva otoka s skupno 4 + 8 = 12 in 4 + 4 = 8 hlodi.

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.



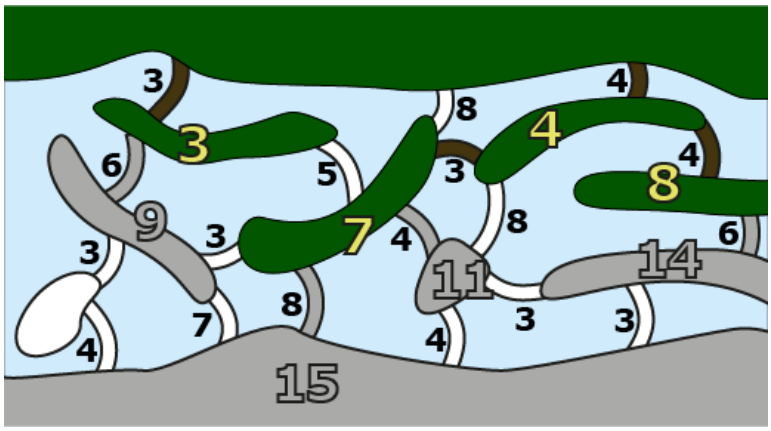
Osrednji otok označimo kot dosegljiv s sedmimi hlodi. Z njega bi lahko dosegli levega z desetimi, vendar se to ne splača, saj znamo do njega s 3 + 6 hlodi z levega gornjega. Pač pa zdaj vidimo, da lahko pridemo do otoka pod osrednjim s 7 + 4 = 11 hlodi, torej zamenjamo 12 z 11.

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.

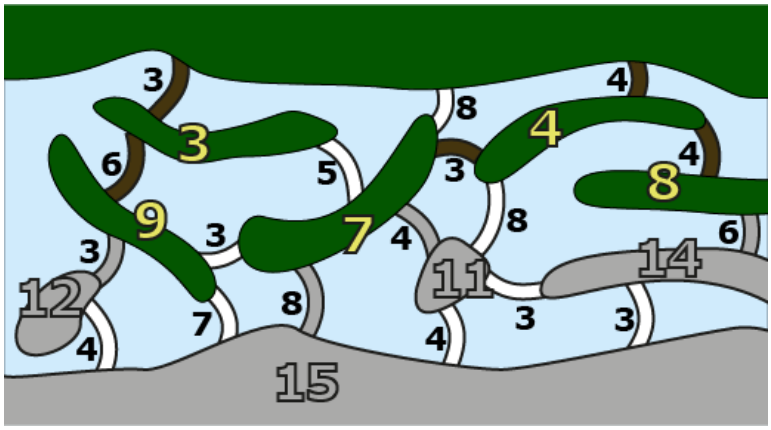


Zdaj dodamo levi otok.

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

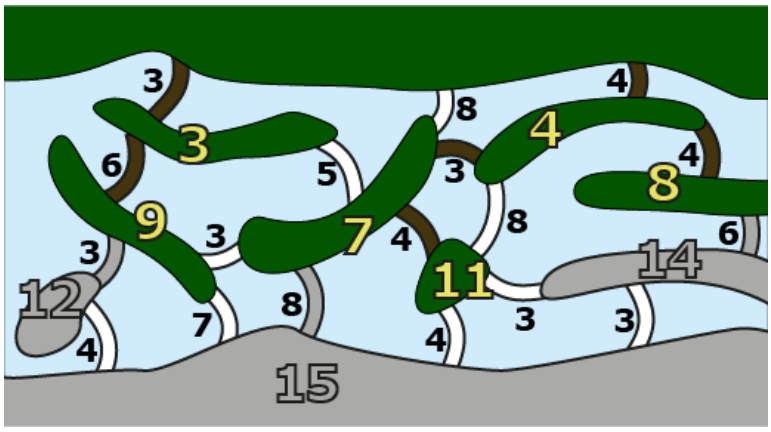
S tega otoka lahko gremo tudi na spodnji breg s skupno 9 + 7 = 16 hlodi, vendar se to ne splača, saj že poznamo krajšo pot.

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



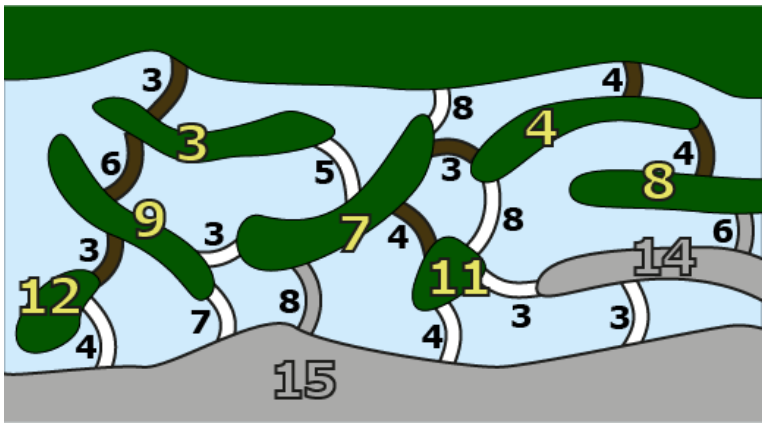
Dodamo otok, do katerega pridemo s skupno 11 hlodi. Z njega dosežemo spodnji breg z 11 + 4 hlodi. Ta rešitev je enako dobra kot pot z osrednjega otoka.

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.



