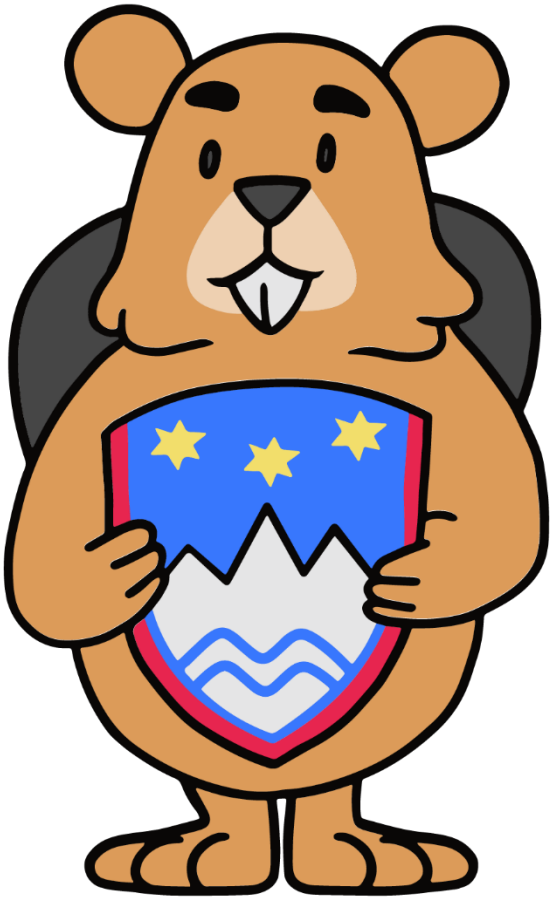
Bober 2025/26

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

Alenka Kavčič (UL FRI)
Nežka Rugelj (OŠ Trzin)
Rok Požar (UP FAMNIT)
Špela Cerar (UL PEF)



Bebras

ACM Slovenija

Naloge in rešitve nalog z državnega tekmovanja

Uredniki zbirke nalog in rešitev:

Alenka Kavčič (UL FRI)
Nežka Rugelj (OŠ Trzin)
Rok Požar (UP FAMNIT)
Špela Cerar (UL PEF)

Razvoj tekmovalnega sistema:

Gašper Fele Žorž (UL FRI)
Gregor Jerše (UL FRI)

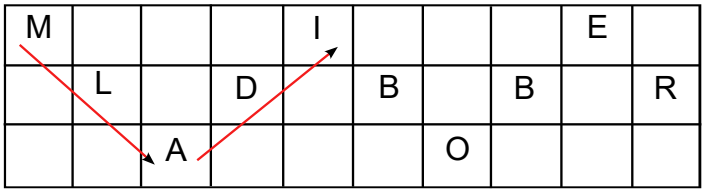
Kazalo nalog

Cik-cak šifra

6. razred

Za šifriranje sporočil s cik-cak šifro uporabljamo pravokotno mrežo podane višine (to je število vrstic v mreži). V zgornje levo polje napišemo prvo črko sporočila, drugo črko v polje desno spodaj in tako naprej v cik-cak obliki.

Spodnja slika prikazuje, kako v mrežo višine 3 vnesemo črke za kodiranje sporočila **MLADIBOBER**:



Znake iz mreže preberemo po vrsticah od leve proti desni in v zgornjem primeru dobimo šifrirano sporočilo: **MIELDBBRAO**

Kako izgleda sporočilo BOBROVTEDEN, šifrirano s cik-cak šifro na mreži višine 4?

A) BBDEENOORTV
B) BTOVEBODNRE
C) BTBODNOVERE
D) BODOVEBTNRE
E) BODORVEEBTN

Rešitev

Sporočilo najprej zapišemo s cik-cak šifro v mrežo višine 4:

B						T				
	O				V		E			
		B		O				D		N
			R						E	

Ko sporočilo preberemo po vrsticah, dobimo: BTOVEBODNRE.

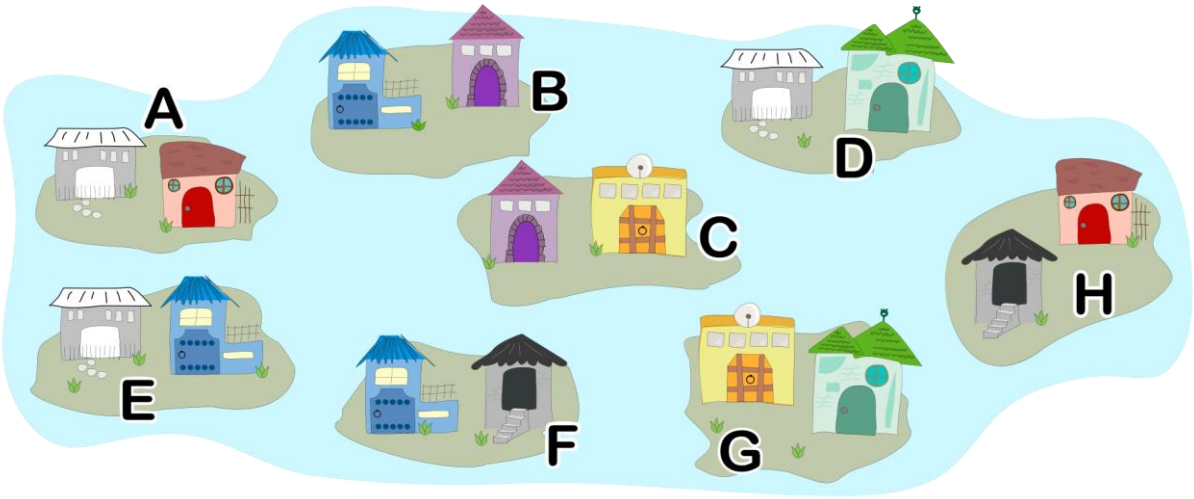
Računalniško ozadje

Šifrirni algoritmi imajo v računalništvu pomembno vlogo. Napredna kriptografija zagotavlja večjo varnost naših podatkov, pogovorov in bančnih transakcij.

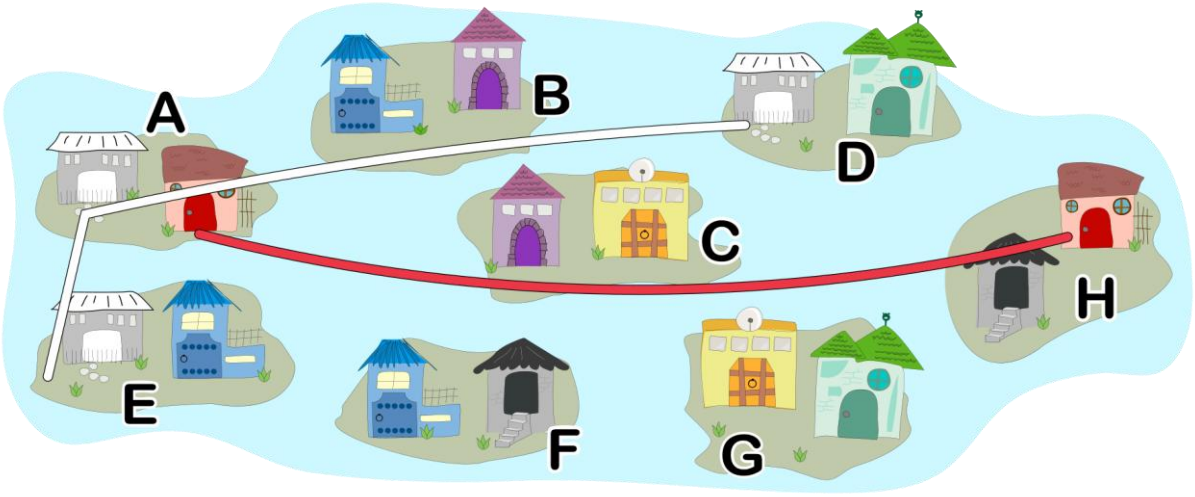
Čudežni otoki

6. razred

Med osmimi čudežnimi otoki lahko potuješ tako, da vstopiš v eno hišo in izstopiš iz enake hiše (enake oblike in barve) na drugem otoku.



Če si na otoku A, lahko skozi belo hišo odideš na otoka D ali E, skozi rdečo hišo pa do otoka H.

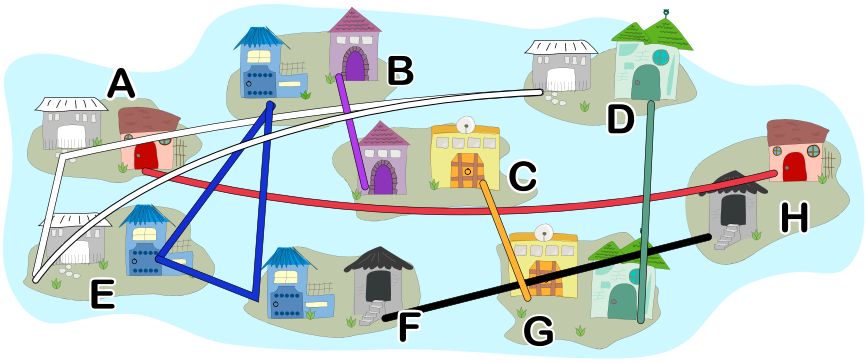


Nora ima zemljevid, na katerem so otoki označeni s črkami. Prikazane so tudi vse hiše na otokih.

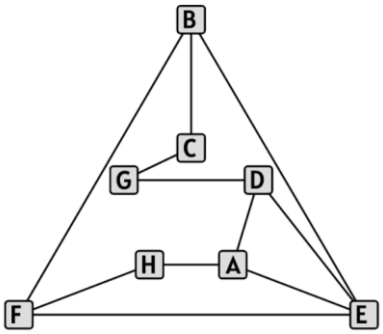
Najmanj koliko parov enakih hiš mora uporabiti Nora, da bo prišla z otoka A na otok C?

Rešitev

Nalogo lahko rešimo tako, da najprej povežemo enake hiše:

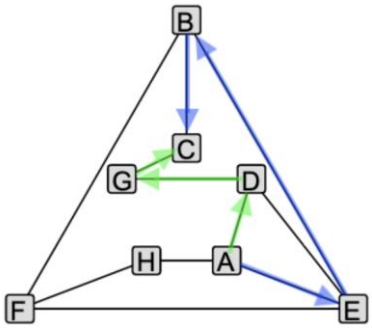


Ta zemljevid lahko prerišemo v obliko grafa, na katerem so prikazane povezave med posameznimi otoki:



Najkrajša pot med otokom A in C je dolga 3. Taki poti sta 2:

a) preko otokov D in G – v tem primeru mora Nora izstopiti iz bele hiše na otoku D, iz zelene hiše na otoku G in iz rumene hiše na otoku C.
b) preko otokov E in B – v tem primeru mora nora izstopiti iz bele hiše na otoku E, iz modre hiše na otoku B in iz vijolične hiše na otoku C.



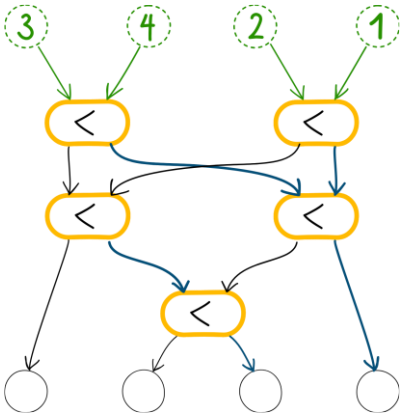
Računalniško ozadje

Za razlago rešitve smo sliko prerisali kot graf. Otoki v nalogi predstavljajo vozlišča, povezave v grafu pa ponazarjajo možnost potovanja z enega otoka na drugega.

Čudovita naprava 1

6. razred

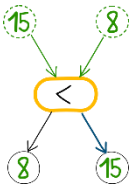
Bobri imajo čudovito napravo. Vanjo na označenih mestih na vrhu vstavijo štiri števila: 3, 4, 2 in 1.



Števila potujejo skozi napravo od vhodov preko puščic in stikal  do izhodov na dnu.

Vsako od petih stikal primerja števili, ki jih dobi, in ju usmeri naprej: manjše število na levo in večje število na desno.

Na primer:

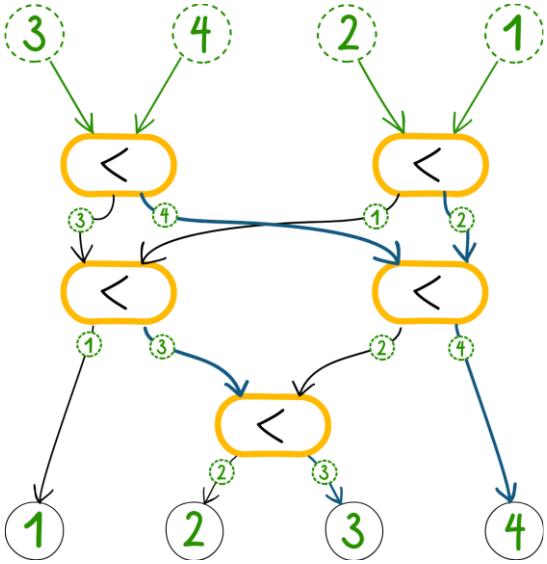


Katero nalogo opravlja čudovita naprava?

A) Števila vrne v obratnem vrstnem redu. Rezultat: 1, 2, 4, 3
B) Števila uredi od najmanjšega do največjega. Rezultat: 1, 2, 3, 4
C) Števila uredi od največjega do najmanjšega. Rezultat: 4, 3, 2, 1
D) Števila vrne v istem vrstnem redu. Rezultat: 3, 4, 2, 1

Rešitev

Pravilen odgovor je B. Naprava razvrsti števila po velikosti od najmanjšega do največjega.



Računalniško ozadje

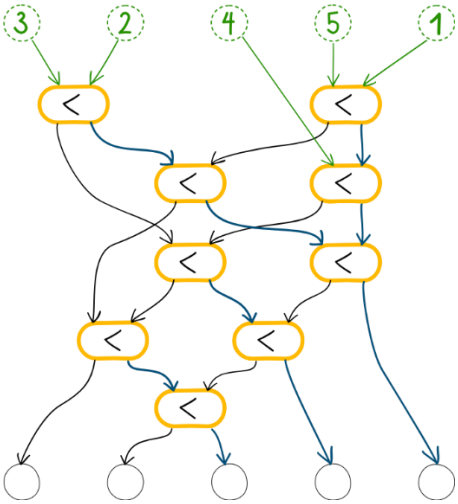
Čudovita naprava v nalogi je sortirna mreža (angl. sorting network). Zgrajena je iz več enakih, zelo preprostih komponent – primerjalnikov (angl. comparators). Vsak primerjalnik prejme dve številski vrednosti po dveh linijah in ju primerja. Manjšo vrednost posreduje po levi liniji, večjo pa po desni.

S povezovanjem zadostnega števila primerjalnikov je mogoče po velikosti urediti katero koli zaporedje števil. Za razvrščanje štirih števil je potrebnih najmanj pet primerjalnikov, za pet števil pa devet.

Čudovita naprava 2

8. in 9. razred, srednja šola

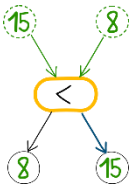
Bobri imajo čudovito napravo. Vanjo na označenih mestih na vrhu vstavijo pet števil: 3, 2, 4, 5 in 1.



Števila potujejo skozi napravo od vhodov preko puščic in stikal  do izhodov na dnu.

Vsako od devetih stikal primerja števili, ki jih dobi, in ju usmeri naprej: manjše število na levo in večje število na desno.

Na primer:

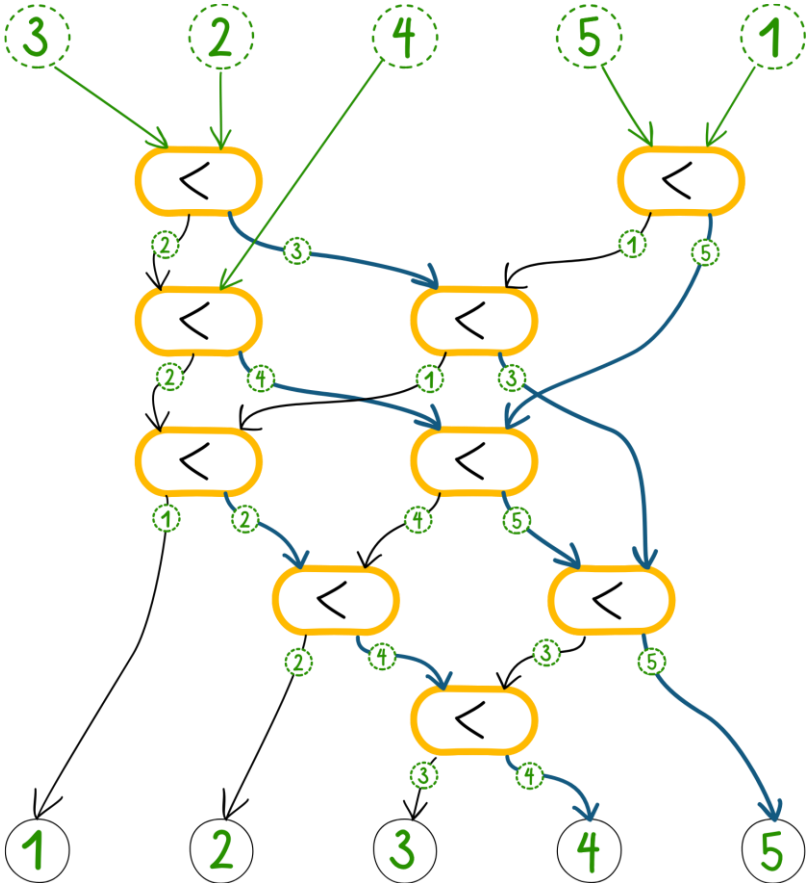


Katero nalogo opravlja čudovita naprava?

A) Števila vrne v obratnem vrstnem redu. Rezultat: 1, 5, 4, 2, 3
B) Števila uredi od najmanjšega do največjega. Rezultat: 1, 2, 3, 4, 5
C) Števila uredi od največjega do najmanjšega. Rezultat: 5, 4, 3, 2, 1
D) Števila vrne v istem vrstnem redu. Rezultat: 3, 2, 4, 5, 1

Rešitev

Pravilen odgovor je B. Naprava razvrsti števila po velikosti od najmanjšega do največjega.