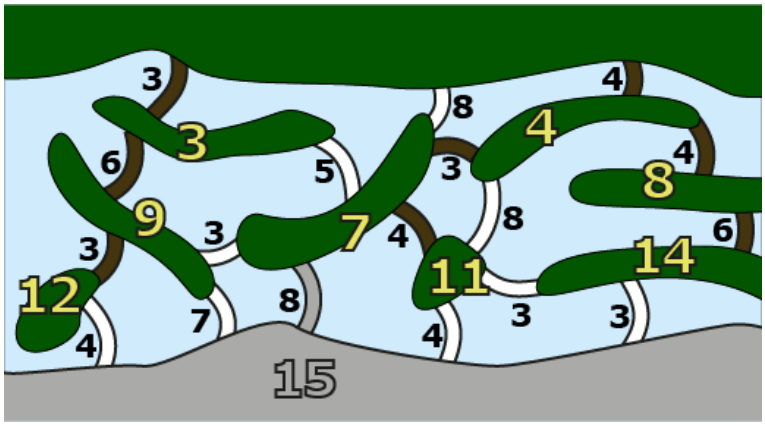
Med otoki označenimi s sivo spet izberemo otok, do katerega pridemo z najmanj hlodi. To je levi spodnji otok.

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

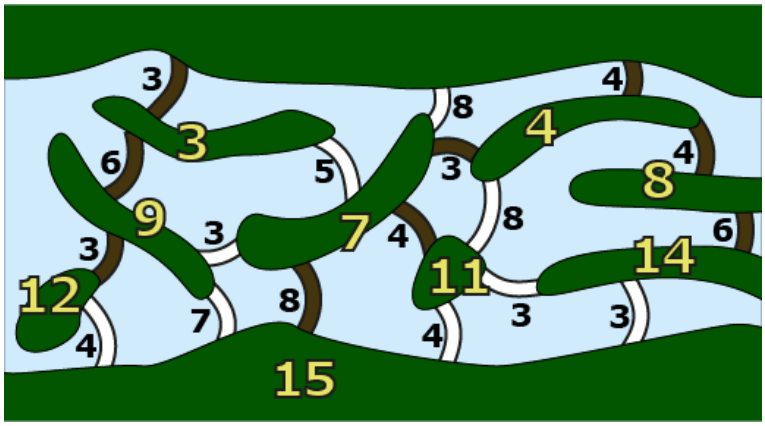


Zdaj je na vrsti desni spodnji otok.

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



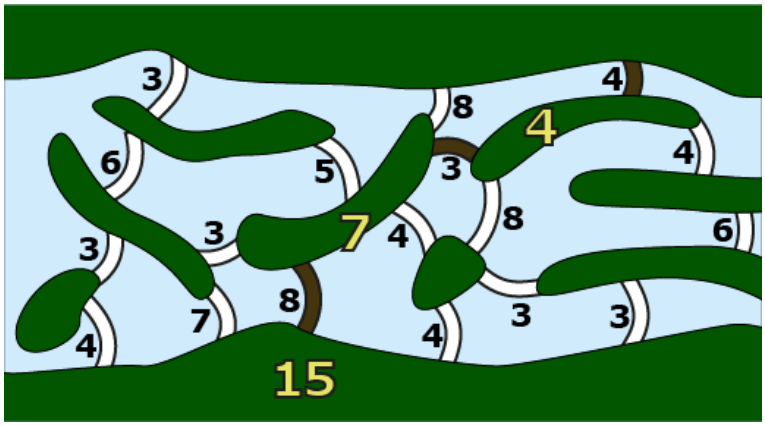
Končno dodamo še nasprotni breg, ki ga lahko s 15 hlodi dosežemo z osrednjega otoka (7 + 8) ali otoka desno pod njim (11 + 4).



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 vseno, 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.

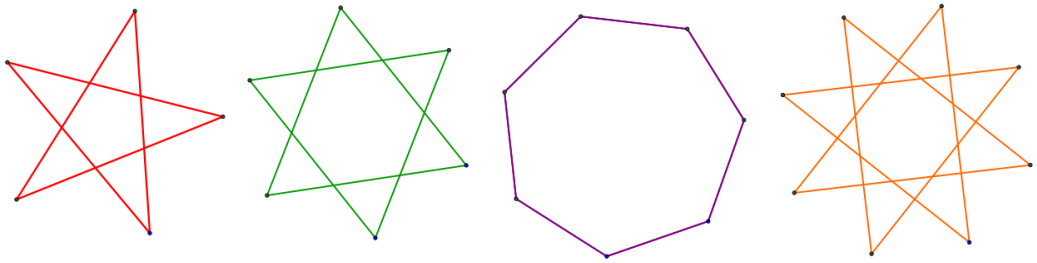
Opisovanje zvezd

6. – 9. razred

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

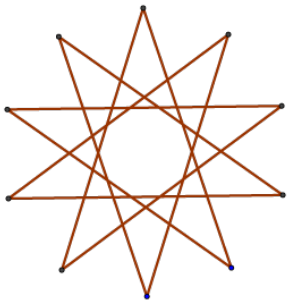
- prva številka pove število krakov in
- druga številka pove, kako so kraki povezani med seboj.

Nekaj primerov kažejo slike.



5:2 **6:2** **7:1** **8:3**

Kako bi Stela označila tole zvezdo?



Rešitev

10:4.

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

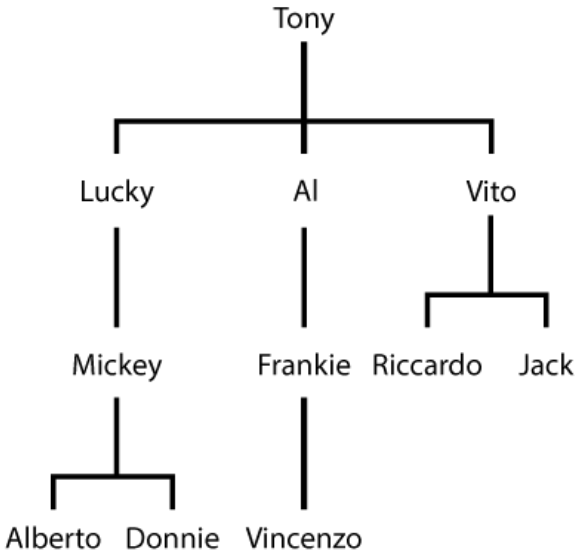
Računalniško ozadje

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.

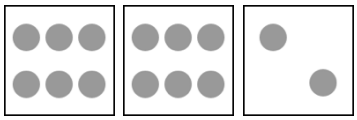
Kam gremo

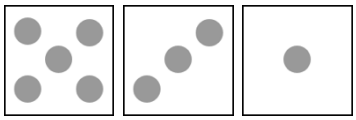
6. – 9. razred

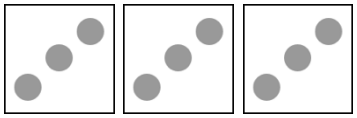
Da se bobri ne bi kregali, kam se gredo igrat po šoli, trikrat vržejo kocko in se odločijo takole

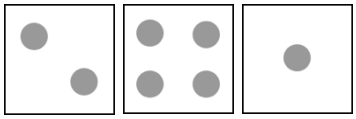
ČE je prvi met večji od drugega
POTEM gredo v gozd
SICER **ČE** je tretji met manjši od prvega
POTEM gredo k reki
SICER gredo na igrišče

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

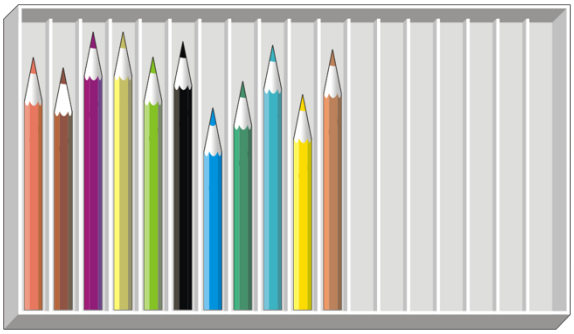
6. – 9. razred

Mali bober Božidar se je naveličal risanja. Na mizi pred njim je osemnajst barvic. Zložil jih bo v mamino in očetovo škatlo.

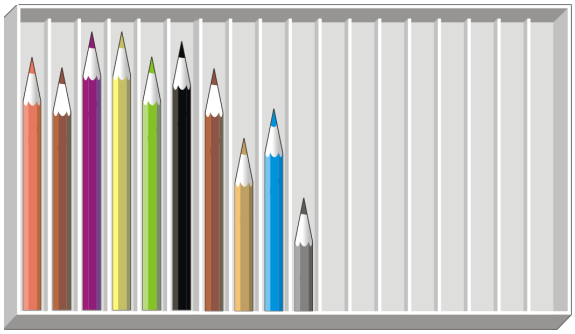
- Jemal jih bo z leve proti desni.
- Odlagal jih bo v mamino škatlo, prav tako z leve proti desni.
- Prva barvica gre v mamino škatlo.
- Vsako barvico primerja z najbolj desno barvico v mamini škatli. Če je manjša od nje, jo doda v mamino škatlo. Sicer jo da v očetovo škatlo.

Kako bo po tem videti očetova škatla?

A.

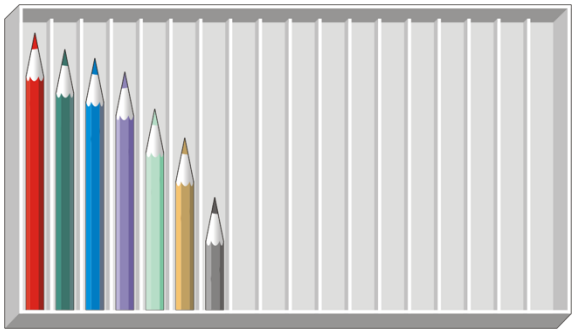


B.



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.

Nalogo lahko rešimo tudi drugače: razmislimo, kaj je v mamini škatli in izkaže se, da je to ravno, kar nam ponuja odgovor D. Če primerjamo začetne barvice in iz njih poberemo te, ki so v škatli D, dobimo ravno škatlo A.

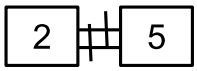
Računalniško ozadje

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

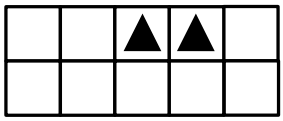
6. – 9. razred

Ukaz 

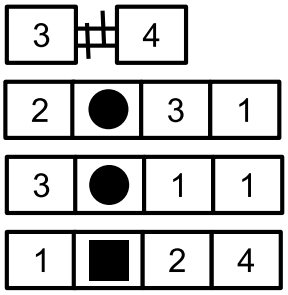
nariše mrežo z dvema vrsticama in petimi stolpci.

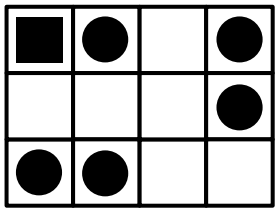
Ukaz 

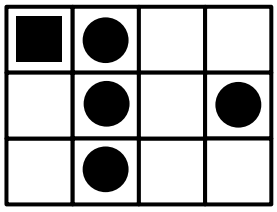
nariše dva trikotnika. Prvi je v prvi vrstici, v tretjem stolpcu. Drugi je desno od njega.

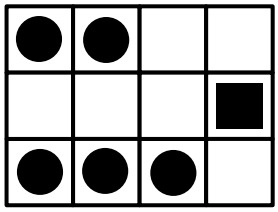
Oba ukaza zapored torej narišeta 

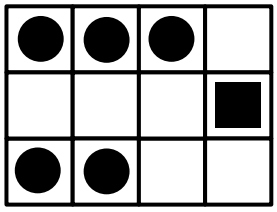
Kakšno sliko nariše spodnje zaporedje ukazov?



A. 

B. 

C. 

D. 