Računalniško ozadje

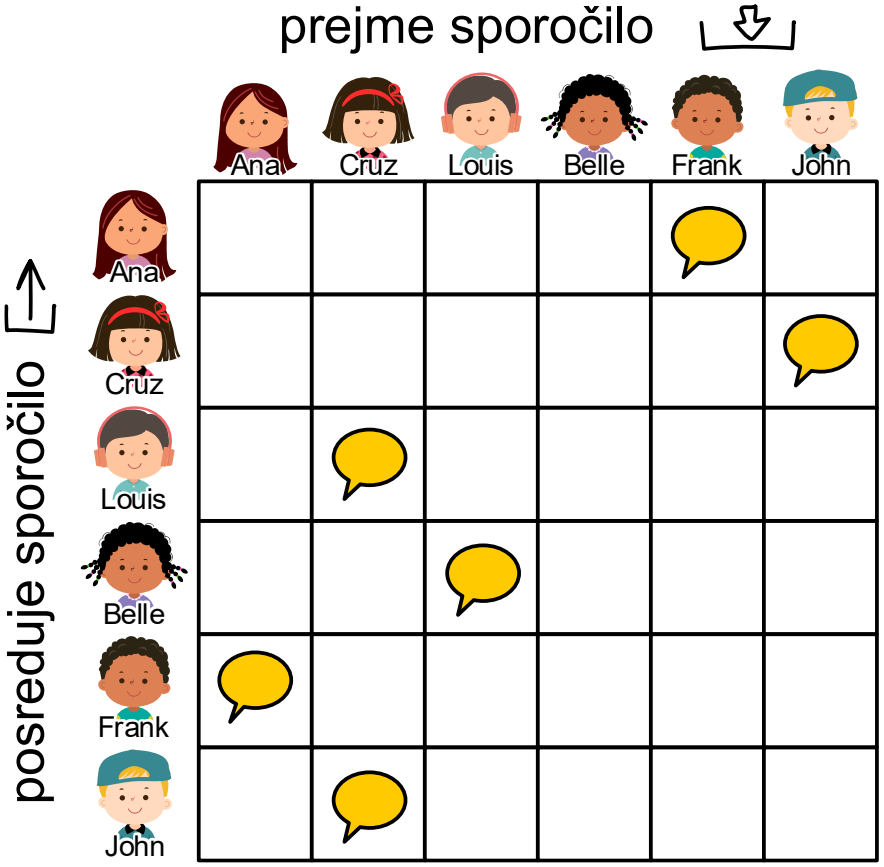
Čudovita naprava v nalogi je sortirna mreža (angl. sorting network). Zgrajena je iz več enakih, zelo preprostih komponent – primerjalnikov (angl. comparators). Vsak primerjalnik prejme dve številski vrednosti po dveh linijah in ju primerja. Manjšo vrednost posreduje po levi liniji, večjo pa po desni.

S povezovanjem zadostnega števila primerjalnikov je mogoče po velikosti urediti katero koli zaporedje števil. Za razvrščanje petih števil je potrebnih najmanj devet primerjalnikov, za šest števil pa dvanajst.

Posredovanje sporočil

6. razred

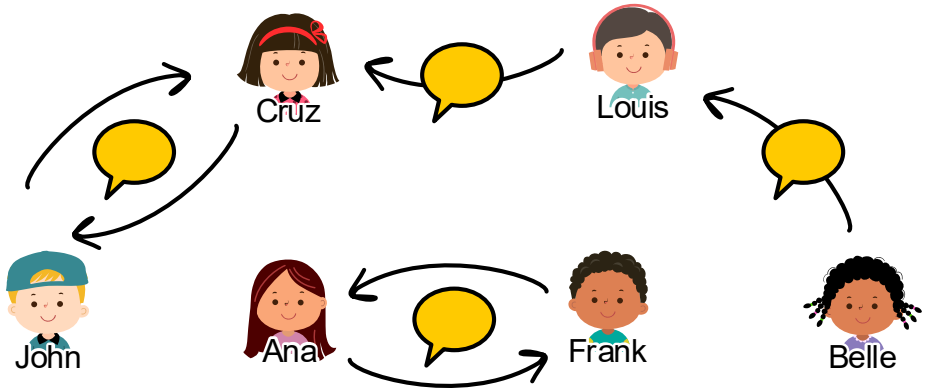
Šest prijateljev ima posebno pravilo: vsakič ko prejmejo sporočilo, ga posredujejo svojim prijateljem, kot določa spodnja tabela. Na primer: v prvi vrsti vidimo Ano, ki posreduje sporočilo samo Franku. Sporočilo nato potuje od prijatelja do prijatelja, dokler ga nekdo ne prejme ponovno.



Želimo, da sporočilo doseže kar največ prijateljev. Komu naj pošljemo sporočilo, da ga bo prejelo čim več prijateljev?

Rešitev

Kdo obvešča koga, lahko prikažemo tudi grafično:



Iz slike vidimo, da če bi sporočilo poslali Ani, bi ga ta posredovala Franku, Frank pa bi ji ga poslal nazaj. Obveščena bi bila le onadva. Podobno velja, če bi sporočilo poslali Franku.

Če bi sporočilo poslali Cruz, bi ga ta posredovala Johnu, on pa bi ji sporočilo vrnil, ostali pa ne bi bili obveščeni. Podobno bi bilo v primeru, ko sporočilo pošljemo Johnu.

Če sporočilo pošljemo Louisu, ga ta posreduje Cruz, Cruz Johnu in John nazaj Cruz – obveščeni so le trije.

Tako vidimo, da je najboljša rešitev, da sporočilo pošljemo Belle, saj ga bo tako prejel tudi Louis, ki ga bo posredoval Cruz, Cruz ga bo posredovala Johnu, John pa Cruz, s čimer se bo pošiljanje sporočil končalo. Neobveščena bosta ostala le Ana in Frank.

Pravilni odgovor je zato Belle.

Računalniško ozadje

Preglednica v tej nalogi predstavlja graf, ki prikazuje, kdo komu v skupini pošlje sporočilo. V tem primeru vsak prijatelj v skupini predstavlja vozlišče grafa, sporočilo, ki ga je mogoče poslati, pa predstavlja usmerjeno povezavo v grafu.

V tej nalogi imamo podane informacije o usmerjenem grafu, v katerem imajo povezave določeno smer. To pomeni, da povezava povezuje dve vozlišči v točno določeni smeri, kar je običajno prikazano s puščico.

Pozabljivi Albert

6. razred

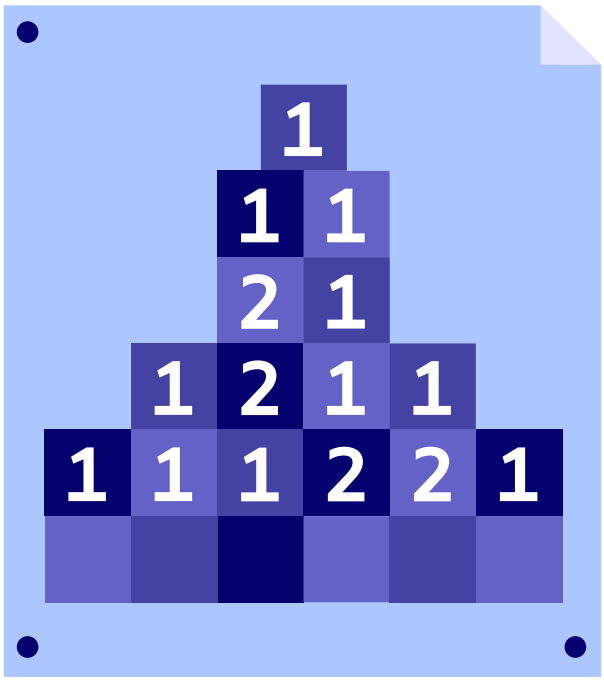
Pozabljivi izumitelj Albert si je naredil plakat, da si bo lažje zapomnil kodo za sef, v katerem hrani načrte svojih izumov.

Da dobi naslednjo vrstico kode, Albert vsako vrstico bere od leve proti desni in pri tem vedno opazuje najdaljši zaporedni niz enakih števk (na primer več enic zapored). Za vsak tak niz pove:

- koliko teh števk je v nizu in
- katera števka to je.

Iz tega nato sestavi naslednjo vrstico.

Primer: V prvi vrstici je zapisano 1. Albert prebere: »ena 1«, zato je druga vrstica 11.
Nato prebere 11 kot »dve 1« (dve enici).



Zadnja vrstica predstavlja kodo za odklepanje sefa. S katero 6-mestno kodo Albert odpre sef?

Rešitev

Predzadnjo (5.) vrstico beremo: »tri 1«, »dve 2« in »ena 1«. Torej je pravilen odgovor 312211.

Računalniško ozadje

V nalogi Albert ne pomni celotne kode za odklepanje, ampak si zapomni postopek: od leve proti desni poišče zaporedne nize enakih števk, prešteje njihovo dolžino in zapiše »koliko« ter

»katera števka«. Pomembno je, da Albert prepozna vzorec (opazi, kje se niz enakih števk začne in kje konča).

Podoben postopek, ki ga uporablja Albert za pomnjenje kode, pa se uporablja tudi pri stiskanju podatkov: namesto da shranimo npr. 1111112, lahko shranimo 6112 (šest enic in nato ena dvojka) in tako prihranimo prostor v pomnilniku. Takemu načinu stiskanja rečemo kodiranje ponovitev ali krajše RLE (angl. *run-length encoding*).

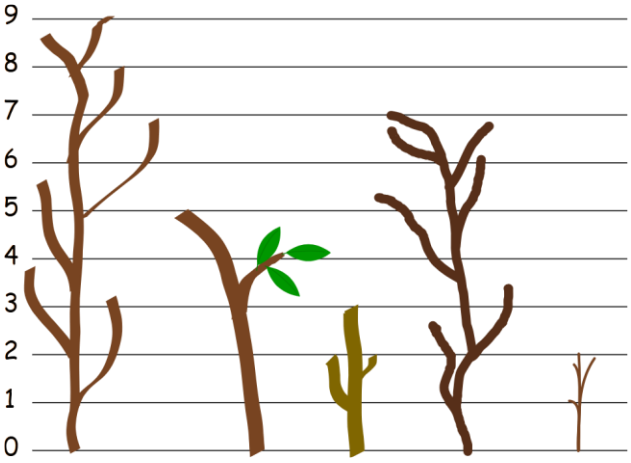
Razvrščanje vejic

6. razred

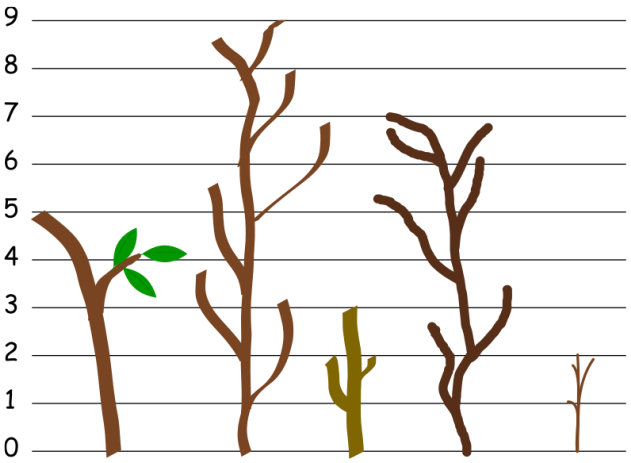
V Bobrovi vasi ima vsak svojo nalogo. Naloga bobra Jake je razvrščanje vejic po velikosti. Ko dobi nov kupček vejic, jih položi v vrsto in jih razvrsti po velikosti od najkrajše do najdaljše.

Pri razvrščanju vejic upošteva naslednja pravila:

1) Vedno začne pri najbolj levi vejici.
2) Primerja dve vejici, ki ležita ena poleg druge. Če je leva daljša od desne, ju zamenja, sicer ju pusti pri miru.
3) Premakne se za eno mesto desno, primerja vejici in ju po potrebi zamenja. Ko pride do konca vrste, ponovno začne z najbolj levo vejico. Postopek ponavlja, dokler niso urejene vse vejice.



Jaka je dobil 5 vejic različnih dolžin in jih postavil v vrsto. Dolžine vejic v vrsti si sledijo tako: 9, 5, 3, 7, 2. V prvem koraku primerja vejici dolžine 9 in 5 ter zamenja njuni mesti, saj je vejica dolžine 5 krajša od vejice dolžine 9. V vrsti ima sedaj vejice dolžine 5, 9, 3, 7, 2.



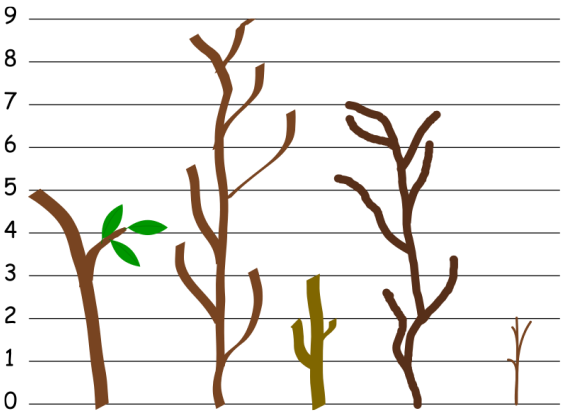
V drugem koraku primerja drugo in tretjo vejico in tako naprej. Vsaka primerjava šteje kot en korak.

V kakšnem vrstnem redu bodo vejice po 5. koraku?

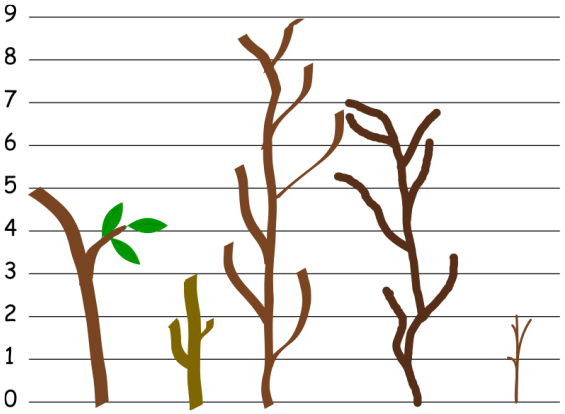
Rešitev

Po 5. koraku bodo vejice v naslednjem vrstnem redu: 3, 5, 7, 2, 9.

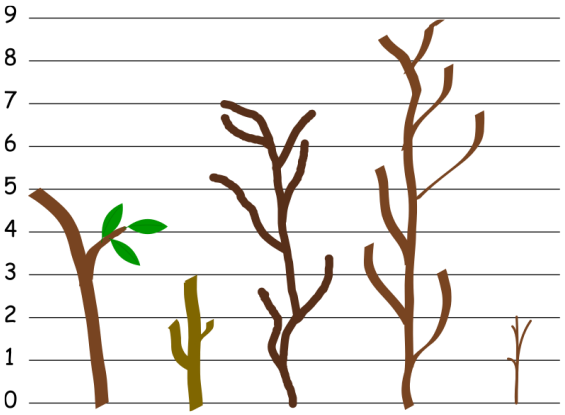
V 1. koraku Jaka primerja vejici 9 in 5 in zamenja njuni mesti, ostale vejice ostanejo, kjer so. Vrstni red je naslednji: 5, 9, 3, 7, 2.



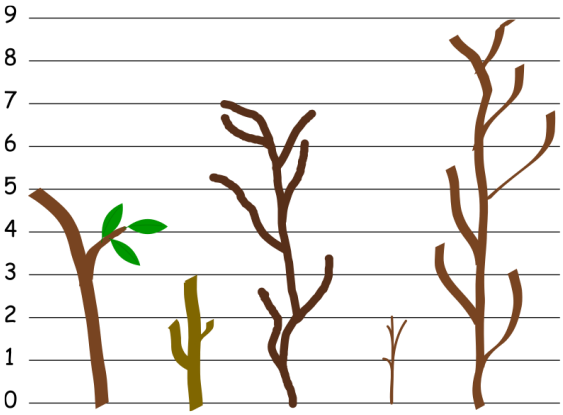
V 2. koraku Jaka primerja vejici 9 in 3 ter zamenja njuni mesti. Vrstni red je naslednji: 5, 3, 9, 7, 2.



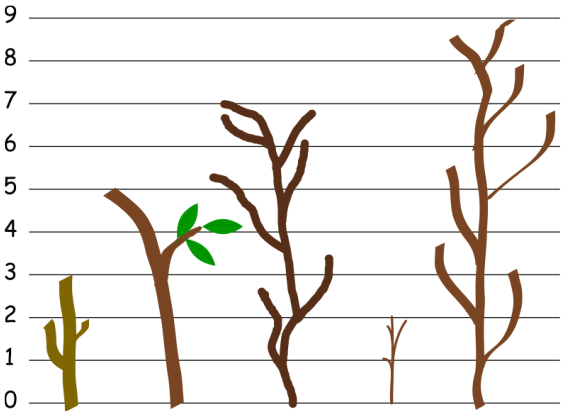
V 3. koraku Jaka primerja vejici dolžin 9 in 7 ter zamenja njuni mesti, saj je 7 manj kot 9. Nov vrstni red je naslednji: 5, 3, 7, 9, 2.



V 4. koraku Jaka primerja vejici dolžin 9 in 2. Ker je 2 manj kot 9, zamenja njuni mesti. Nov vrstni red je 5, 3, 7, 2, 9.



V 5. koraku se Jaka vrne na levo stran vrste in primerja vejici dolžin 5 in 3. Ker je 3 krajša od 5, zamenja njuni mesti. Po 5. koraku je zato vrstni red naslednji: 3, 5, 7, 2, 9.



Računalniško ozadje

V nalogi so predstavljeni koraki urejanja z mehurčki. Urejanje z mehurčki je eden od najbolj znanih algoritmov za urejanje podatkov, a je, kot lahko vidiš v nalogi, precej zamuden.

Skrivno sporočilo

6. razred

Skupina prijateljev uporablja skrivno kodo, s katero nadomešča črke z ujemajočimi števili v abecedi in rezultat prikaže kot črno-belo sliko. Spodnja tabela prikazuje, katero črko predstavlja katera številka:

Črka	A	B	C	Č	D	E	F	G	H	I	J	K	L
Število	1	2	3	4	5	6	7	8	9	10	11	12	13

Črka	M	N	O	P	R	S	Š	T	U	V	Z	Ž	
Število	14	15	16	17	18	19	20	21	22	23	24	25	

Pri kodiranju črke uporabimo vrsto devetih kvadratov, razdeljeno na dva dela (i) in (ii), na naslednji način:

1) Poišči številko črke v abecedi (na primer črka S ima številko 19).
2) Številko črke deli s 5, zaokroži navzdol na celo število in spremeni barvo kvadrata v delu (i) na tem mestu v črno. Izračunaj še ostanek pri deljenju številke črke s 5 in spremeni barvo kvadrata v delu (ii) na tem mestu v črno. Če je rezultat deljenja ali ostanka enak 0, kvadrata ne pobarvaj. Na primer, 19 : 5 = 3 in ostane 4, zato pobarvamo tretji kvadrat v (i) in četrti kvadrat v (ii).

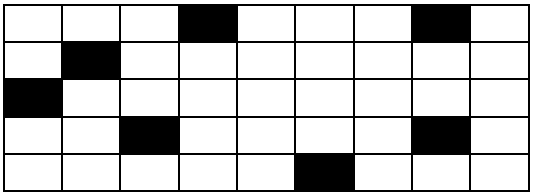
i					ii			
1	2	3	4	5	1	2	3	4
		■						■

Vsako črko sporočila zakodiramo v svoji vrsti. Besedo »BOBER« zakodiramo takole:

	i					ii			
	1	2	3	4	5	1	2	3	4
B							■		
O			■			■			
B							■		
E	■					■			
R			■					■	

Tabelo zakodiranih črk nato pretvorimo v sliko brez glav tabele (le 5 x 9 kvadratkov).

Prijatelj nam je poslal naslednje zakodirano sporočilo:



Kaj piše v sporočilu?

Rešitev

Pravilen odgovor je VIDRA.

Sliki dodajmo še glavo tabele in po vrsti izračunamo številke črk: 23 (4·5+3), 10 (2·5+0), 5 (1·5+0), 18 (3·5+3) in 1 (0·5+1). Številkam ustrezajo črke V, I, D, R in A, po vrsti.

	i					ii			
	1	2	3	4	5	1	2	3	4
V				■				■	
I		■							
D	■								
R			■					■	
A						■			

Računalniško ozadje

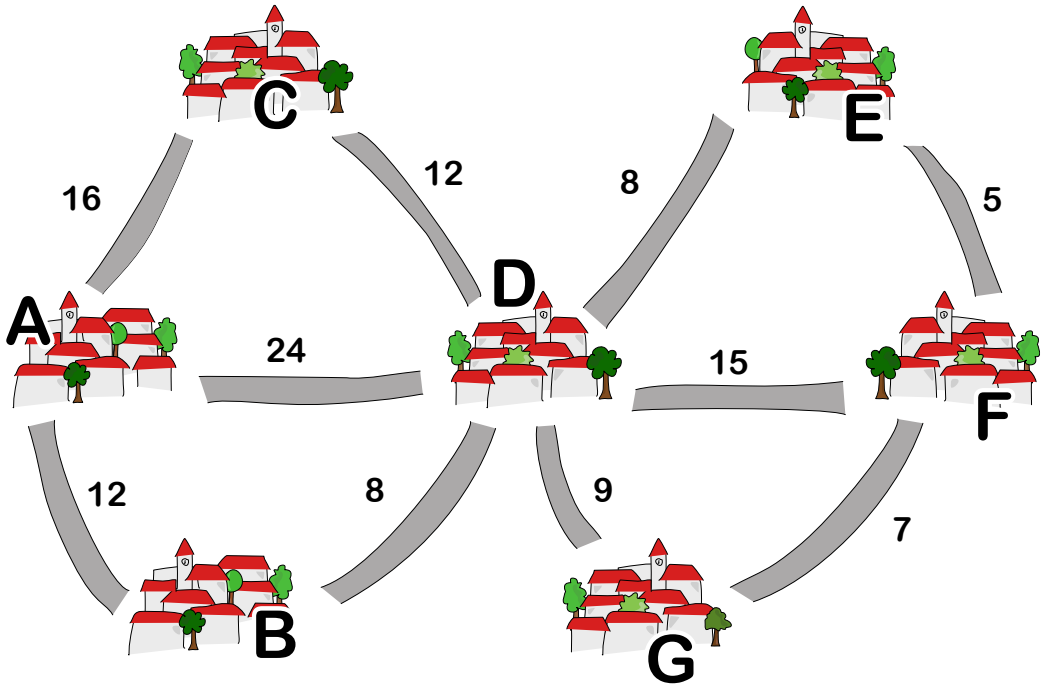
Podoben način kodiranja, kot smo ga uporabili v nalogi, uporablja tudi *Golombova koda*, ki se uporablja za učinkovito stiskanje celih števil, zlasti kadar se manjše vrednosti pojavljajo pogosteje kot večje. Koda temelji na deljenju števila z izbranim parametrom m (v nalogi je $m = 5$), nato pa ločeno zakodira celi del in ostanek (oba sicer kodira precej drugače, kot smo to naredili v nalogi). Golombova koda omogoča zelo enostavno in hitro kodiranje ter dekodiranje, zato se pogosto uporablja pri stiskanju slik, zvoka in videa (npr. FLAC, JPEG).

V računalništvu podatki niso vedno zapisani z znaki kot besedilo ali s številkami. Včasih jih predstavimo tudi vizualno, kar lahko računalnik optično prebere in dekodira. Dva vsakdanja primera take predstavitve podatkov sta črtna koda in QR-koda.

Varčna pot

6. razred

Monika živi v mestu A in želi obiskati svoje stare starše, ki živijo v mestu F. Na sliki so prikazana mesta, ceste in cene prevoza med mesti (v evrih).



Mama ji da za pot 50 evrov. Ker želi Monika od tega prihraniti čim več denarja, bo izbrala najcenejšo pot.

Koliko evrov bo Monika **prihranila**, če izbere najcenejšo pot?

Rešitev

Monika bo do starih staršev gotovo morala potovati skozi mesto D. Do mesta D bo najceneje prišla preko mesta B, za to pot pa bo skupaj odštela 20 evrov. Od mesta D do mesta F pa bo najceneje potovala preko mesta E, za kar bo skupaj odštela 13 evrov. Za celotno najcenejšo pot od A do F bo potrebovala 33 evrov, torej ji bo od 50 evrov ostalo še 17 evrov.

Računalniško ozadje

Naloga prikazuje problem najkrajše poti: med dvema mestoma želimo najti pot z najmanjšim skupnim stroškom. Mesta lahko razumemo kot vozlišča grafa, ceste pa kot povezave med vozlišči, pri čemer ima vsaka povezava svojo ceno (utež). Pri majhnem zemljevidu lahko preverimo več možnosti ročno. Pri večjih omrežjih pa bi bilo preizkušanje vseh poti prepočasno, zato v računalništvu uporabljamo algoritme, ki sistematično iščejo najcenejšo pot.

Avtonomni avtobus

6., 8. in 9. razred

Avtonomni medkrajevni avtobus vozi po poti s postajami od 1 do 5 (v tem vrstnem redu).

Običajno avtobus ustavi na postaji, če tam kdo čaka, da bi vstopil, ali pa želi kdo izstopiti z avtobusa.

Danes pa računalnik avtobusa ne deluje pravilno: avtobus ustavi na postaji samo takrat, ko na postaji kdo čaka, da bi vstopil, in hkrati želi kdo izstopiti z avtobusa. Ne glede na napako avtobus vedno ustavi na prvi (1) in zadnji (5) postaji.

Na postajah čakajo potniki:

- na postaji 1 čaka Maša, ki se želi peljati do postaje 2,
- na postaji 1 čaka Voranc, ki se želi peljati do postaje 4,
- na postaji 2 čaka Simon, ki se želi peljati do postaje 5,
- na postaji 3 čaka Elena, ki se želi peljati do postaje 4,
- na postaji 3 čaka Katarina, ki se želi peljati do postaje 5.

Kdo od čakajočih na postajah zaradi napake v delovanju ne more do svojega cilja?

Rešitev

Do svojega cilja ne morejo Voranc, Elena in Katarina.

Avtobus bo v vsakem primeru ustavil na postajah 1 (kjer se vkrcata Maša in Voranc) in 5. Ustavil bo tudi na postaji 2, saj želi Maša izstopiti, Simon pa vstopiti. Torej bo Maša prišla na svoj cilj.

Na postaji 3 avtobus ne bo ustavil, saj nihče od potnikov na avtobusu tam ne bo izstopil. Elena in Katarina tako ne moreta vstopiti na avtobus, zato ne moreta priti do svojega cilja.

Voranc želi izstopiti na 4. postaji, a tam avtobus ne bo ustavil, saj ni čakajočih potnikov. Torej tudi Voranc ne bo prišel na svoj cilj.

Na zadnji (5.) postaji bo avtobus gotovo ustavil, zato bo Simon prišel na svoj cilj.

Računalniško ozadje

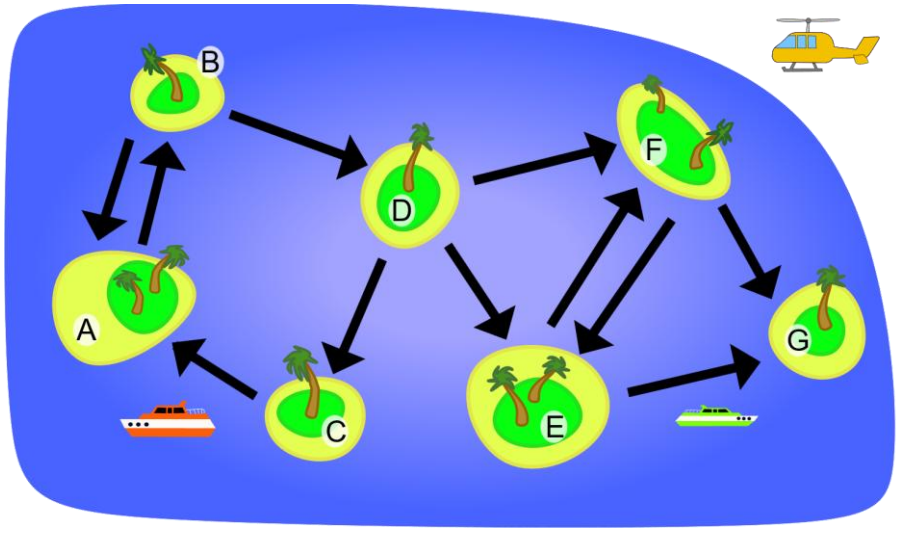
V nalogi smo morali upoštevati razliko med logičnima povezavama ALI in IN. Avtobus običajno ustavi, če je izpolnjen vsaj eden od pogojev (nekdo vstopa ***ali*** nekdo izstopa ***ali*** pa velja oboje). Zaradi napake pa avtobus ustavi le takrat, ko sta oba pogoja izpolnjena hkrati (nekdo vstopa ***in*** nekdo izstopa).

Oskrbovalne odprave

6., 8. in 9. razred

V oceanu leži otočje. Prebivalce vseh otokov je potrebno oskrbeti s hrano in pitno vodo. Ekipa oskrbovalcev lahko s helikopterjem pristane na kateremkoli otoku. Med otoki vozijo trajekti, vendar zaradi močnih tokov med nekaterimi otoki vozijo le v določeno smer, kot to prikazujejo puščice na zemljevidu.

Med posameznim oskrbovalnim izletom ekipa pristane na izbranem otoku, nato pa se lahko poljubno pelje s trajekti. Na koncu se mora vrniti na otok, kjer je pristala s helikopterjem, da se lahko vrne domov.



Najmanj kolikokrat mora ekipa s helikopterjem poleteti na otočje, da bodo oskrbeli prebivalce vseh otokov?

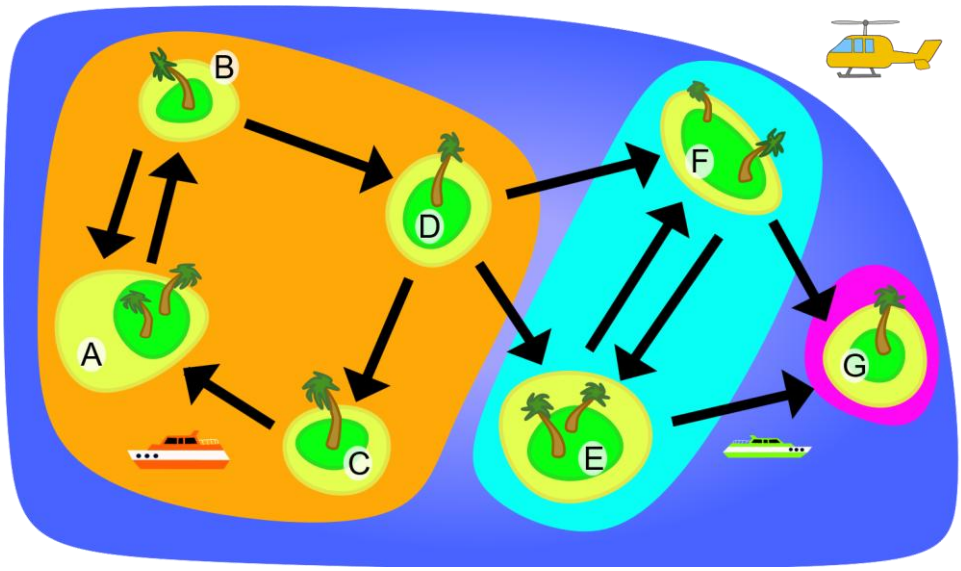
Rešitev

Ekipa mora na otočje prileteti najmanj trikrat.

Ker se mora ekipa po vsakem oskrbovalnem izletu vrniti k helikopterju, lahko med enim izletom oskrbi le tiste otoke, s katerih se je mogoče s trajekti vrniti na začetni otok.

Zato otoke razdelimo v skupine, za katere velja, da je z vsakega otoka mogoče s trajekti priti do vsakega drugega otoka te skupine. Takšno skupino poiščemo tako, da izberemo poljuben otok, ki še ni dodeljen nobeni skupini, in mu dodamo vse otoke, med katerimi je mogoče potovati tja in nazaj.

Na sliki so otoki razdeljeni v tri skupine, označene z različnimi barvami. Ker je potovanje med temi skupinami mogoče le v eni smeri, je za vsako skupino potreben poseben oskrbovalni izlet, torej skupno najmanj trije.



Računalniško ozadje

Barvno označene skupine otokov v razlagi rešitve predstavljajo množice otokov, znotraj katerih je mogoče z vsakega otoka priti do vsakega drugega in se tudi vrniti. V računalništvu takšno skupino imenujemo krepko povezana komponenta (angl. *strongly connected component*, SCC).

Krepko povezane komponente ne opisujejo le povezav med otoki v naši nalogi, temveč se pojavljajo v različnih omrežjih in sistemih, na primer:

- Spletne strani: skupine spletnih strani, ki se med seboj povezujejo z neposrednimi povezavami.
- Družbena omrežja: skupine uporabnikov, ki si med seboj sledijo v obe smeri.
- Prometna omrežja: območja mest ali krajev, med katerimi je mogoče potovati v obe smeri.

Razumevanje krepko povezanih komponent pomaga pri analizi omrežij, priporočilnih sistemih in pri reševanju različnih optimizacijskih problemov.

Robot sadi rože

6., 8. in 9. razred

Robot sadi rože v vrsti v gredico glede na oznake. Mesto je prazno, če nima ne oznake ne rože. Robot sadi po naslednjih pravilih:

0: Pojdi na mesto, označeno z **X**.

Ponavljaj korake 1–5, dokler ne stopiš izven gredice.

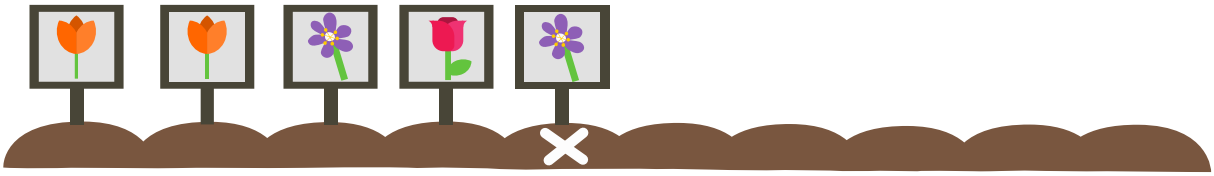
1: Če je na tvojem mestu znak, na to mesto posadi rožo, ki je prikazana na znaku.

2: Zapomni si rožo, ki si jo pravkar posadil.

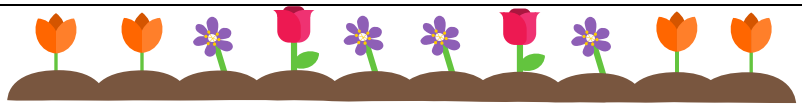
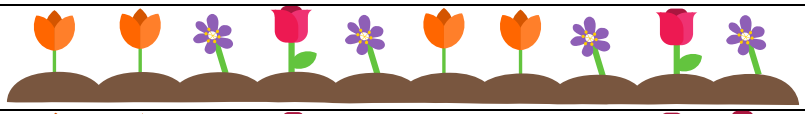
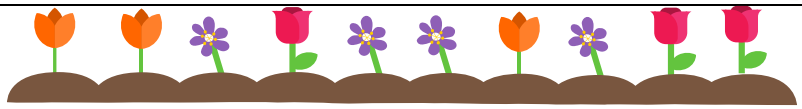
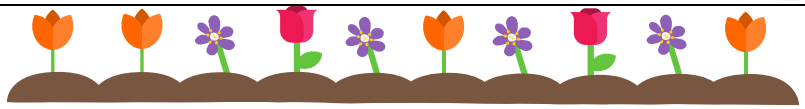
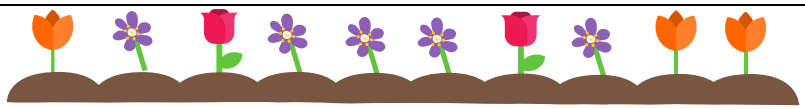
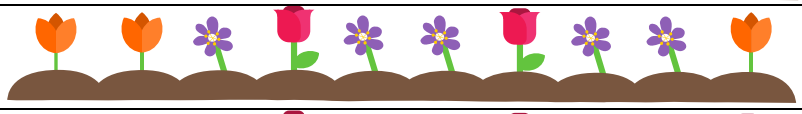
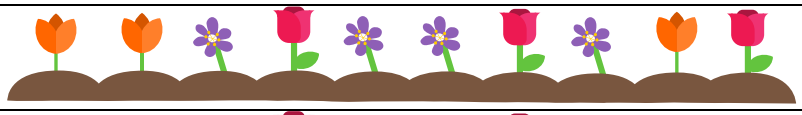
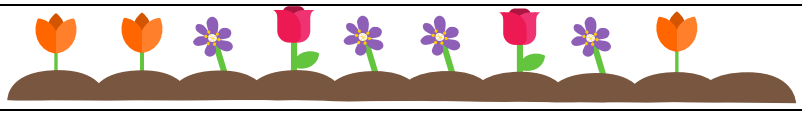
3: Odstrani znak.

4: Premikaj se desno, dokler ne prideš do praznega mesta; tam posadi rožo, ki si si jo nazadnje zapomnil.

5: Premikaj se levo, dokler ne prideš do mesta z znakom ali dokler ne stopiš izven gredice.



Kako bo zgornja gredica videti po končanem sajenju?

A	
B	
C	
D	
E	
F	
G	
H	

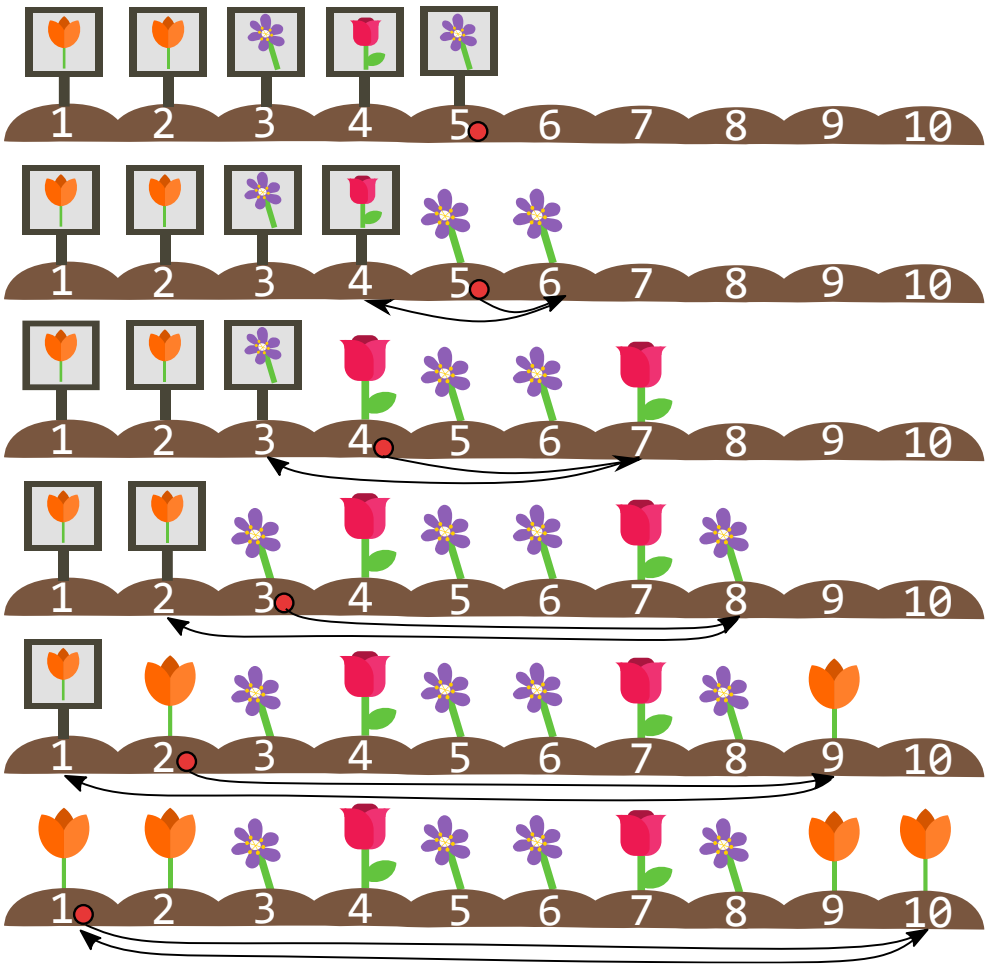
Rešitev

Pravilen odgovor je A. Oštevilčimo mesta v gredici. Robot začne na označenem mestu, to je na 5. mestu. Tam posadi vijolico, si jo zapomni in se premakne v desno. Takoj desno je prazno mesto (6. mesto), zato tja posadi rožo, ki si jo je zapomnil (vijolico). Nato se premika levo, dokler ne pride do prvega mesta z znakom, to je 4. mesto. Tam posadi vrtnico in si jo zapomni.