Rešitev

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

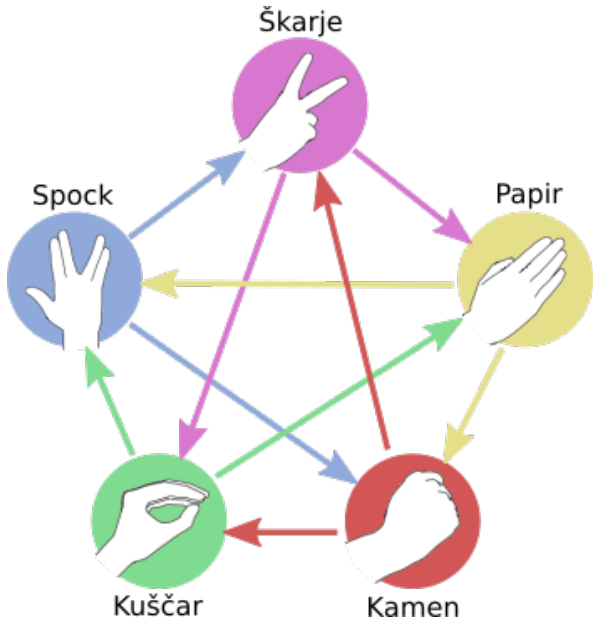
Računalniško ozadje

Kar opisuje naloga, je preprost programski jezik.

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.

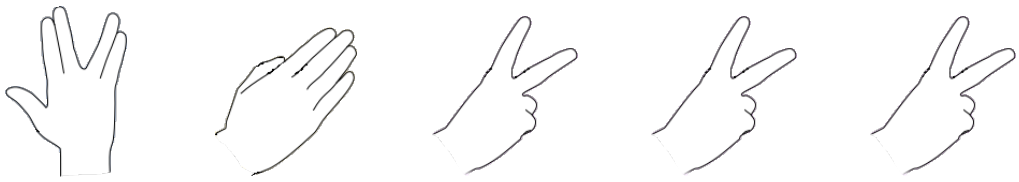


Sheldon in Rajesh sta v igro dodala kuščarja in Spocka. Kdo premaga koga, kaže skica. Tako, na primer, kamen premaga (zmečka) kuščarja.

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*.

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

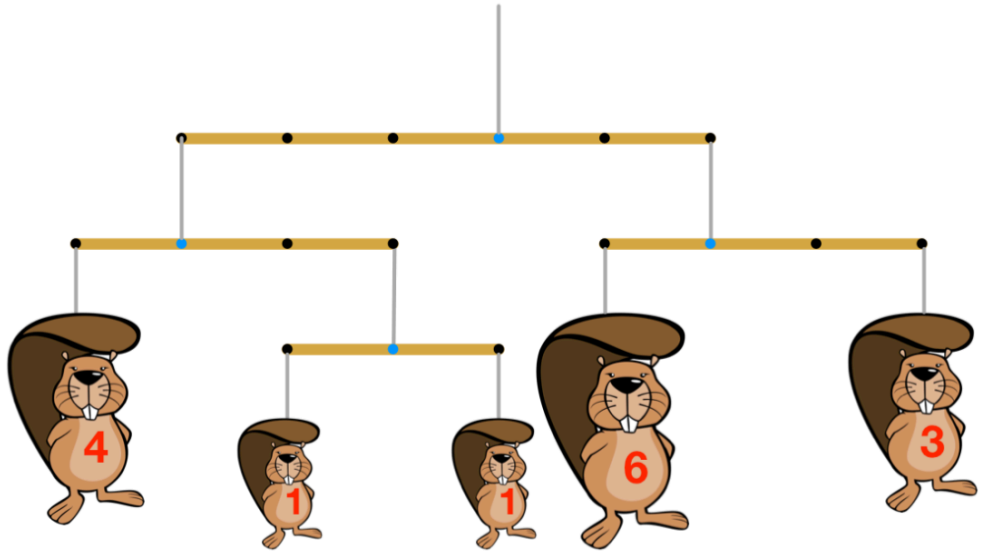
Po 5 urah bo torej imela Bernarda na svojem kupu 16 dreves, Alja pa le 15.

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.

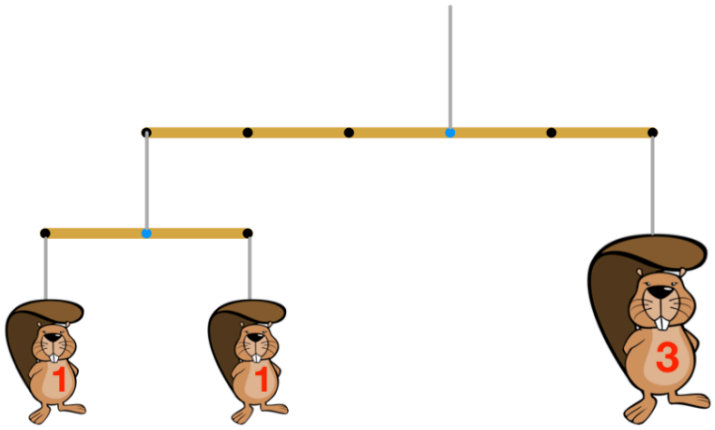
Obeski

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 pritjena eno plalica in en obesek. Palica je pritrjena 3 mesta levo od pritdilne vrvice, obesek, to je bober s številko 3, pa 2 mesti desno od pritdilne vrvice. Ta del zapišemo kot (-3 ?) (2 3).

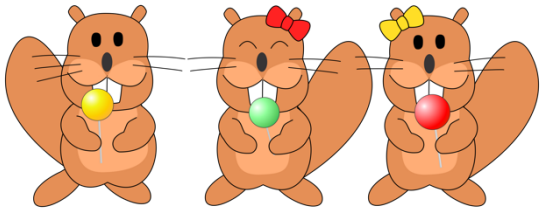
Zdaj pogledamo še kaj visi na mestu -3, kjer smo zaenkrat napisali ?. Tu imamo palico, na kateri sta pritrjena 2 obeska, in sicer na mestu -1 bober s številko 1 in na mestu 1 prav tako bober s številko 1. Ta del obeska zapišemo z (-1 1) (1 1).

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.