Robot se nato premakne v desno do prvega praznega mesta, to je 7. mesto, in tja posadi zapomnjeno rožo (vrtnico). Nato se premika levo, dokler ne pride do mesta z znakom, to je 3. mesto. Tam posadi vijolico, si jo zapomni in se premakne desno do prvega praznega mesta (8. mesto), kjer posadi še eno vijolico.

Na 2. in 9. mestu robot posadi tulipana, na 1. in 10. mestu pa še dodatna tulipana.

Opazimo, da robot rože v prazna mesta sadi v obratnem vrstnem redu kot v prvi polovici, kjer so bile oznake. Na sliki rdeča pika označuje trenutni položaj robota, puščice pa prikazujejo njegovo gibanje.



Računalniško ozadje

V tej nalogi je skrit osnovni model Turingovega stroja, enega temeljnih pojmov računalništva. Robot predstavlja glavo Turingovega stroja, ki se premika po traku (vrstici v gredici), razdeljenem na posamezna polja. Na teh poljih lahko robot bere podatke (oznake), jih spreminja (odstrani oznako) in zapisuje nove podatke (posadi rožo). Robot si zapomni, katero rožo je nazadnje posadil. To ponazarja pomnilnik, ki ga ima Turingov stroj.

Sodobna pošta

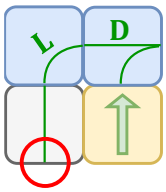
6., 8. in 9. razred

Mihael ima poštno podjetje, v katerem pošiljke po oddelkih prenašajo robotizirani vozički, ki sledijo zaporedju danih ukazov. Ker gre za različne oddelke, ima vsaka soba drugačno razporeditev poti in zidov.

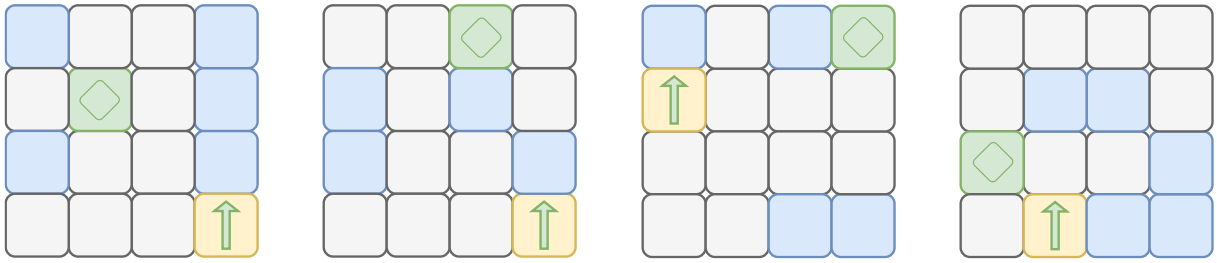
Ko v sobo prispe pošiljka, se pojavi na rumenem polju, kjer jo voziček prevzame. Voziček se nato začne premikati v smeri puščice.

- Ko voziček pride na sivo polje, nadaljuje gibanje v trenutni smeri. Če zadene zid, se ustavi na tem sivem polju.
- Ko voziček pride na modro polje, zavije levo (L) ali desno (D) glede na podan ukaz. Če ga ukaz usmeri v zid, se od zidu odbije in nadaljuje gibanje v nasprotni smeri.
- Ko voziček vstopi na zeleno polje, prispe na odpremno mesto, kjer se pošiljka odda. Voziček se ustavi in vsi preostali ukazi se preskočijo.

Na primer, če vozičku damo ukaza D, L, bi v spodnjem primeru z rumenega polja šel na zgornje desno modro polje, kjer bi zavil desno (D), se odbil od zidu, šel na zgornje levo modro polje, kjer bi zavil levo (L) na sivo polje. Tam bi nadaljeval naravnost, se zaletel v zid in se ustavil.



Sobe različnih oddelkov izgledajo takole:



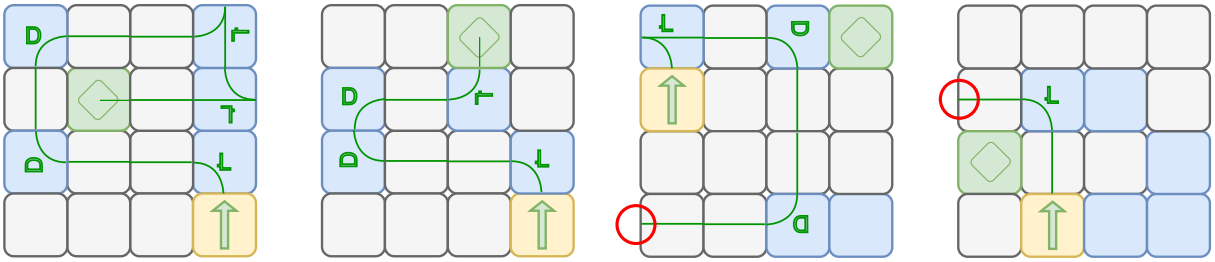
Da bi imel Mihael čim manj dela z vozički, želi za vozičke v vseh sobah uporabiti isto zaporedje ukazov. Katero zaporedje ukazov naj uporabi, da bodo pošiljke v vseh sobah prišle do odpremnega mesta?

A) L, D, D, L, L
B) D, D, L, L, D
C) D, L, D, D, L
D) L, L, D, L, L

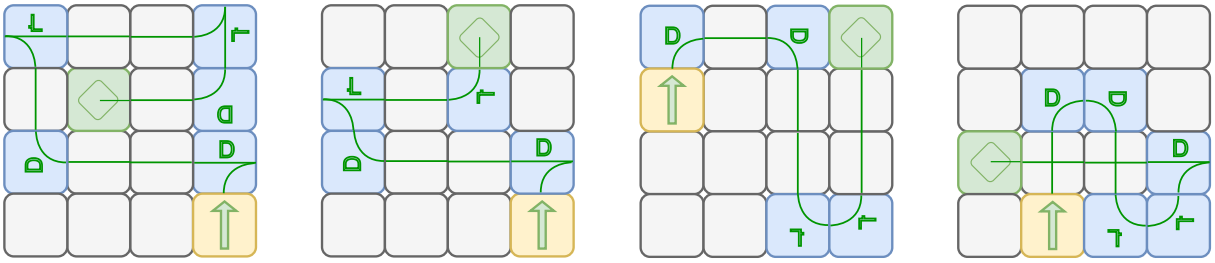
Rešitev

Pravilni odgovor je B. V vseh ostalih primerih se vsaj en voziček zaleti v zid na sivem polju. Prikažimo, kako se gibljejo vozički v vsaki sobi pri vsakem zaporedju ukazov:

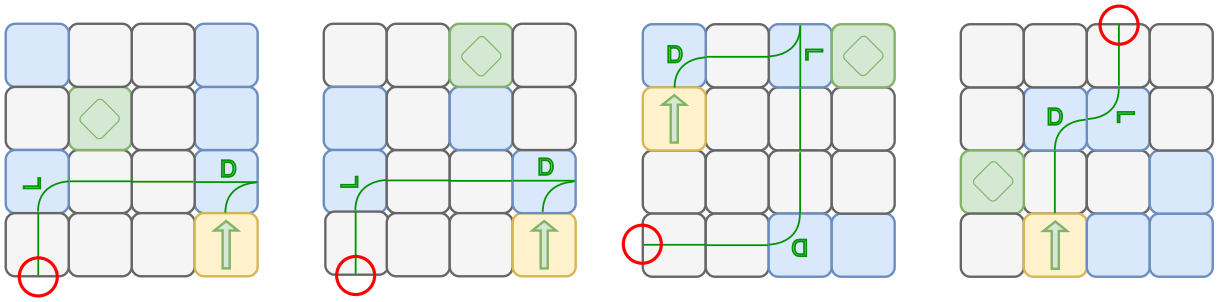
Odgovor A: L, D, D, L, L



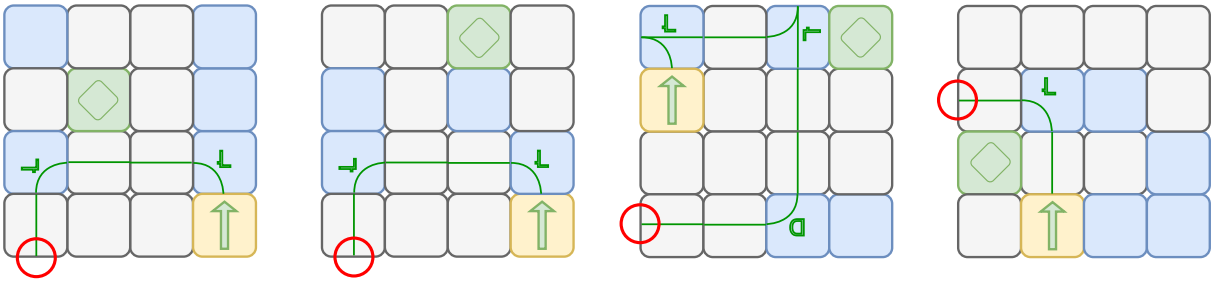
Odgovor B: D, D, L, L, D



Odgovor C: D, L, D, D, L



Odgovor D: L, L, D, L, L



Računalniško ozadje

V računalništvu pogosto ne izumljamo rešitve vsakič znova, ampak uporabimo že znan algoritem in ga preizkusimo v različnih situacijah. V tej nalogi je zaporedje ukazov en algoritem, štiri sobe pa so štirje različni problemi (različni “vhodi”). Če algoritem deluje v vseh sobah, pomeni, da je dovolj splošen in ga lahko uporabimo za več različnih primerov.

Šifriranje v dveh korakih

6., 8. in 9. razred

Tilen šifrira besede z algoritmom, ki ima dva koraka:

1. korak:
   Besedo bere **od leve proti desni**. Vzame črke na lihih mestih (1., 3., 5. …) in jih zapiše skupaj.
   Primer: pri besedi *mreža* dobi v tem koraku *mea*.
2. korak:
   Nato prvotno besedo bere **od desne proti levi** in vse preostale črke (tiste na sodih mestih) po vrsti dodaja na konec.
   Tako se *mreža* na koncu šifrira v *meažr*.

Tilen je ugotovil, da če algoritem ponovi večkrat, lahko spet dobi prvotno besedo.

Primer: če šifrira besedo *svet*, dobi *setv*. Če šifrira dobljeno besedo *setv*, dobi *stve*, če pa šifrira *stve*, dobi spet *svet*. Prvotno besedo dobi po 3 šifriranjih.

Kolikokrat (vsaj enkrat) mora Tilen uporabiti algoritem, da iz besede *trgovec* spet dobi *trgovec*?

Rešitev

Pravilen odgovor je 6.

1. trgovec se šifrira v tgvceor.
2. tgvceor se šifrira v tverocg.
3. tverocg se šifrira v teogcrv.
4. teogcrv se šifrira v tocvrge.
5. tocvrge se šifrira v tcregvo.
6. tcregvo se šifrira nazaj v trgovec.

Računalniško ozadje

V tej nalogi Tilen črk ne zamenja z drugimi črkami, ampak jih samo premeša v drugačen vrstni red. Tak način šifriranja imenujemo transpozicijsko šifriranje. Če poznamo pravilo, lahko besedo šifriramo. Če želimo šifrirano besedo spet spremeniti v originalno, pa uporabimo obraten postopek.

Barvna polja

6., 8. in 9. razred, srednja šola

V zabaviščnem parku v Bobrovem imajo novo atrakcijo – igro na zaporedju LED-polj, ki lahko spreminjajo barvo. Vsako polje prikazuje številko, ki določa število korakov, ki jih mora narediti igralec, ki pristane na tem polju (en korak pomeni premik na sosednje polje). Polja so v treh barvah: črni  , rdeči  in modri  .

Ko igralec pride na polje, se mora naprej premakniti po naslednjih pravilih:

- **Rdeče** polje dovoljuje le premik v **levo**. Ko igralec zapusti polje, to spremeni barvo v modro.
- **Modro** polje dovoljuje le premik v **desno**. Ko igralec zapusti polje, to spremeni barvo v rdečo.
- **Črno** polje pa omogoča igralcu, da sam izbere njegovo barvo:
  - Če izbere rdečo, se polje spremeni v modro, ko ga igralec zapusti.
  - Če izbere modro, se polje spremeni v rdeče, ko ga igralec zapusti.

Igralec začne igro na skrajnem levem polju, cilj igre pa je, da doseže polje z zvezdo na skrajnem desnem. Če igralec ne more priti do cilja ali če se premakne preko prvega polja v levo oziroma preko zadnjega polja v desno, igro izgubi.

Črna polja so na začetku neosvetljena. Igralec pred začetkom igre izbere barvo za vsako črno polje.

Bobrovka Luna bo začela igro, kot prikazuje zgornja slika. Kako naj izbere barve črnih polj, da bo dosegla cilj v čim manj korakih?

Rešitev

Pravilen odgovor je: modra, karkoli, rdeča, karkoli.

Odgovor je prikazan na spodnji sliki:

Nalogo lahko rešimo tako, da preverimo različne možnosti za črna polja.

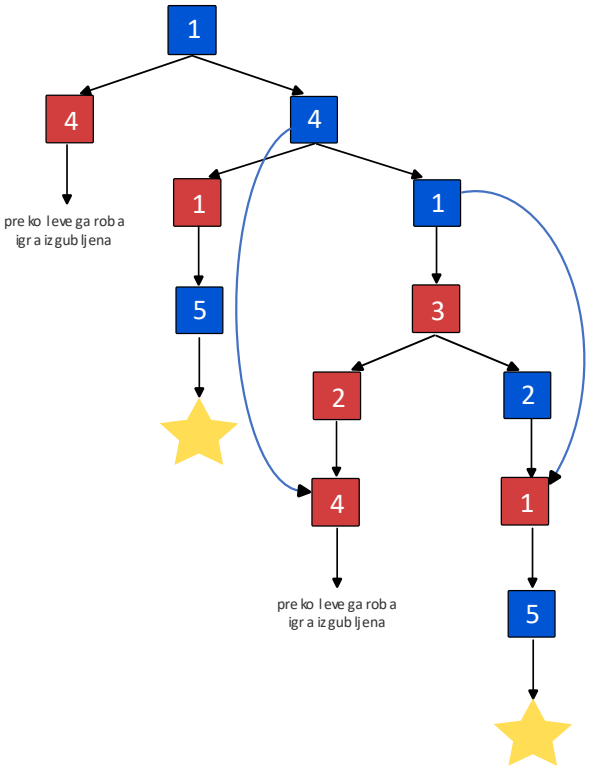
Luna začne igro na prvem polju (modro 1), kar jo pripelje na drugo polje (črno 4). Če bi za to polje izbrala rdečo barvo, bi jo premaknilo štiri korake v levo, kar bi pomenilo izgubo igre. Zato mora biti to polje modre barve (modro 4). To jo nato pripelje na črno polje s številom 1 (črno 1).

Če za črno polje s številom 1 izbere rdečo barvo (rdeče 1), se premakne levo na srednje modro polje (modro 5), to pa jo premakne pet korakov v desno do zvezde.

Ker je srednje modro polje s številom 5 edino polje, ki igralca popelje do polja z zvezdo, torej do cilja, je zgoraj opisana pot tudi rešitev, za katero porabimo najmanj korakov.

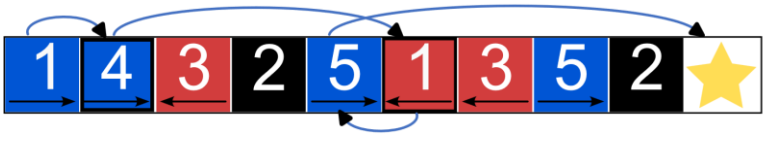
Preostali dve črni polji sta lahko poljubne barve, saj ne vplivata na rešitev v najmanj korakih.

Do rešitve lahko pridemo tudi tako, da sledimo možnim premikom s prvega polja in vsakič, ko naletimo na črno polje, preverimo obe možnosti, rdečo in modro. Takšno sledenje možnim premikom lahko predstavimo z drevesom, v katerem najdemo vse možne poteke igre:



Ker se barva polja spremeni, ko ga igralec zapusti, smo v drevesu to spremembo označili z modro puščico.

Vidimo, da obstajata dve poti, ki pripeljeta do zmage (do polja z zvezdo). Ker iščemo najkrajšo pot (najmanj korakov do cilja), iz drevesa razberemo, da lahko Luna pride do zvezde v 4 korakih, če za prvo črno polje (črno 4) izbere modro barvo, za drugo črno polje, na katero pride (črno 1), pa rdečo barvo. V tem primeru se bo Luna premikala, kot prikazujejo modre puščice:



Obstaja tudi daljša pot do zvezde. Daljša pot večkrat uporabi isto polje (polje 1, ki je sprva modro, nato pa zamenja barvo v rdečo), poleg tega pa uporabi tudi prvo črno polje s številko 2, ki mora biti modre barve, da pot pripelje do cilja.

Računalniško ozadje

Naloga temelji na pravilih, ki jih moramo upoštevati pri reševanju problema. Ta pravila lahko uporabimo za izdelavo algoritma, ki po korakih podaja navodila, kako priti od začetka do zvezde. Smer premikanja je odvisna od barve polja, kar lahko preverimo s pogojnimi stavki:

> ČE je polje rdeče, POTEM se premakni levo.
>
> ČE je polje modro, POTEM se premakni desno.

Na enak način lahko ugotovimo, ali je igralec izgubil, na primer:

> ČE se igralec premakne izven polj, POTEM je igre konec.

Pogojni stavki so eden temeljnih konstruktov programskih jezikov, saj omogočajo odločanje in nadzor poteka izvajanja programa.

Pri reševanju naloge smo si pomagali tudi z *grafom*, ki ga sestavljajo vozlišča (polja) in povezave med njimi (prehodi z enega na drugo polje). Graf je *usmerjen*, saj so prehodi med polji vedno le v eno smer. Uporabili smo posebno vrsto usmerjenega grafa, imenovano *drevo*. Drevo je graf brez ciklov, kar pomeni, da ne obstaja pot, ki bi od vozlišča vodila nazaj do istega vozlišča.

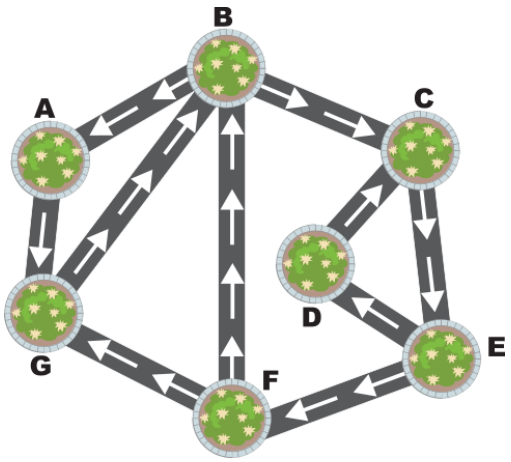
Drevo, ki smo ga zgradili v rešitvi naloge, lahko razumemo kot *odločitveno drevo*, ki prikazuje, kako različne odločitve (izbire barve za črna polja) vplivajo na rezultat (vodijo do različnih izidov, zmage ali poraza). Takšna drevesa se pogosto uporabljajo v umetni inteligenci, bančništvu ali medicini za napovedovanje in sprejemanje odločitev.

Čeprav obstaja več poti do zmage, nas zanima najboljša – tista z najmanj koraki. Temu pravimo *optimizacija*: iskanje najboljše rešitve ob danih omejitvah.

Delo na cesti

6., 8. in 9. razred, srednja šola

Državno tekmovanje letos poteka v središču Maribora na sedmih lokacijah (na spodnjem zemljevidu označenih od A do G). Te so povezane z desetimi enosmernimi ulicami, ki tekmovalcem omogočajo prihod z ene lokacije na katero koli drugo.



Mestna občina namerava med tekmovanjem zaradi obnove popolnoma zapreti eno od ulic. Da ne bi ovirali prihoda tekmovalcev na državno tekmovanje, morajo izbrati tako ulico, da bo dostop do vsake lokacije še vedno mogoč po preostalih ulicah, ob upoštevanju enosmernega prometa.

Katero izmed navedenih ulic naj izberejo za obnovo?

A) Ulico od A do G
B) Ulico od B do C
C) Ulico od C do E
D) Ulico od D do C
E) Ulico od E do F
F) Ulico od F do G
G) Ulico od G do B

Rešitev

Pravilen odgovor je F. Za obnovo morajo izbrati ulico, ki povezuje lokaciji F in G.

Če so vse lokacije povezane in dosegljive iz katere koli druge lokacije po enosmernih ulicah, moramo za zaprto ulico od X do Y preveriti, ali obstaja ustrezen obvoz po ostalih ulicah, ki nas pripelje od X do Y ob upoštevanju enosmernih ulic. Če tak obvoz najdemo, potem vse lokacije ostanejo povezane, četudi zapremo ulico od X do Y.

Poglejmo po vrsti, ali najdemo ustrezne obvoze za navedene ulice.

A) Če zaprejo ulico od A do G, iz A ne bo mogoče priti nikamor.

B) Če je zaprta ulica od B do C, so lokacije C, D in E povezane z lokacijami A, B, F in G le preko enosmerne ulice od E do F, promet v nasprotni smeri pa ni mogoč. To pomeni, da lokacije C, D ali E niso dosegljive iz A, B, F ali G.
C) Če je zaprta ulica od C do E, lokacija E ni dosegljiva, saj le ta ulica pripelje do E.
D) Ob zapori ulice od D do C iz lokacije D ne moremo nikamor drugam.
E) Zapora ulice od E do F povzroči, da lokacija F ni več dosegljiva.
F) Če zaprejo ulico od F do G, lahko iz F vseeno pridemo do G po obvozu preko B in A. Torej lahko na občini zaprejo ulico od F do G, saj to ne vpliva na dostopnost ostalih lokacij.
G) Preverimo še ulico od G do B – v tem primeru iz G ne moremo nikamor, saj je to edina ulica, ki vodi iz G.

Računalniško ozadje

V nalogi smo enosmerne ulice prikazali z *usmerjenim grafom*. Usmerjen graf je sestavljen iz vozlišč (v tej nalogi jih predstavljajo lokacije) in usmerjenih povezav (v nalogi jih predstavljajo enosmerne ulice). Vsaka povezava ima svojo smer: začne se v enem vozlišču in se konča v drugem vozlišču.

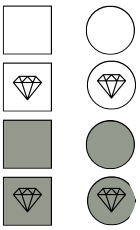
Usmerjen graf je *krepko povezan*, če lahko iz vsakega vozlišča pridemo do vsakega drugega vozlišča, pri tem pa moramo vedno upoštevati smeri povezav. Zemljevid mesta, prikazan v tej nalogi, lahko obravnavamo kot krepko povezan usmerjen graf.

Usmerjeni grafi se v računalništvu uporabljajo za prikaz različnih problemov, kot je na primer vrstni red opravil (kaj mora biti narejeno prej) ali zemljevid mesta (kot v naši nalogi). Če bi v grafu želeli prikazati dvosmerno ulico, bi vozlišči (na primer X in Y) povezali z dvema usmerjenima povezavama: eno iz X v Y in drugo iz Y v X. To bi pomenilo, da ulica poteka v obe smeri.

Tri v vrsto

6., 8. in 9. razred, srednja šola

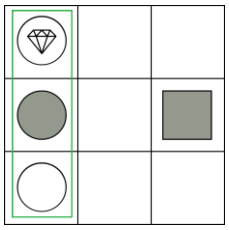
Ana in Bor igrata različico igre tri v vrsto. Na igralno ploščo velikosti 3 x 3 morata razporediti osem žetonov (na desni sliki). Vsak žeton ima tri lastnosti:



- **barvo**: siv ali bel;
- **obliko**: kvadrat ali krog;
- **oznako**: z ali brez diamanta.

Zmaga tisti igralec, ki prvi zapolni vrstico, stolpec ali diagonalo s tremi žetoni, ki imajo eno skupno lastnost (enako barvo ali obliko ali oznako).

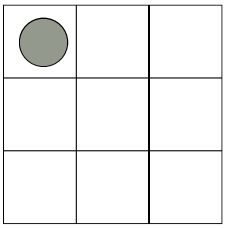
Primer:



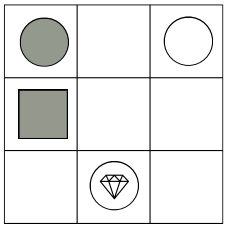
Na levi igralni plošči je zmagal igralec, ki je v prvi stolpec (označen z zelenim pravokotnikom) postavil še tretji žeton okrogle oblike in tako dobil stolpec treh žetonov z enako lastnostjo – krog.

Igra se vedno začne s prazno igralno ploščo. Igralca izmenično polagata žetone na prosta polja igralne plošče, pri čemer žeton, ki ga mora igralec postaviti, vedno izbere nasprotni igralec. Ko igralec postavi žeton na polje, izbere naslednji žeton, ki ga mora na igralno ploščo postaviti nasprotnik. Tako nadaljujeta do konca igre.

Igro je začela Ana z izbiro sivega kroga brez diamanta (●), ki ga mora na igralno ploščo postaviti Bor. Bor je žeton postavil na polje, kot prikazuje desna slika.



Ana in Bor sta nato izmenično naredila več potez, dokler igralna plošča ni bila videti takole:



Katerega od preostalih žetonov naj Ana izbere za Bora, da Bor **zagotovo ne bo** mogel **zmagati** s postavitvijo žetona na igralno ploščo?

A) 

B) 

C) 

D) 

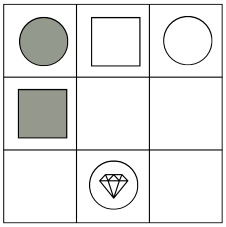
Rešitev

Pravilen odgovor je A.

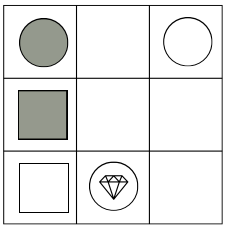
Le izbira žetona pri odgovoru A, to je  , zagotovi, da Bor ne bo mogel sestaviti zaporedja treh žetonov z enakimi lastnostmi.

Odgovor B ni pravilen, saj ima Bor z žetonom  kar dve možnosti za zmago.

Bor lahko postavi žeton na srednje polje prve vrstice in tako dobi tri žetone brez diamanta v prvi vrstici:

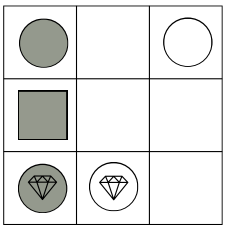


Lahko pa žeton postavi na polje levo spodaj in dobi tri žetone brez diamanta v prvem stolpcu:

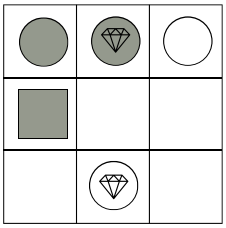


Tudi odgovor C ni pravilen. Pri žetonu  ima prav tako dve možnosti za zmago.

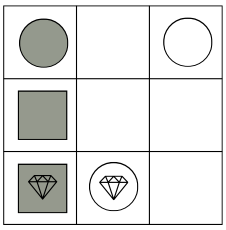
Če žeton postavi na polje spodaj levo, dobi v prvem stolpcu tri sive žetone:



Druga možnost pa je, da žeton postavi na srednje polje prve vrstice in v prvi vrstici dobi tri okrogle žetone:



Tudi odgovor D ni pravilen. Če Bor postavi žeton  na polje spodaj levo, dobi v prvem stolpcu tri sive žetone:



Preverimo še za odgovor A, če postavitev žetona  na katero koli polje res ne vodi do zmage. Možnost za zmago preverimo po posameznih lastnostih žetona. Treh zaporednih žetonov z diamantom ne moremo sestaviti, saj imamo po postavitvi tega žetona na igralni plošči le dva žetona z diamantom. Podobno velja tudi za lastnost oblika: na igralni plošči imamo le dva kvadrata. Torej preostane le še barva – ali lahko sestavimo tri zaporedne bele žetone? Tudi to ne gre, saj sta na igralni plošči pred potezo le dva bela žetona, a sta postavljena tako, da ne moreta biti del istega zaporedja treh žetonov (nista niti v isti vrstici, niti v istem stolpcu, niti na isti diagonali). Torej je z žetonom pri odgovoru A nemogoče sestaviti zaporedje treh žetonov z enako lastnostjo.

Računalniško ozadje

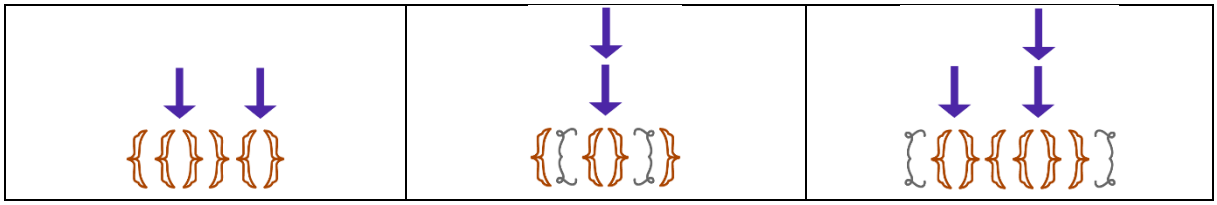
V računalništvu se pri razvoju aplikacij in sistemov pogosto uporablja objektno usmerjen razvoj in objektno usmerjeni programski jeziki. Program gradimo iz objektov, kjer vsak objekt predstavlja neko stvar, lahko realno (tj. fizično, npr. oseba, avto, zgradba) ali abstraktno (tj. pojem ali idejo, npr. igra, račun, šolski predmet). Objekti imajo svoje lastnosti (tako kot žetoni v naši nalogi), tem lastnostim rečemo atributi. Opisujejo stanje objekta, kot so na primer barva, oblika, velikost in podobno. Objekti združujejo podatke (atribute) in obnašanje (metode) ter med seboj sodelujejo. Tak način programiranja pomaga, da so programi bolj pregledni, hkrati pa tudi lažji za razumevanje, dopolnjevanje in popravljanje.

Oklepajoči okraski

7. razred

V zlatarni izdelujejo zapestnice, sestavljene iz parov okraskov v obliki oklepajev. Zapestnico izdelajo tako, da za osnovo vzamejo enega od naslednjih dveh parov okraskov:  ali .

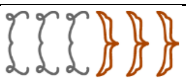
Nato v katerikoli del zapestnice dodajo zaporedoma več parov okraskov, kot kažejo naslednji primeri:



Katero od zapestnic so z opisanim postopkom naredili v zlatarni?

A. 

B. 

C. 

D. 

Rešitev

Pravilen odgovor je D. Začeli so s parom okraskov, nato so v sredino vstavili še en par, naslednji par pa so vstavili še v sredino predhodnega para.

Vse ostale zapestnice ne ustrezajo opisani metodi. Pri odgovoru A je napačna pozicija tretjega okraska, saj je desni del prvega para okraskov pred desnim delom drugega para okraskov. Pri odgovoru B je napačna pozicija prvega okraska: zapestnica se začne z desnim delom para okraskov, namesto z levim delom, kar ni pravilno. Pri odgovoru C pa je napačna pozicija četrtega okraska; imamo tri leve dele enega para okraska in nato tri desne dele drugega para okraska, nikjer pa nimamo nobenih parov.

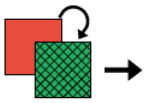
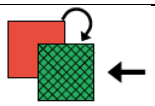
Računalniško ozadje

Pravila, ki jih uporabljajo v zlatarni pri izdelavi zapestnic so enaka pravilom, ki jih uporabljamo za oklepaje pri zapisu izrazov. Računalnikarji rečejo, da so izrazi s pravilno postavljenimi oklepaji dobro sestavljeni, tisti z napakami pa so slabo sestavljeni. Za dobro sestavljeni izraz rečemo, da je sintaktično pravilen, kar pomeni, da ustreza zahtevani sintaksi. Sintaktične napake v kompleksnih izrazih je zelo težko odkriti, čeprav je sintaktične napake praviloma lažje poiskati kot semantične napake, ki se nanašajo na logične napake programerja.

Robot

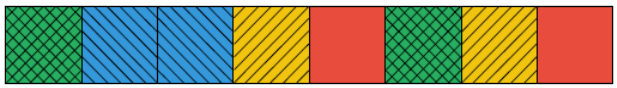
7. razred

Milan je sestavil robota, ki bere barvne kvadrate, spremeni njihovo barvo ter se prestavi za en kvadrat na levo ali desno. Robot deluje po navodilih, kot sta na primer naslednji:

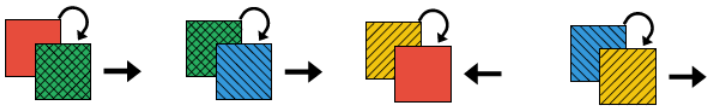
	Če si na rdečem kvadratu, spremeni barvo kvadrata na zeleno in se premakni za en kvadrat v desno.
	Če si na rdečem kvadratu, spremeni barvo kvadrata na zeleno in se premakni za en kvadrat v levo.

Na začetku stoji robot na najbolj levem kvadratu. Robot preveri barvo kvadrata, poišče pravilo, ki ustreza tej barvi, spremeni barvo kvadrata glede na pravilo ter se premakne, kot zahteva pravilo. V naslednjem koraku robot ponovi postopek za kvadrat, na katerem se nahaja. To ponavlja, dokler se ne ustavi. Robot se ustavi, če ne najde ustreznega pravila za kvadrat, na katerem se nahaja, ali če se premakne izven kvadratov.

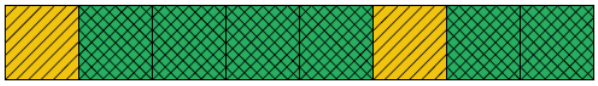
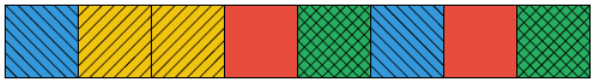
Robotu smo podali naslednje zaporedje kvadratov:



in naslednja pravila:



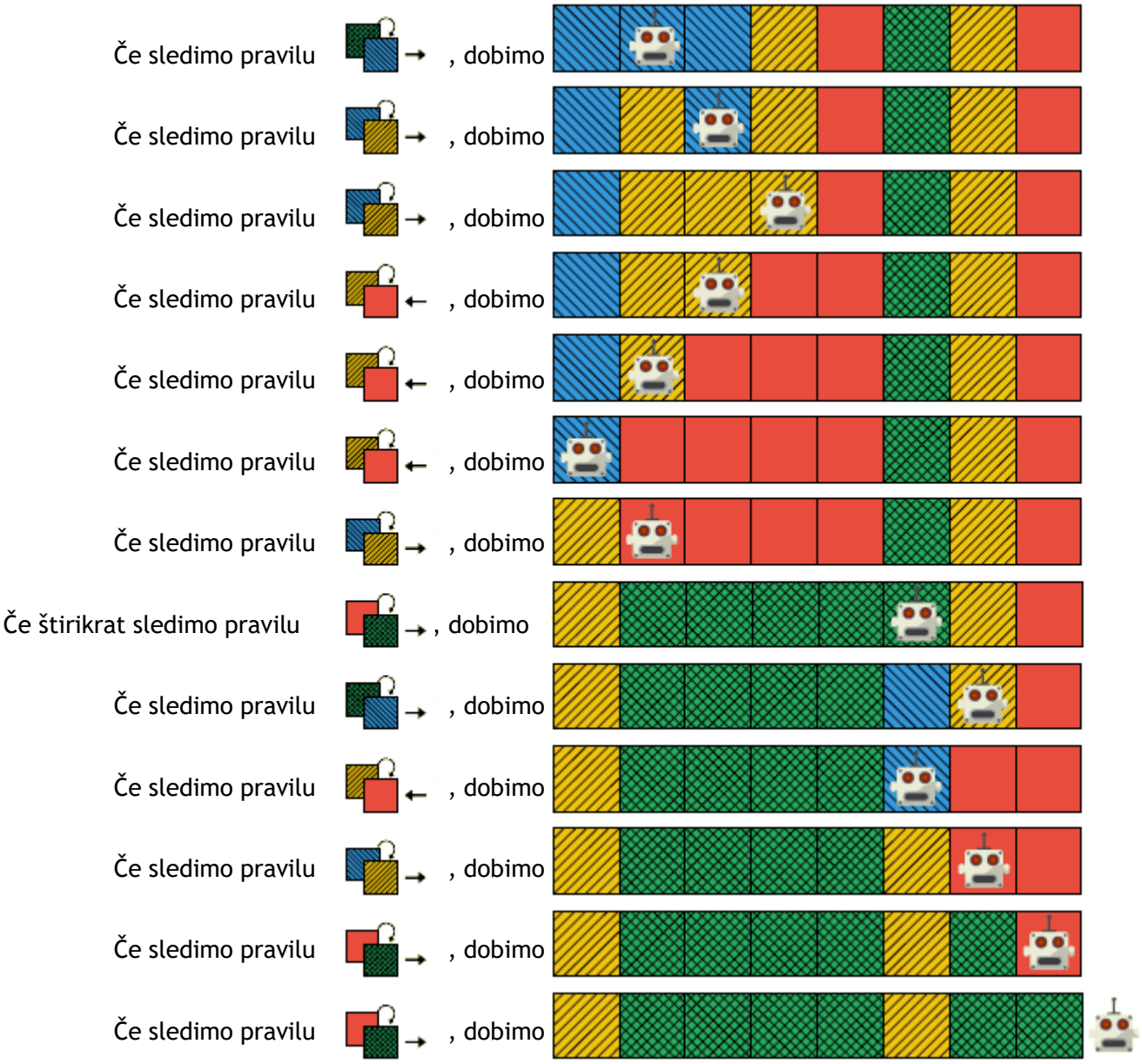
Kako bodo izgledali kvadrati, ko se bo robot ustavil?

A.	
B.	
C.	
D.	

Rešitev

Pravilen odgovor je A.