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

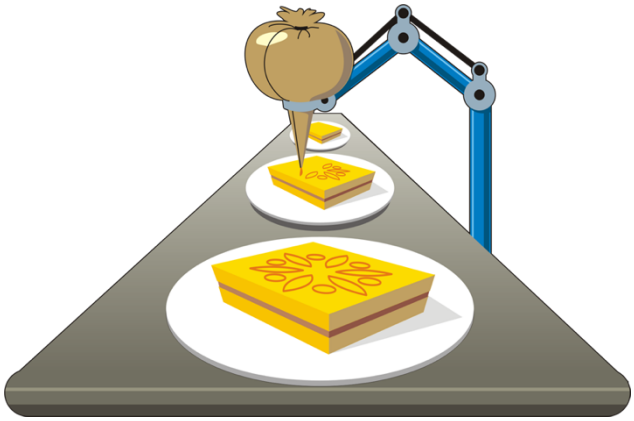
Naloga je primer problema, ki ga v računalništvu imenujemo *problem nahrbtnika*. Ta spada v družino problemov, pri katerih je naloga napolniti nahrbtnik s predmeti različnih velikosti ali cen. Obstaja več različic: včasih lahko v nahrbtnik spravimo le en primerek vsakega predmeta, včasih lahko predmete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

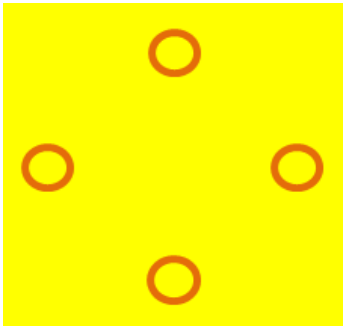
2. Ukaz `krog` nariše:

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.

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
									10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
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.

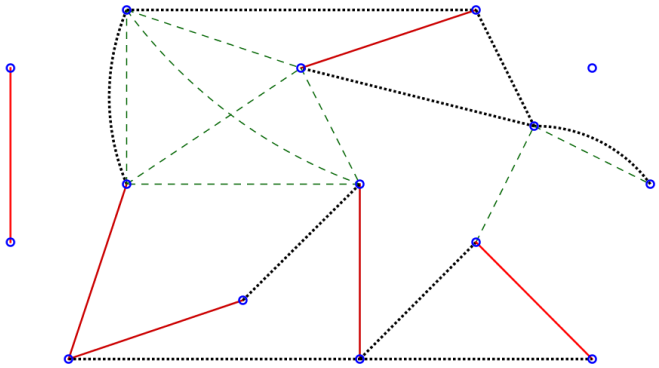
Računalniško ozadje

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.

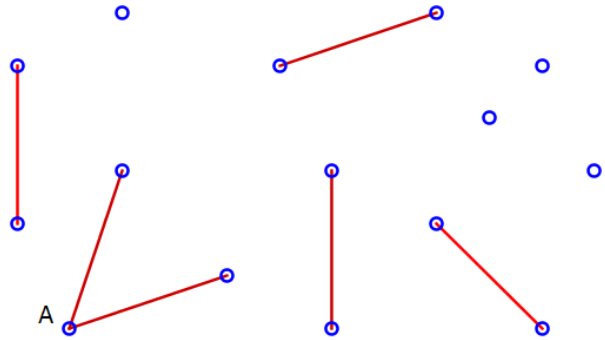
- Peter je povezal dvojčke z neprekinjeno rdečo črto.
- Jure je povezal prijatelje s pikčasto črno črto.
- Vid je povezal sošolce s prekinjeno zeleno črto.

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

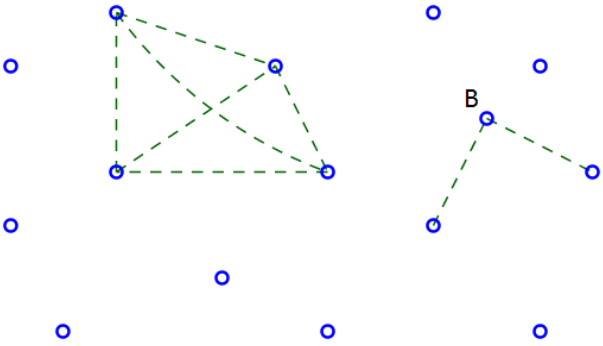
Rešitev

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.



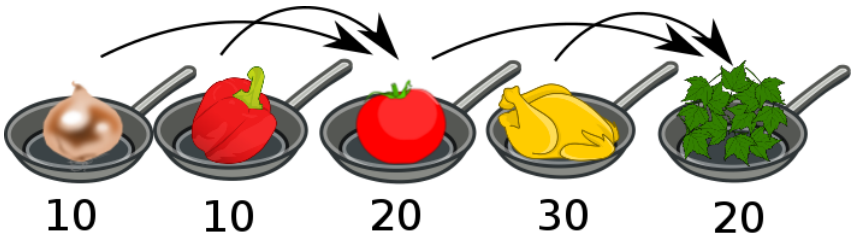
Računalniško ozadje

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.

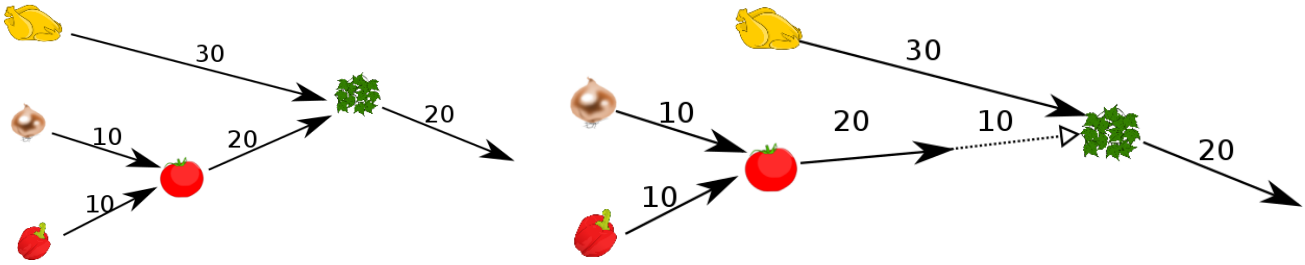
Kadar Sergej kuha v domači kuhinji, uporablja več kuhalnikov hkrati, zato je jed pripravljena hitreje. Katera od navedenih trditev **ni pravilna**?