Začetno stanje: 



Sedaj je robot prišel izven kvadratov, zato se ustavi.

Računalniško ozadje

Naš robot v nalogi se obnaša zelo podobno kot Turingov stroj. Čeprav je zelo enostaven, je Turingov stroj zelo uporaben model računanja za računalnikarje, saj je ekvivalenten večini programskih jezikov. To pomeni, da lahko katerikoli računalniški program spremenimo v Turingov stroj in nasprotno, vsak Turingov stroj lahko pretvorimo v računalniški program.

Skrivna tipkovnica

7. razred

Pri običajni računalniški tipkovnici se ob pritisku na tipko na zaslonu prikaže ustrezna črka.

Pri skrivni tipkovnici pa se ob pritisku na tipko izpiše druga črka. S tem je preprosto napisati skrivna sporočila. Na primer, če pritisneš na tipko za črko Q, se izpiše L.

Če napišeš »JUTRI SE DOBIMO NA PICI«, se izpiše »KIZEU AR FLVUNL MS ŽUŠU«. Kakšna je povezava med črkami?

A.

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

B.

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

C.

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

D.

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

Rešitev

D. Povezave med črkami lahko ugotovimo tako, da gledamo povezavo med posameznimi črkami in najdemo povezave med njimi. Vse črke I se na primer preslikajo v črko U in tako naprej za vse črke v besedilu.

A ni pravilen odgovor, saj se P preslika v Ž in ne v Č, kot je zapisano v tabeli. B ni pravilen odgovor, ker se P preslika v Ž in ne v Š, kot kaže tabela. C ni pravilen odgovor, ker se A preslika v S in ne v F, kot kaže tabela.

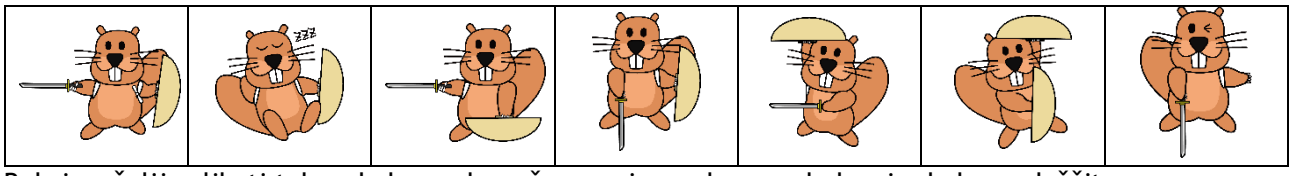
Računalniško ozadje

Ta tipkovnica je podobna Enigmi, ki jo je med 2. svetovno vojno nemška vojska uporabljala za pošiljanje skrivnih sporočil. Enigma je bila nadgrajena še z rotorjem, s pomočjo katerega so se povezave med črkami po vsaki zašifrirani črki še dodatno premešale.

Speči bober

7. razred

Lucija se s sedmimi prijatelji igra igro meč in ščit. Spodnje slike prikazujejo najljubše položaje njenih prijateljev:

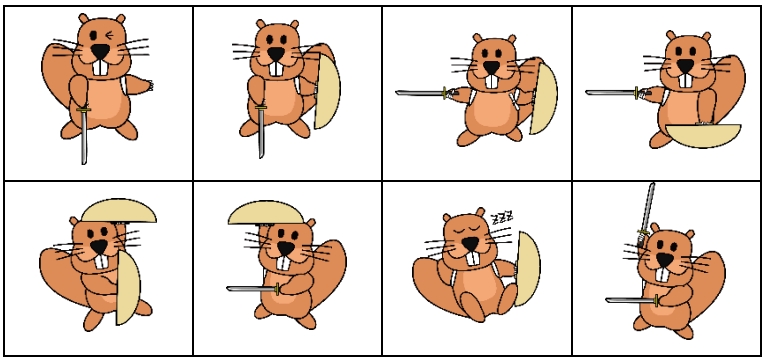


Bobri se želijo slikati tako, da bo vsak meč usmerjen v drugega bobra in da bo vsak ščit preprečil, da bi meč bobra poškodoval. Lucija je že zasedla svoje mesto na sliki:

A	B	C	D
E	F	G	

Kam na sliki se bo postavil speči bober?

Rešitev



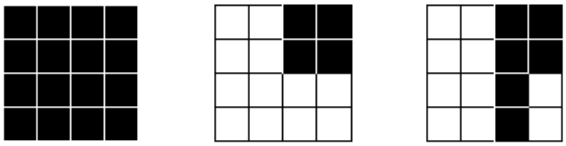
Računalniško ozadje

Tako kot pri sestavljanju slike v tej nalogi, tudi v računalništvu pogosto iščemo zakonitosti, ki jih moramo upoštevati pri reševanju nalog.

Stiskanje slik

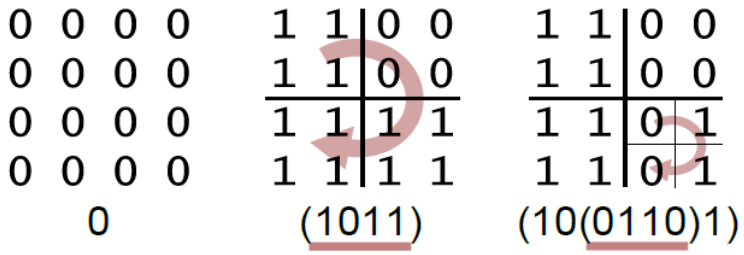
7. razred

Poglejmo naslednje črno-bele bitne slike velikosti 4 x 4:



Te slike lahko shranimo z uporabo binarnih števk: 1 zapišemo za bele piksle, 0 pa za črne. Tako bi za slike velikosti 4 x 4 potrebovali 16 števk.

Vendar pa lahko za shranjevanje takih slik uporabimo tudi drugačno metodo stiskanja, ki omogoča shranjevanje z uporabo manj prostora (predvsem za enostavne vzorce):



Binarne števke so razporejene v mrežo kot piksli na sliki. Metoda stiskanja sestavi rezultat kot niz znakov, pri tem pa uporabi naslednji pravili:

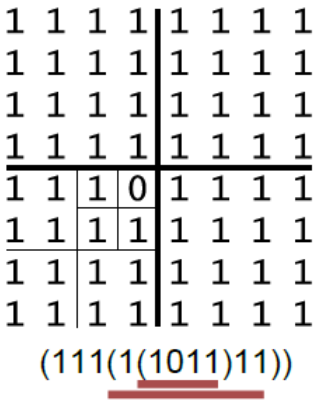
1. Če so vse števke v mreži enake 0, je rezultat niz "0" (glej levo sliko). Če so vse števke enake 1, je rezultat "1".
2. Sicer pa mrežo razdeli na štiri podmreže. Na vsaki podmreži uporabi to metodo stiskanja, po vrsti od zgornje leve podmreže v smeri urinega kazalca. Rezultate stiskanj posameznih podmrež združi z uporabo oklepajev "(" in ")".

Srednja in desna slika prikazujeta dva taka primera stiskanja. Na desni sliki v spodnjem desnem kotu lahko vidimo, da podmreža lahko vsebuje tudi le eno samo števko; v tem primeru uporabimo le pravilo 1.

Za sliko velikosti 8 x 8 je mreža binarnih števk narisana na desni sliki. Kakšen niz dobimo kot rezultat, če na tej sliki uporabimo opisano metodo stiskanja?	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

Rešitev

Mrežo na sliki razdelimo na podmreže, kot prikazuje spodnja slika. Prve tri podmreže velikosti 4 x 4 stisnemo v niz "111", zadnjo podmrežo pa moramo spet razdeliti na štiri 2 x 2 podmreže. Tri od njih lahko zakodiramo z eno številko, drugo podmrežo pa moramo ponovno razdeliti na štiri podmreže velikosti 1 x 1. Niz torej sestavimo takole. Najprej imamo (111X), kjer je X zakodirana četrta podmreža. To zakodiramo kot (1Y11), kjer je Y zakodirana druga podmreža. To pa zakodiramo kot (1011). Če sestavimo celoten niz, dobimo (111(1(1011)11)).

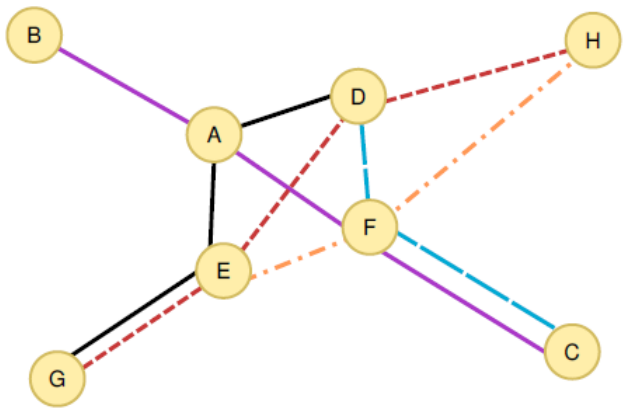


Računalniško ozadje

Čeprav opisana metoda stiskanja deluje nekoliko nenavadno, tak način predstavitve slik uporabljajo tudi bolj resni algoritmi za stiskanje slik. Takemu stiskanju rečemo stiskanje s pomočjo štiri-dreves (angleško *quad-trees*). Štiri-drevo je podatkovna struktura, pri kateri ima vsako notranje vozlišče natanko štiri otroke. Če sliko razdelimo na štiri območja, vsak otrok predstavlja eno območje. Območja lahko nadalje delimo v manjše dele. Pri delitvi slike navadno ne uporabljamo enako velikih delov (kot v našem primeru), saj nam delitev na neenake dele lahko prinese veliko boljše stiskanje.

Nižanje stroškov

7. razred



Železniško podjetje vzdržuje osem železniških postaj in pet železniških povezav, ki jih kaže slika (vsaka povezava je narisana v drugi barvi). Železniško omrežje je zastavljeno tako, da lahko potujemo iz katerega koli kraja v kateri koli drug kraj z največ enim prestopanjem. Tako na primer za pot od B do H lahko vzamemo vijolično povezavo med B in F, tam prestopimo na oranžno povezavo in se peljemo do H.

Železniško podjetje želi znižati stroške poslovanja, zato bodo ukinili eno ali več povezav. Vendar morajo paziti, da vse postaje še vedno ostanejo povezane z železniškim omrežjem in da potovanje s katere koli postaje na katero koli drugo postajo še vedno zahteva največ eno prestopanje.

Največ koliko povezav lahko ukinejo?

Rešitev

Odgovor je največ 2 povezavi (ostanejo 3 povezave).

Če odstranimo rdečo povezavo (GEDH) in modro povezavo (DFC), je še vedno zadoščeno vsem pogojem. Povezave, ki ostanejo (BAFC, GEAD in EFH), namreč očitno povezujejo vseh osem postaj, poleg tega pa tudi velja, da ima vsak par povezav še vedno (najmanj) eno skupno postajo. Torej lahko s katere koli postaje pridemo na katero koli drugo postajo z enim samim prestopanjem.

Ali bi lahko odstranili tri povezave? Oziroma, če vprašanje zastavimo drugače: ali bi še zadostili vsem pogojem, če bi uporabili le dve povezavi? Odgovor je ne. Opazimo lahko, da je vijolična povezava (BAFC) edina, ki pelje do postaje B, zato jo moramo obdržati. Potem mora druga povezava povezovati vse ostale štiri postaje; edina taka možna povezava je rdeča (GEDH). Vendar pa ti dve povezavi nimata skupne postaje, zato z vlakom ne moremo potovati, na primer, od postaje B do postaje H. Torej dve povezavi ne zadostujeta za izpolnjevanje postavljenih pogojev.

Računalniško ozadje

V nalogi smo za predstavitev železniškega omrežja uporabili graf. V računalništvu pogosto uporabljamo grafe za predstavitev kompleksnih problemov.

Zabava

7. razred

Študenti Bobrove akademije organizirajo zabavo, ki bo potekala od 10.00 do 20.00. Ves čas trajanja zabave potrebujejo vsaj enega prostovoljca, ki bo pri vhodu preverjal vstopnice obiskovalcev. Študenti, ki so se javili, da bodo pomagali, so v spodnjo tabelo napisali čas, ko so lahko dežurni.

11.00–12.00	15.30–16.30	19.00–20.00
10.00–10.30	10.15–11.15	19.15–19.30
17.15– 17.45	14.00–15.00	16.15–17.30
18.15–19.00	17.30–19.00	12.00–13.30
13.45–14.30	14.45–16.00	

Ostal je en termin, ko ni na voljo nikogar. Kdaj je to?

Rešitev

Najprej razvrstimo časovne rezine po času začetka:

10.00	10.15	11.00	12.00	13.45	14.00	14.45	15.30	16.15	17.15	17.30	18.15	19.00	19.15
|	|	|	|	|	|	|	|	|	|	|	|	|	|
10.30	11.15	12.00	13.30	14.30	15.00	16.00	16.30	17.30	17.45	19.00	19.00	20.00	19.30

Nato pogledamo časovne rezine in združimo tiste sosednje rezine, ki se prekrivajo. Tako dobimo dve ločeni časovni rezini 10.00-13.30 in 13.45-20.00. Vidimo, da se ni nihče javil, da bi bil pri vhodu v času od 13.30 do 13.45.

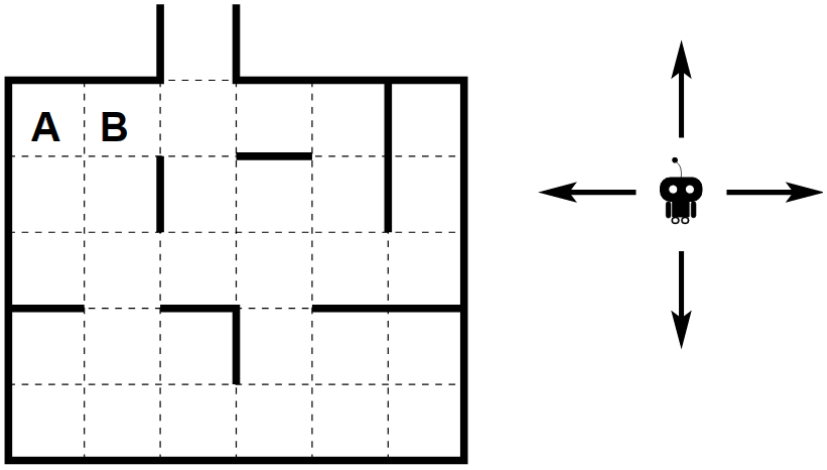
Računalniško ozadje

Nalogo lahko pogosto rešimo hitreje, če podatke prej uredimo po smiselnem vrstnem redu. Tako lažje najdemo iskani element, podvojene elemente in podobno.

Labirint

7. razred

Marko je naredil robota in ga postavil v manjši labirint z neprehodnimi stenami – označujejo jih debelejše črte na sliki. Črtkane črte na sliki pa označujejo mrežo, ki določa možne pozicije robota.



Marko poda robotu zaporedje ukazov. Na ukaz se robot odzove s premikanjem v eno od smeri, ki jih nakazujejo puščice. Robot se premika, dokler ne pride do stene (ali pa ven iz labirinta). Ko se robot začne premikati, ga ne moremo več ustaviti in mu podati drugega ukaza, dokler se sam ne ustavi ob steni.

Če je na primer robot v kvadratu A, se lahko premakne štiri kvadrate v desno ali pa dva kvadrata navzdol, ne more pa se začeti premikati in se ustaviti na sredi poti, na primer v kvadratu B.

Marko želi ugotoviti, ali njegov robot lahko pride iz labirinta, če mu poda primerno zaporedje ukazov. To ga zanima za dve različni začetni poziciji v labirintu, to sta kvadrata A in B.

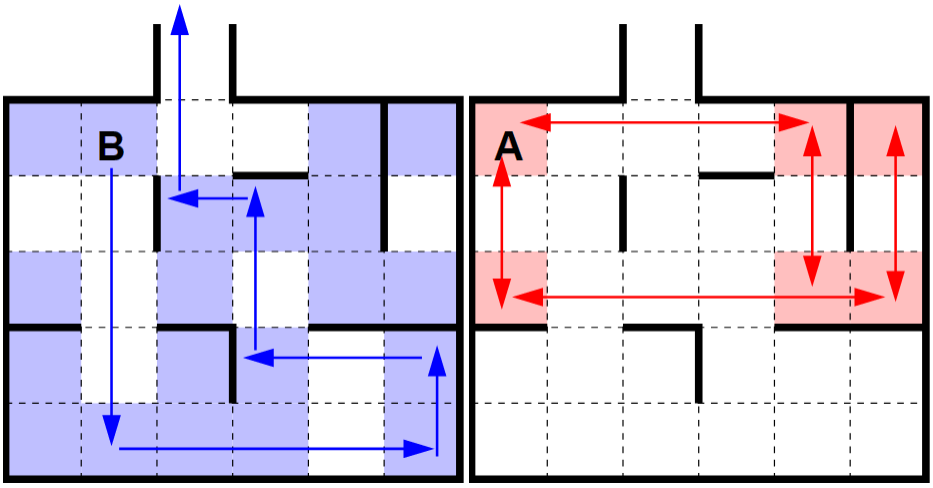
Katera od spodnjih trditev drži?

A. Robot lahko pride iz labirinta z obeh začetnih pozicij A in B.
B. Robot lahko pride iz labirinta z začetne pozicije A, ne pa tudi z B.
C. Robot lahko pride iz labirinta z začetne pozicije B, ne pa tudi z A.
D. Robot ne more priti iz labirinta z nobene od začetnih pozicij A ali B.

Rešitev

Pravilen odgovor je C: robot lahko pride iz labirinta, če začne na poziciji B, a ne z začetne pozicije A.

Rešitev je prikazana na slikah.



Modre puščice na levi sliki prikazujejo premike z začetne pozicije B, ki vodijo do izhoda iz labirinta (torej zaporedje ukazov, ki uspešno pripelje robota iz labirinta). Modro obarvani kvadrati prikazujejo vse pozicije, kjer se robot lahko ustavi.

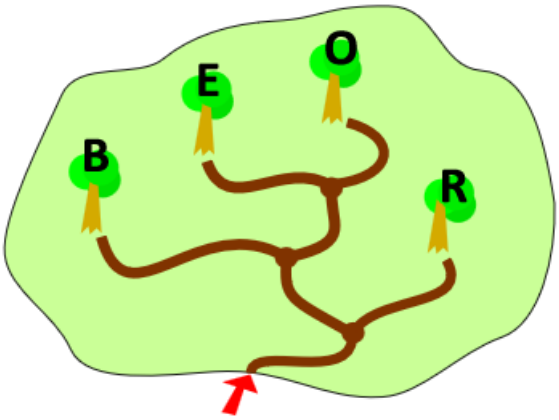
Podobno so na desni sliki prikazane rdeče puščice, ki pomenijo vse možne premike robota z začetne pozicije A. Tudi na tej sliki so rdeče obarvani kvadrati tisti, kjer se robot lahko ustavi.

Računalniško ozadje

Programiranje robotov je dober uvod v programiranje, saj lahko jasno vidimo, kaj se zgodi na vsakem koraku. Principi pa so enaki kot pri drugih vrstah programiranja: vsak ukaz v računalniškem programu na nek način spremeni »stanje« računalnika, naš cilj pa je doseči želeno stanje, na primer dobiti rezultat izračuna ali prikazati sliko na zaslonu.

Levo-desno

7. razred



Za kodiranje svojih sporočil uporabljajo bobri kodo, ki temelji na zemljevidu njihovega osrednjega parka (glej sliko). Vsako drevo v parku je označeno z eno črko. Kodo za vsako črko dobimo tako, da sledimo poti od vhoda v park do ustreznega drevesa, na vsakem razpotju pa zavijemo levo (L) ali desno (D).