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.

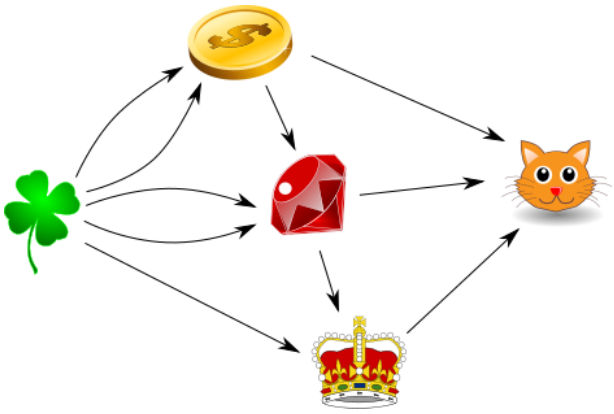
Računalniško ozadje

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

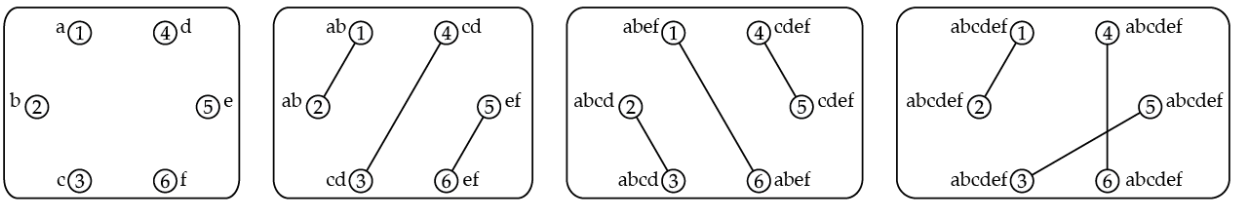
Naloga prikazuje, kako lahko uporabimo graf za prikaz odvisnosti med stvarmi. Graf je podatkovna struktura, ki se veliko uporablja v računalništvu za prikaz razmerij. Graf nam tudi pomaga pri vizualizaciji naloge, kar je veliko lažje razumeti kot pri branju tekstovnega opisa povezav.

Vohuni

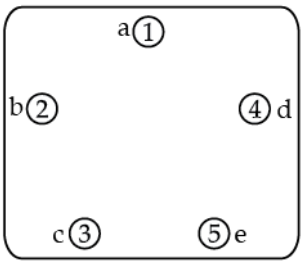
1. – 4. letnik

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?



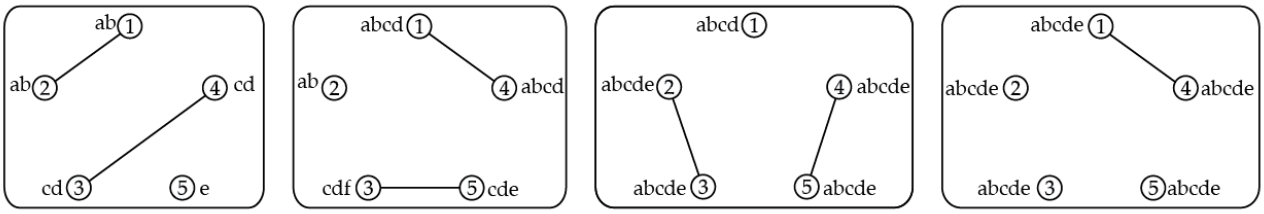
Rešitev

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

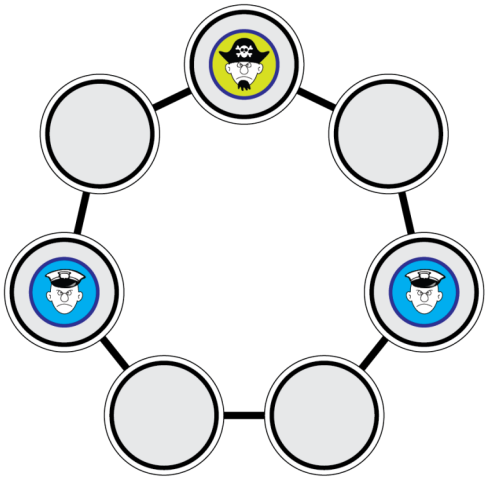
Kadar si računalniki izmenjujejo informacije, to pogosto delajo v parih. Tako lahko nastane problem, kako naj si v najkrajšem času izmenjajo informacije vsi računalniki v mreži. Torej morajo računalničarji rešiti podoben problem, kot so ga imeli vohuni. Problem je poznan tudi kot problem opravljivcev (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

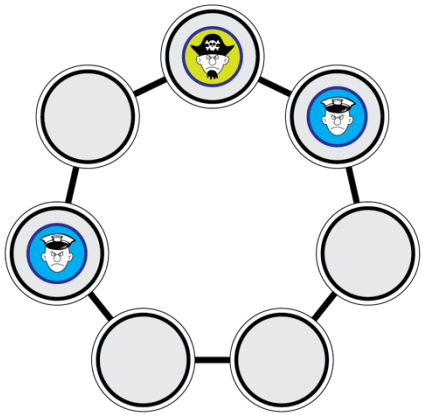
Jana in Janko igrata namizno igro *Lovec na pirate*. Ob vsaki potezi se eden od policistov (a ne oba) premakne na sosednje polje. V naslednji potezi se premakne pirat, ki pa je hitrejši od policistov in se vedno premakne za dve mesti. Policist se lahko premakne le na prosto mesto (ne more se premakniti 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.