Tako ima na primer črka B kodo LL, saj moramo dvakrat zaviti levo, če želimo priti od vhoda v park do drevesa B. Kodo za besedo pa sestavimo iz kod za posamezne črke. Koda za besedo OBE je tako LDDLLLDL.

Katero besedo smo zakodirali s kodo LLLDDD?

Rešitev

Odgovor je BOR.

V tabeli smo zapisali kode levo-desno za vse črke na drevesih v parku:

B	E	O	R
LL	LDL	LDD	D

Če želimo odkodirati besedo LLLDDD, poiščemo najprej prvo zakodirano črko in njeno kodo. Ker nobena črka nima kode L, mora biti prva črka zakodirana z najmanj dvema znakoma in edina možnost je LL, torej je to črka B. Podobno odkodiramo še preostanek besede, to je LDDD. Druga črka je zakodirana v LDD, torej je to črka O. Ostane nam še koda D, v katero se zakodira črka R. Iskana beseda je torej BOR.

Računalniško ozadje

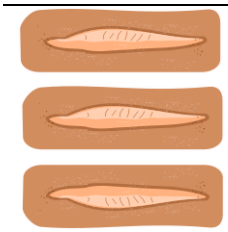
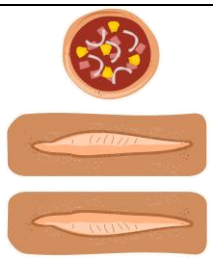
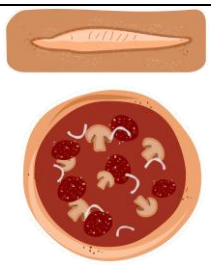
Če bi črko L zamenjali z 0 in črko R z 1, bi dobili dvojiški zapis. V tem primeru se zemljevid parka spremeni v nekaj, kar računalnikarji imenujemo *binarno drevo*. Tako lahko dolgo in zapleteno pot zelo enostavno shranimo v računalniku in pri tem porabimo tudi le malo prostora.

Pri tej kodi je zanimivo dejstvo, da nam za uspešno dekodiranje sporočila ni potrebno vedeti, kje se začne oz. konča koda posamezne črke (torej ne potrebujemo v kodi nobenih presledkov ali posebnih oznak, posledično pa je zakodirana beseda lahko še krajša). To je zato, ker smo kode izbrali tako, da se nobena koda ne začne s katero koli drugo kodo. Tako kodo imenujemo *prefiksna koda* in se v računalništvu pogosto uporablja.

Picerija

7. razred

V piceriji imajo majhno peč za pico, zato lahko picopek hkrati speče le nekaj jedi. Spodaj lahko vidimo vse možne kombinacije in čas peke.

		
3 ciabatte	1 majhna pica in 2 ciabatti	1 ciabatta in 1 velika pica

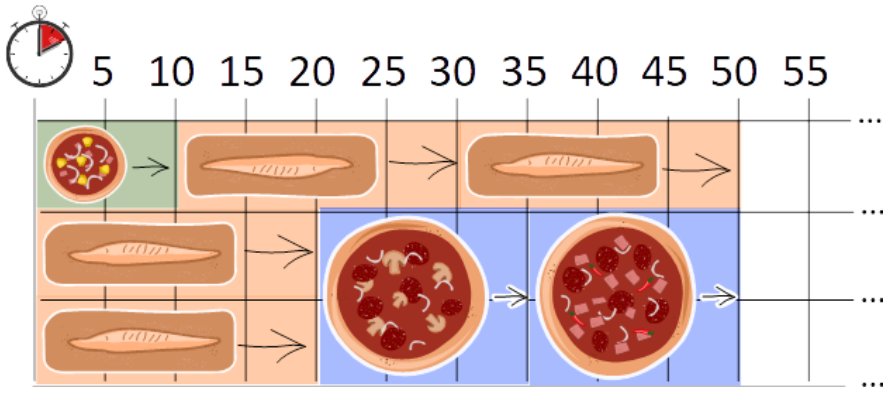
Čas peke:

Majhna pica 10 minut
Velika pica 15 minut
Ciabatta 20 minut

V piceriji imajo veliko gostov, zato mora picopek pametno načrtovati peko, da gostje lahko dobijo hrano, kolikor hitro je to mogoče. Ciabatte in pice lahko položi v peč v poljubnem vrstnem redu. Vsaka posamezna jed mora ostati v peči, dokler ni do konca pečena.

Gostje naročijo eno majhno pico, dve veliki pici in štiri ciabatte. Najmanj koliko časa bo minilo, preden bo celotno naročilo pečeno?

Rešitev



Obstaja več enako dobrih rešitev. Ključno je, da ugotovimo, da traja enako dolgo, če zaporedoma spečemo dve ciabatti in malo pico ali če zaporedoma spečemo dve veliki pici in dve ciabatti, ki se pečeta istočasno. Za oboje je potrebnih 50 minut, čas in prostor v peči sta tako polno zasedena, zato hitrejše rešitve ni. Ena od možnih rešitev je prikazana na sliki.

Računalniško ozadje

V računalniku ves čas poteka razvrščanje procesov. Picopekovo nalogo v računalniku prevzame razvrščevalnik.

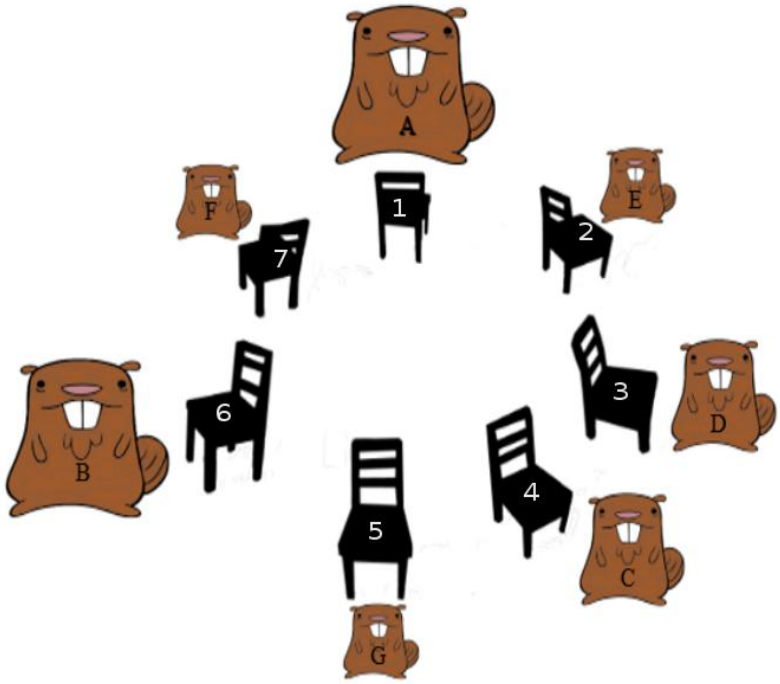
Presedanje

7. razred

Bobri se postavijo vsak k svojemu stolu, kot kaže slika.

Nato se začnejo premikati:

- velika bobra se v vsakem koraku igre pomakneta za tri stole v nasprotni smeri urinega kazalca,
- srednja bobra se v vsakem koraku premakneta za dva stola v nasprotni smeri urinega kazalca,
- mali bobri se premikajo na sosednji stol v smeri urinega kazalca.



Na istem stolu lahko sedi tudi več bobrov hkrati.

Naredili bodo tri korake igre. Katera stola bosta tedaj prosta?

Rešitev

2 in 7.

Velika bobra se premakneta za 9 korakov. A se premakne z 1 na **6** in B se premakne s 6 na **4**.

Srednja bobra se premakneta za 6 korakov: C s 4 na **5** in D s 3 na **4**.

Mali bobri se premaknejo za 3 korake: E z 2 na **5**, F s 7 na **3**, G s 5 na **1**.

Zasedeni so torej stoli 1, 3, 4, 5 in 6. Prosta sta 2 in 7.

Računalniško ozadje

Nalogo bi lahko reševali tako, da bi opazovali, kaj se dogaja po vsakem koraku. A to bi bilo težje. Če opazimo, da se vsak bober giba neodvisno od ostalih, lahko namesto posameznih korakov opazujemo posamezne bobre.

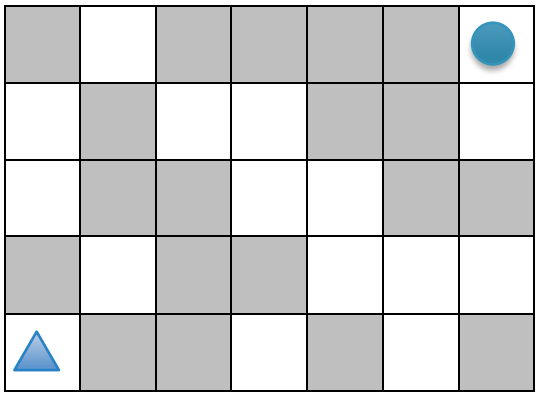
Računalniških problemov se je potrebno lotiti na pravi način, pa je rešitev preprostejša.

Rušenje zidov

7. razred

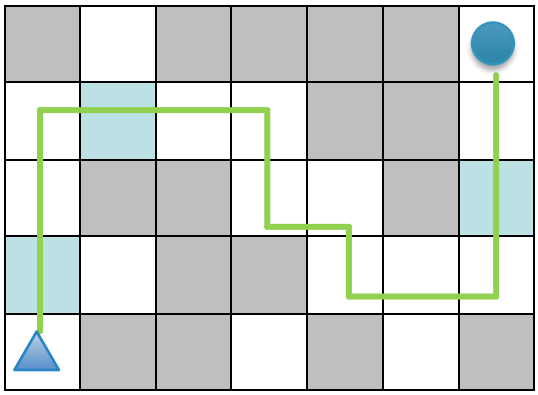
Labirint je sestavljen iz praznih polj (beli kvadrati) in zidov (sivi kvadrati).

Premikamo se lahko le po praznih poljih, ki so navpično ali vodoravno povezana med seboj.



Najmanj koliko zidov je potrebno porušiti, da se lahko po labirintu iz spodnjega levega kota sprehodimo v zgornji desni kot?

Rešitev



Zidovi, ki jih je potrebno porušiti, so označeni z rdečo barvo.

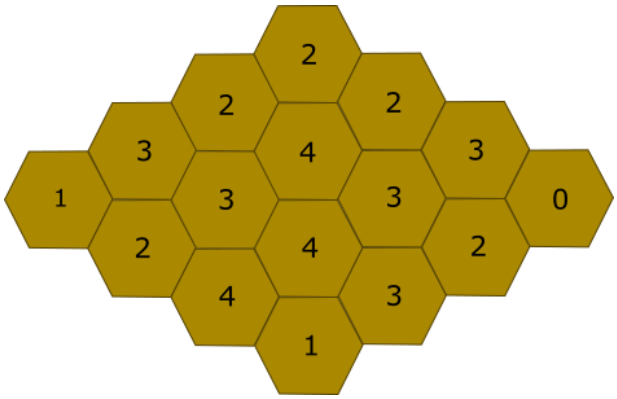
Računalniško ozadje

Za sistematično rešitev naloge je potrebno algoritmično razmišljanje. Za vsako polje je potrebno pogledati, koliko zidov je potrebno porušiti, da bi prišli do posameznega prostega polja.

Satovje

7. razred

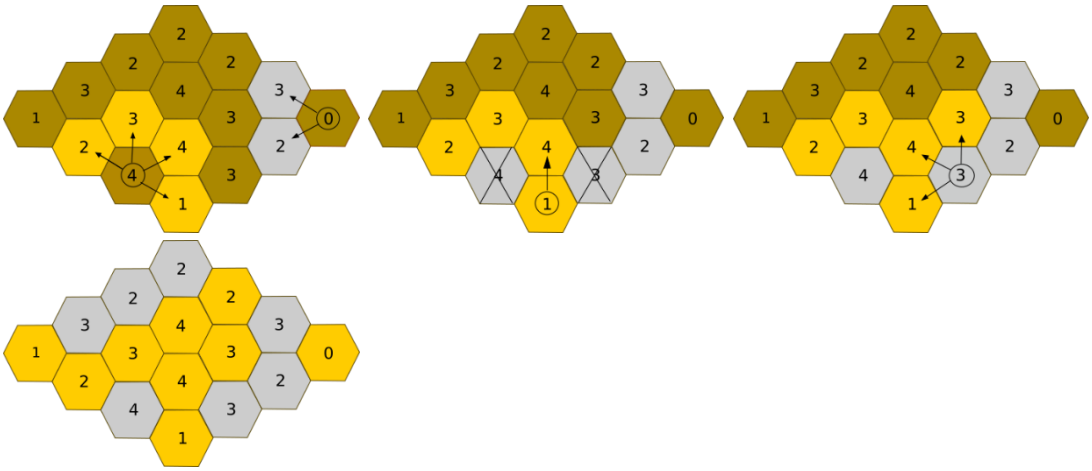
Manca ima rada med. Dedek ji je prinesel satovje. Nekateri deli so polni, nekateri prazni. Na vsakem delu je zapisano, koliko izmed sosednjih delov je polnih.



Koliko delov je polnih?

Rešitev

Začni pri delu, ki ima vse sosednje dele polne, in delu, ki ima vse sosednje dele prazne. Potem lahko postopno ugotoviš, kateri deli so še polni in kateri prazni. Polnih je 9 delov.



Računalniško ozadje

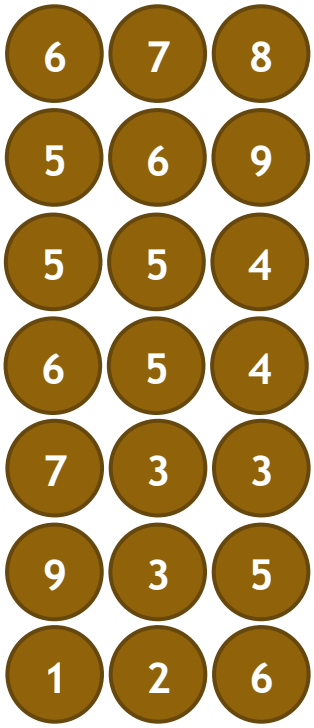
Da si lahko rešil nalogo, si si moral izmisliti primeren postopek reševanja. Računalnikarji temu rečejo *algoritem*.

Skakanje po štorih

7. razred

Katja se na Bobrovem vrtu igra na drevesnih štorih. Z vsakega štora opazuje štiri sosednje štore (spredaj, zadaj, levo in desno) in sledi tem trem pravilom:

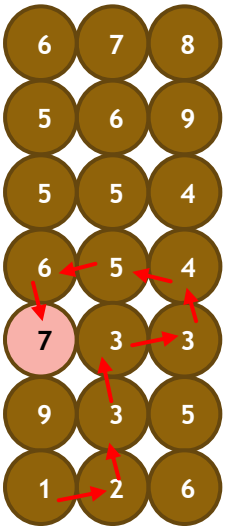
1. Če je eden od sosednjih štorov za ena višji od tistega, na katerem stoji, skoči nanj.
2. Če ne, skoči na štor, ki je enako visok, ampak samo, če na njem še ni bila.
3. Če se s trenutnega štora ne more premakniti po prejšnjih dveh pravilih, se ustavi.



Višine in mesta, na katerih so posamezni štori, so na sliki.

Katja začne na štoru z višino 1. Kako visoko bo priskakala?

Rešitev



Računalniško ozadje

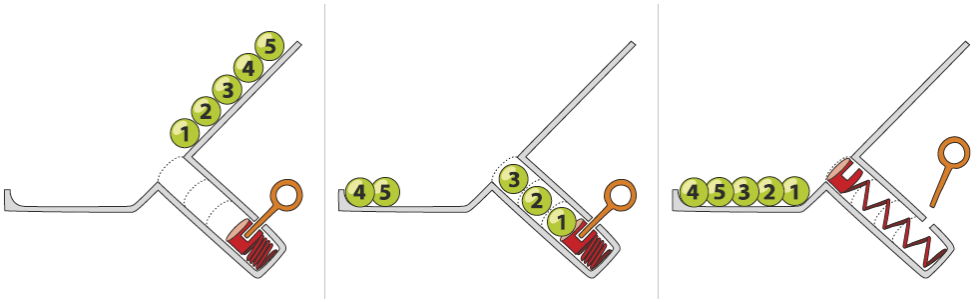
Vsak vidi, da obstaja pot, po kateri lahko pridemo do štora z višino 9 zgoraj desno, Katja pa gleda le en korak naprej in ne celotne slike, zato tega ne ve. V računalništvu marsikdaj napišemo program, ki najde rešitev na podoben način, tako da upošteva le en korak naenkrat. Take algoritme imenujemo požrešni algoritmi, saj požrešno izbirajo tisto, kar je v posameznem koraku videti kot najboljša rešitev. A kot vidimo, to marsikdaj ne pripelje do optimalnega rezultata.

Zakaj potem to počnemo? Zakaj programi ne pogledajo »celotne slike«? Ker je število možnih poti, ki bi jih morali preiskati, lahko tako veliko, da noben računalnik ne bo nikoli uspel pregledati vseh. Zato se moramo zadovoljiti s postopkom, ki včasih da le "dovolj dobro" rešitev.

Žogice

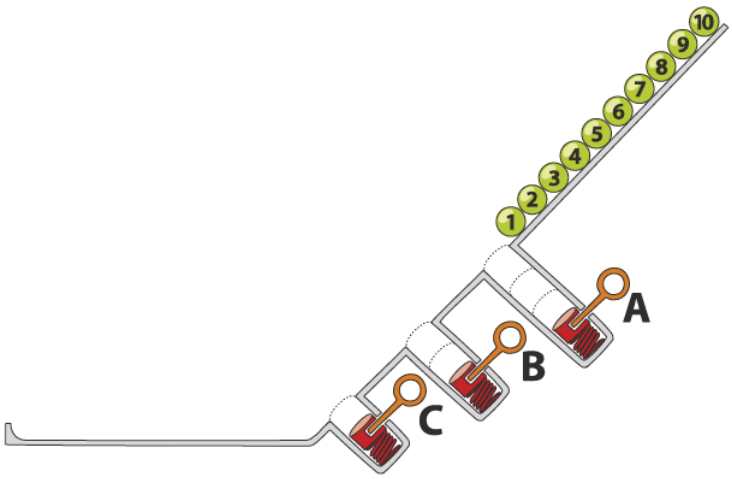
7. razred

Oštevilčene žogice spustimo po klančini. Žogice lahko padejo v luknje na klančini in tako se vrstni red žogic spremeni. Ko žogica pride do luknje, pade vanjo, če je še dovolj prostora. Če prostora v luknji ni, se žogica odkotali dalje. Vsaka luknja ima na dnu tudi zatič; če ga odstranimo, sprožimo vzmet, ki vrže žogice iz luknje. Poglejmo primer:



stanje, preden pet žogic spustimo po rampi	ko se žogice ustavijo	končno stanje, potem ko smo odstranili zatič

Po klančini spustimo deset žogic. Na klančini so tri luknje, A, B in C, v katerih je prostora za 3, 2 in 1 žogico, po vrsti (glej spodnjo sliko). Zatiče odstranjujemo po vrsti, najprej A, nato B in na koncu C, a vedno najprej počakamo, da se žogice nehajo kotaliti.



Kakšen bo končni vrstni red žogic?