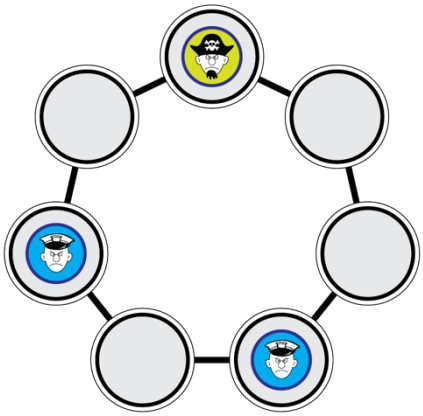


Jana, ki igra pirata, je precej spretna pri izogibanju policistom. Pomagajte Janku, ki igra policista, da bo odigral popolno igro in zmagal. Koliko potez mora narediti, preden bo lahko ujel pirata?

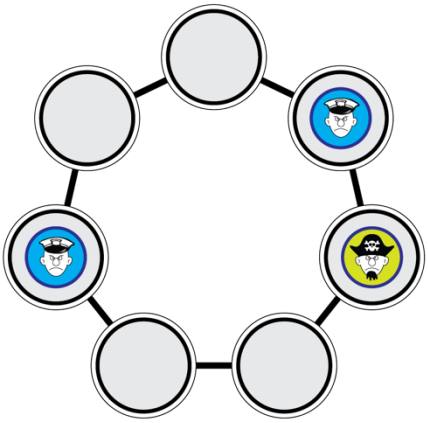
Rešitev

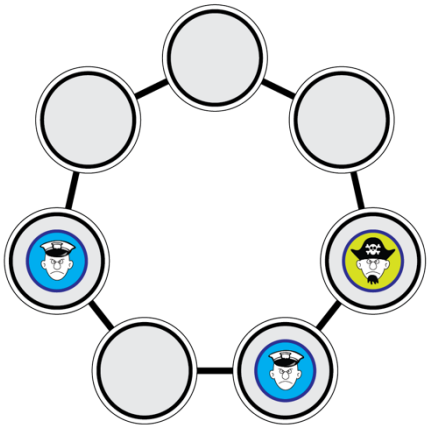
Če Jana igra dovolj spretno, Janko ne more zmagati. Predpostavimo, da bi mu jo uspelo prisiliti v položaj, ki ga prikazuje slika, in bi bila tako prisiljena v premik na polje, ki ga zaseda policist, s čimer izgubi igro. Kakšna je bila postavitev pred Jankovo zadnjo potezo? Janko je lahko premaknil levega 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:

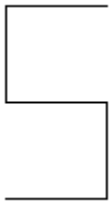
- **naprej** – robot gre en korak naprej,
- **obrat** – robot se obrne za 90 stopinj.

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.

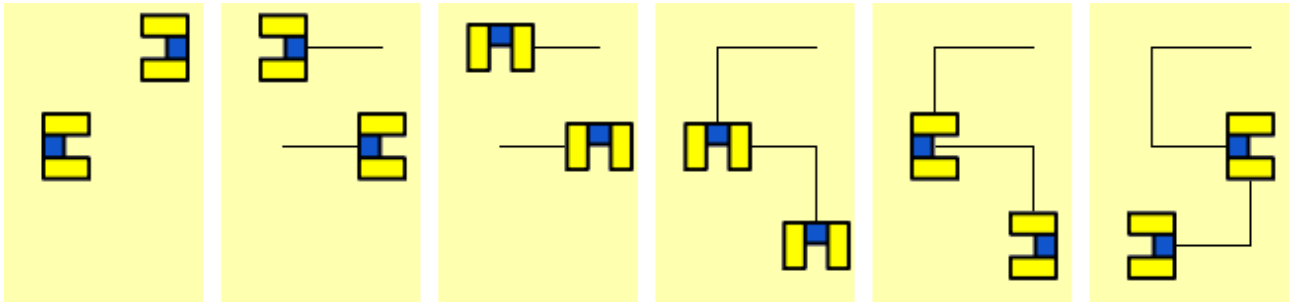
D.

Rešitev

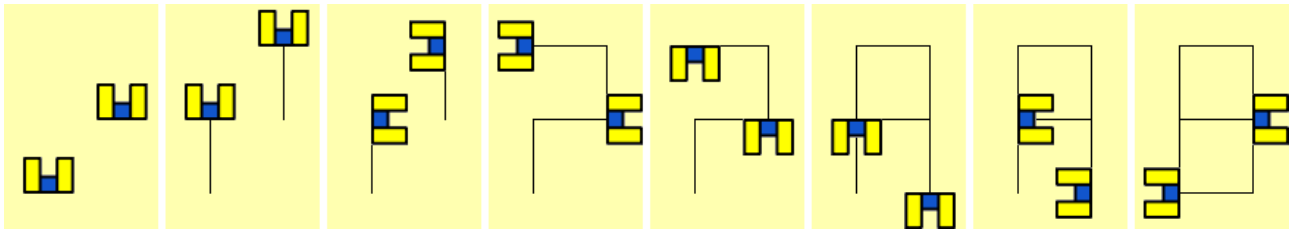
Pravilni odgovor je D. Robota nista mogla narisati črke L.

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

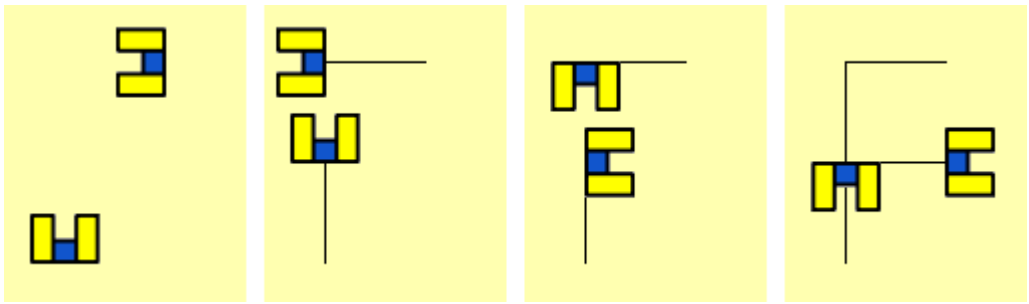
A: Prva slika prikazuje začetni poziciji in orientaciji obeh robotov. Vsakemu robotu pošljemo zaporedje naslednjih ukazov: naprej, obrat, naprej, obrat, naprej.



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.

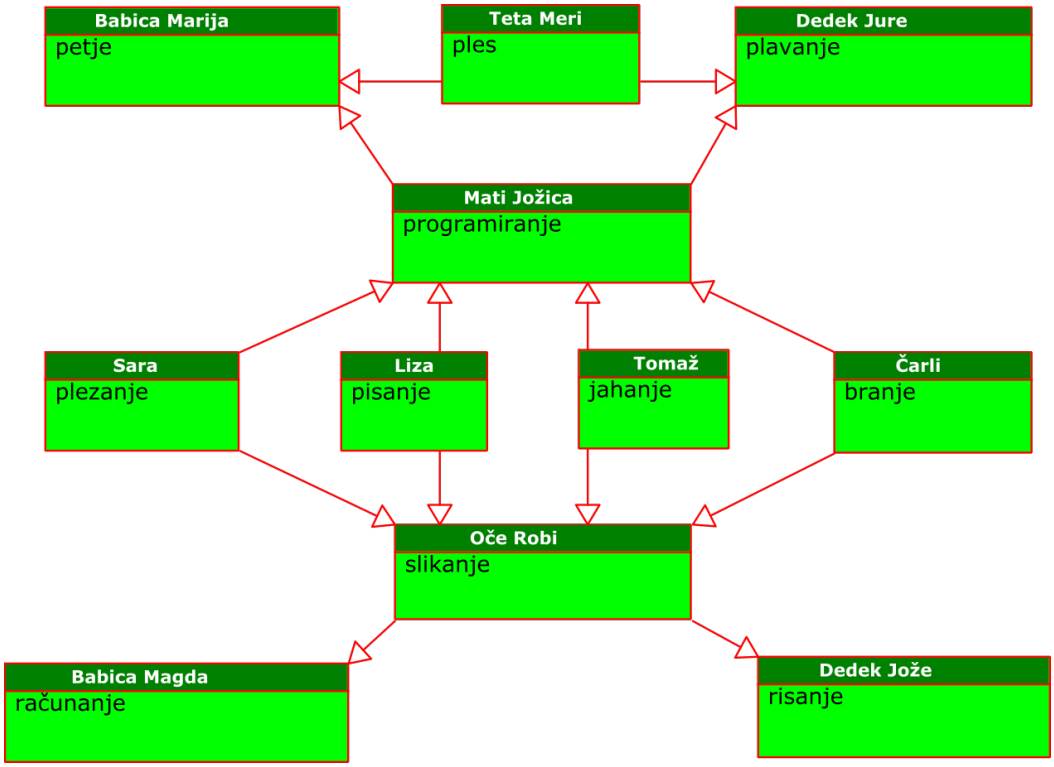
Računalniško ozadje

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 dedeka 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 objetno orientiranjega programiranja. Dedovanje nam omogoča logično povezovanje in razširjanje podatkovnih struktur in njihovih algoritmov.

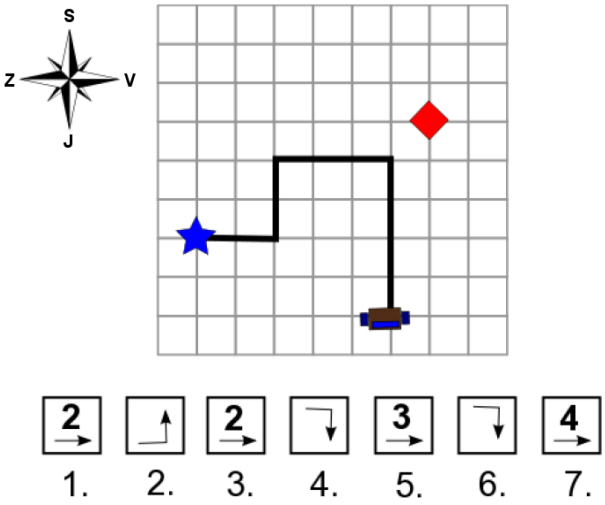
Robota nadziramo s tremi tipkami.

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

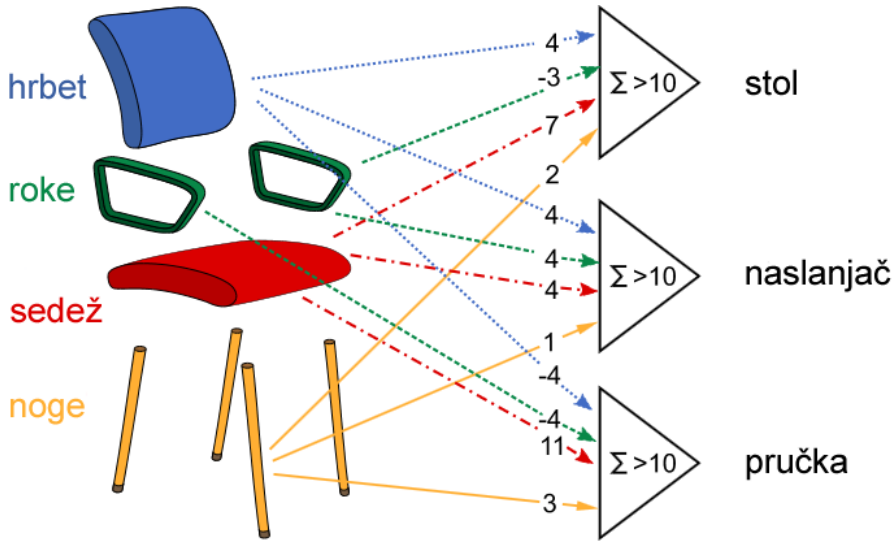
Računalnik je stroj, ki ga programiramo podobno, kot upravljamo robota, le da imamo na voljo veliko več različnih ukazov. Iz zaporedja ukazov sestavljamo programe. Programi vsebujejo od samo nekaj, pa do več tisoč ukazov.

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).