A.		C.	
B.		D.	

Rešitev

Pravilen odgovor je D: 

Luknja A ima prostora za tri zogice, zato zogice od 1 do 3 po vrsti padejo vanjo, žogice od 4 do 10 po vrsti pa se odkotalijo mimo nje. Luknja B ima prostora za dve žogici, zato žogici 4 in 5 po vrsti padeta vanjo, žogice od 6 do 10 po vrsti pa se odkotalijo mimo nje. Luknja C ima prostor le za eno žogico, vanjo pade žogica 6, žogice od 7 do 10 po vrsti pa se odkotalijo mimo nje. Torej so prve štiri žogice na dnu tiste s številkami od 7 do 10 po vrsti. Nato odstranimo zatič pri A, kar izvrže žogice 3, 2 in 1 po vrsti. Te tri žogice se zakotalijo do dna, kjer imamo sedaj žogice 7, 8, 9, 10, 3, 2, 1. Nato odstranimo še zatič pri B ter izvržemo žogici 5 in 4, po vrsti. Tudi ti dve žogici se odkotalita do dna, kjer imamo sedaj naslednje zaporedje žogic: 7, 8, 9, 10, 3, 2, 1, 5, 4. Na koncu odstranimo še zatič C in žogica 6 se odkotali kot zadnja do dna. Zaporedje žogic na dnu je sedaj 7, 8, 9, 10, 3, 2, 1, 5, 4, 6, kar je tudi pravilen odgovor.

Računalniško ozadje

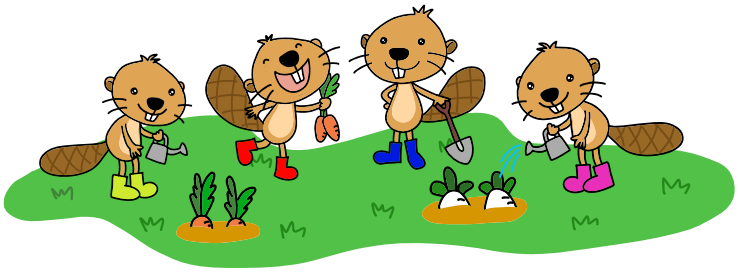
Luknja na klančini se obnaša kot *sklad*, ki je zelo uporabna *podatkovna struktura* (to je, način organiziranja podatkov). Sklad deluje po principu zadnji noter, prvi ven (angleško *last-in first-out* ali krajše *LIFO*). Prva žogica, ki je padla v luknjo, je bila iz nje izvržena zadnja. Čeprav je sama ideja sklada zelo preprosta, lahko sklad uporabimo v mnogih različnih situacijah.

Primer uporabe sklada je pri preverjanju ustrezno postavljenih oklepajev v izrazih: npr. izraz ((1+2)*3) ima pravilno postavljene oklepaje, izraz ((4+5)*(6-7) pa ne. Algoritem preverjanja ujemanja oklepajev je naslednji: vse predklepaje postavimo na sklad (operaciji dodajanja na sklad rečemo angleško *push*) in ko najdemo ujemajoči zaklepaj, s sklada odstranimo en predklepaj (operaciji odstranjevanja rečemo angleško *pop*). Če je na koncu sklad prazen in če smo lahko uspešno izvedli vse operacije pop, se oklepaji v izrazu ujemajo. Tako smo namesto kompliciranega algoritma za preverjanje ujemanja oklepajev uporabili le ustrezen način za organiziranje podatkov – zadostovala je le uporaba principa zadnji noter, prvi ven.

Dure

8. in 9. razred

V Hanini vasi pripravljajo tradicionalni Dure. Dure je korejski sistem sodelovanja, pri katerem vaščani skupaj opravljajo kmetijska opravila.



V vasi morajo vsak teden določiti tri dni za Dure ob upoštevanju naslednjih pravil:

1. Na vsak Dure dan morajo sodelovati najmanj štirje vaščani.
2. Vsak vaščan mora sodelovati vsaj en Dure dan.
3. Nihče ne sme delati vse tri Dure dni.

Razpoložljivost vaščanov je prikazana v spodnji tabeli (O pomeni, da je oseba na voljo).

ime	ponedeljek	torek	sreda	četrtek	petek	sobota	nedelja
Ain	O		O		O	O	
Boa	O	O	O				
Chaewon		O			O		
Doyun			O	O		O	
Eunwoo	O			O			O
Felix		O		O		O	
Gaon	O		O				O
Hana		O			O	O	

Na katere dni naj bo razporejen Dure?

A) ponedeljek, torek, sreda
B) ponedeljek, torek, sobota
C) ponedeljek, sreda, sobota
D) torek, sreda, sobota

Rešitev

Pravilen odgovor je B. Dure morajo razporediti na ponedeljek, torek in soboto.

Glede na tabelo z razpoložljivostjo vaščanov lahko za vsak dan v tednu preštejemo, koliko vaščanov je tisti dan na voljo:

- Ponedeljek: Ain, Boa, Eunwoo, Gaon = 4 vaščani
- Torek: Boa, Chaewon, Felix, Hana = 4 vaščani
- Sreda: Ain, Boa, Doyun, Gaon = 4 vaščani
- Četrtek: Doyun, Eunwoo, Felix = 3 vaščani

- Petek: Ain, Chaewon, Hana = 3 vaščani
- Sobota: Ain, Doyun, Felix, Hana = 4 vaščani
- Nedelja: Eunwoo, Gaon = 2 vaščana

Ker morajo vsak Dure dan sodelovati najmanj štirje vaščani (glede na 1. pravilo), je lahko Dure dan le v ponedeljek, torek, sredo ali soboto. Izmed teh štirih dni moramo izbrati tri.

Chaewon je od navedenih štirih dni na voljo le v torek, Eunwoo pa le v ponedeljek. Ker mora vsak vaščan sodelovati najmanj en dan (2. pravilo), mora biti Dure v ponedeljek in torek.

Izbrati moramo še tretji dan za Dure. To je lahko sreda ali sobota. Ker pa po 3. pravilu noben vaščan ne sme delati vseh treh Dure dni, sreda ne more biti Dure dan. Če bi bila, bi morala Boa delati vse tri Dure dni, saj zaradi 1. pravila (sodelovanje vsaj štirih vaščanov) ne bi mogla izpustiti nobenega od njih. Tako ostane le še sobota.

Za Dure dni lahko torej izberemo le ponedeljek, torek in soboto.

Preverimo še enkrat, ali rešitev ustreza vsem trem pravilom. Pomagamo si s spodnjo tabelo, v kateri je prikazano, koliko dni sodeluje posamezni vaščan in koliko vaščanov sodeluje na posamezni dan.

ime	ponedeljek	torek	sobota	število dni
Ain	O		O	2
Boa	O	O		2
Chaewon		O		1
Doyun			O	1
Eunwoo	O			1
Felix		O	O	2
Gaon	O			1
Hana		O	O	2
št. vaščanov	4	4	4	

Pravilu 1 je zadoščeno, saj vsak Dure dan sodelujejo natanko 4 vaščani (zadnja vrstica tabele). Pravilu 2 je zadoščeno, saj vsak vaščan sodeluje vsaj en dan (zadnji stolpec v tabeli). Poleg tega iz zadnjega stolpca vidimo, da nihče od vaščanov ne sodeluje vse tri dni, tako da je zadoščeno tudi pravilu 3.

Računalniško ozadje

V nalogi smo reševali problem zadovoljevanja omejitev. Pri reševanju tega problema iščemo vrednosti spremenljivk tako, da so hkrati izpolnjene vse podane omejitve (pravila). Problem vključuje spremenljivke, ki jih je treba določiti, množico možnih vrednosti za vsako spremenljivko ter pogoje, ki jih je treba upoštevati.

Takšni problemi se pogosto pojavljajo v vsakdanjih situacijah, kot so izdelava urnikov, načrtovanje razporedov, planiranje, razdeljevanje virov in podobno.

Izhod

8. in 9. razred

Bober Jaka želi sprogramirati svojo prvo spletno igro in se uči, kako preoblikovati sliko. Za začetek dovoli le dve operaciji nad sliko, **Z** in **R**:

Vhod	**Operacija**	Izhod
	**Z** (zrcaljenje čez navpično os)	
	**R** (rotacija za 90 stopinj v smeri urinega kazalca)	

Jaka ima na začetku tole originalno sliko:

Na njej od leve proti desni izvede naslednje zaporedje operacij: **R R R R Z R Z**. Kako bo izgledal izhod po izvedbi tega zaporedja operacij?

A) 

B) 

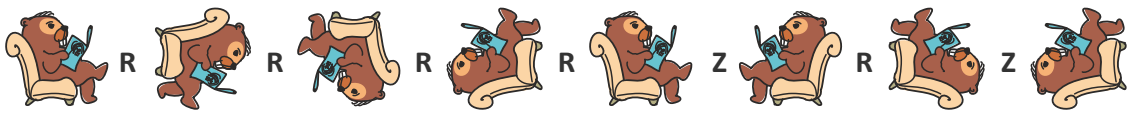
C) 

D) 

Rešitev

Pravilen odgovor je A.

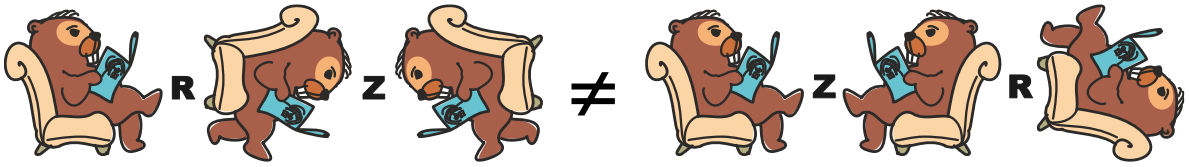
Spodaj je rezultat po posameznih operacijah:



Opazimo, da štiri rotacije (**R**) vrnejo sliko v enako orientacijo kot na začetku, enako pa velja za dve zrcaljenji (**Z**).

Na prvi pogled bi lahko sklepali, da je pet rotacij in dve zrcaljenji enako eni rotaciji, kar prikazuje odgovor B. Zakaj to ni res? Ker na izhod ne vpliva le število rotacij in zrcaljenj, ampak tudi njihovo zaporedje.

Na primer, **R Z** ne da enakega rezultata kot **Z R**:



Le štiri zaporedne rotacije vrnejo sliko v začetno orientacijo. Podobno velja tudi za dve zaporedni zrcaljeni.

Računalniško ozadje

Prostorsko razmišljanje je še posebej pomembno pri algoritmih, ki opisujejo fizične postopke, na primer pri delu z roboti. V računalniškem programiranju, zlasti na področjih, kot sta razvoj iger in animacija, morajo programerji pogosto predvideti, kako se bodo objekti na zaslonu spremenili po zaporedju ukazov (kot so vrtenje, zrcaljenje ali premikanje). Ta naloga preverja sposobnost miselnega izvajanja preprostega programa preoblikovanja slik.

Nepravilno zaporedje

srednja šola

Bober Jaka želi sprogramirati svojo prvo spletno igro in se uči, kako preoblikovati sliko. Za začetek dovoli le dve operaciji nad sliko, **Z** in **R**:

Vhod	**Operacija**	Izhod
	**Z** (zrcaljenje čez navpično os)	
	**R** (rotacija za 90 stopinj v smeri urinega kazalca)	

Jaka ima na začetku tole originalno sliko:

Njegova končna slika izgleda takole:

Katero zaporedje operacij ne preoblikuje originalne slike v končno sliko? (Vse operacije so narejene od leve proti desni.)

A) **Z R**
B) **R R R Z**
C) **R Z**
D) **Z R Z R Z R**

Rešitev

Pravilen odgovor je C.

Poglejmo rezultate po posameznih operacijah podanega zaporedja za vse ponujene odgovore.

Odgovor A:

Odgovor B:

Odgovor C:

Odgovor D:

Računalniško ozadje

Prostorsko razmišljanje je še posebej pomembno pri algoritmih, ki opisujejo fizične postopke, na primer pri delu z roboti. V računalniškem programiranju, zlasti na področjih, kot sta razvoj iger in animacija, morajo programerji pogosto predvideti, kako se bodo objekti na zaslonu spremenili po zaporedju ukazov (kot so vrtenje, zrcaljenje ali premikanje). Ta naloga preverja sposobnost miselnega izvajanja preprostega programa preoblikovanja slik.

Črno-belo kodiranje

8. in 9. razred, srednja šola

Imamo zaporedje kvadratkov, ki so lahko črni ali beli. To zaporedje želimo zakodirati, kodo pa sestavimo na naslednji način:

- Če so vsi kvadratki trenutnega zaporedja **beli**, zapišemo **B**.
- Če so vsi kvadratki trenutnega zaporedja **črni**, zapišemo **Č**.
- Sicer zapišemo **X**, ki mu sledi:
  - koda **leve polovice** trenutnega zaporedja, dobljena po enakih pravilih,
  - koda **desne polovice** trenutnega zaporedja, dobljena po enakih pravilih.

Poglejmo nekaj primerov tako sestavljene kode za zaporedje osmih kvadratkov:

Zaporedje 8 kvadratkov	Koda zaporedja
	**B**
	**XBČ**
	**XXČBČ**
	**XČXBXČB**

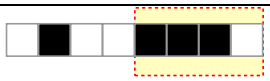
Kakšna je koda spodnjega zaporedja osmih kvadratkov?

Rešitev

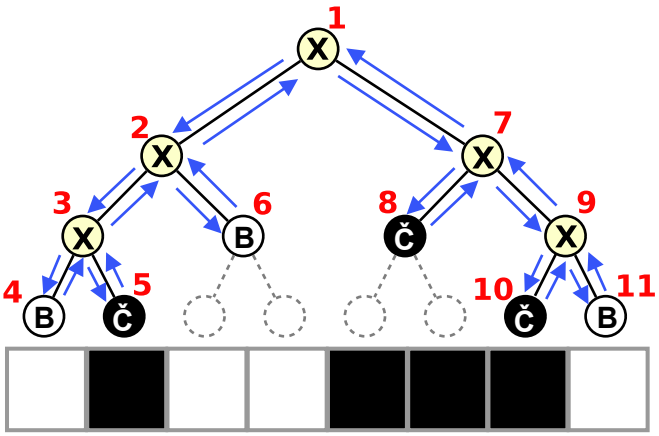
Pravilen odgovor je XXXBČBXČXČB.

Odgovor lahko poiščemo tako, da zaporedje kvadratkov zakodiramo po navedenih pravilih. Kodo sestavljamo znak za znakom:

1.		**X** – ker vseh 8 kvadratkov ni iste barve, zapišemo X.
2.		X**X** – sledi leva polovica, ki tudi nima kvadratkov iste barve, zato zapišemo še en X.
3.		XX**X** – temu sledi leva polovica leve polovice, kjer kvadratka nista iste barve, zato zapišemo še en X.
4.		XXX**B** –sledi leva polovica od leve polovice leve polovice, to je en bel kvadratek, torej zapišemo B.
5.		XXXB**Č** – sledi koda desne polovice od leve polovice leve polovice, to je en črn kvadratek, torej zapišemo Č.

6.		XXXBČ**B** – sledi koda desne polovice leve polovice, ki ima dva bela kvadratka; ker sta oba bela, zapišemo B.
7.		XXXBČB**X** – sledi desna polovica, ki tudi nima vseh kvadratkov iste barve, zato zapišemo X.
8.		XXXBČBX**Č** – sledi leva polovica desne polovice, to sta dva črna kvadratka; ker sta oba črna, zapišemo Č.
9.		XXXBČBXČ**X** – sledi desna polovica desne polovice, to sta zadnja dva kvadratka v zaporedju; ker sta različne barve, zapišemo X.
10.		XXXBČBXČXČ – sledi leva polovica od desne polovice desne polovice, to je en črn kvadratek, zato zapišemo Č.
11.		XXXBČBXČXČ**B** –sledi še koda desne polovice od desne polovice desne polovice, to je en bel kvadratek, zapišemo še B.

Drug način predstavitve sestavljanja kode prikazuje spodnja slika. Sestavimo drevo, v katerem so v vozliščih znaki B, Č ali X. Modre puščice nakazujejo pot po drevesu, rdeče številke pa določajo vrstni red zapisovanja kode.



Kako sestavimo drevo na zgornji sliki? Najprej zapišemo zaporedje s črkami za vsak kvadratek: B Č B B Č Č Č B. Vsaka črka predstavlja eno vozlišče v drevesu, kar tvori najnižji nivo drevesa oziroma liste. Nato gradimo drevo po korakih proti višjim nivojem. Po vrsti od leve proti desni združujemo po dve vozlišči v novo vozlišče višjega nivoja. Če imata oba sinova vozlišča isto črko (oba B ali oba Č), s to črko označimo tudi vozlišče (očeta) in odstranimo oba sinova. Sicer (če imata sinova različni črki) vozlišče (očeta) označimo z X. Združevanje začnemo pri listih in gradimo nove, višje nivoje, dokler ne pridemo do korena drevesa (ko ni več vozlišč za združevanje).

Kodo zaporedja kvadratkov sestavimo tako, da začnemo na najvišjem nivoju drevesa (pri korenu) in se premikamo po drevesu navzdol, pri čemer na vsakem razcepu najprej izberemo levo pot. Ob obisku vsakega novega vozlišča si zabeležimo črko, s katero je vozlišče označeno. Ko pridemo do lista (vozlišča brez sinov), se vrnemo do predhodnega vozlišča in nadaljujemo po desni poti. Ko pregledamo obe poti, se vrnemo na predhodno vozlišče. Ko tako pregledamo celo drevo, nam zabeležene črke po vrsti dajo kodo zaporedja kvadratkov. Prehod preko vozlišč drevesa na zgornji sliki prikazujejo modre puščice, rdeče številke pa določajo vrstni red zapisovanja znakov kode.

Računalniško ozadje

Pravilo, ki smo ga v nalogi uporabili za kodiranje zaporedja kvadratkov, je *rekurzivno*. To pomeni, da smo rešitev problema (kako sestaviti kodo) opisali s pomočjo iste rešitve: v enostavnem primeru, ko je zaporedje kvadratkov iste barve, je koda B ali Č. Sicer je koda znak X, ki mu sledita kodi za levo in desno polovico. Tako smo kompleksen problem razdelili na dva manjša problema (koda leve polovice in koda desne polovice), vsakega pa rešimo na enak način kot prvotni problem.

Rekurzija je torej način reševanja problemov, kjer problem razdelimo na manjše podobne probleme in jih rešujemo na enak način. Vsaka rekurzivna funkcija ima običajno dva dela: osnovni primer, kjer je rešitev enostavna in jo poznamo, ter rekurzivni del, kjer problem razdelimo na manjše probleme in jih rešimo s pomočjo iste funkcije. Pomembno je, da rekurzija vedno doseže osnovni primer, sicer bi se program (vsaj v teoriji) vedno znova ponavljal in se ne bi nikoli končal.

Rekurzijo pogosto uporabljamo v računalništvu, na primer pri izračunu Fibonaccijevih števil, risanju fraktalov ali preiskovanju grafov in dreves.

Izmenjava knjig

8. in 9. razred, srednja šola

Šest bobrov, Ana, Bor, Cvetka, Deni, Eva in Franci, si izmenjuje knjige v šolskem klubu mladih bralcev. Vsak bober ima svoje priljubljene literarne žanre. Za kar najboljšo bralno izkušnjo so postavili naslednja pravila za izmenjavo knjig:

- Vsak bober podari natanko eno knjigo svojega najljubšega žanra drugemu bobru, s katerim deli isti najljubši žanr.
- Vsak bober prejme eno knjigo svojega drugega najljubšega žanra od drugega bobra, s katerim deli isti najljubši žanr.

Priljubljeni žanri bobrov so prikazani s kljukico (✓) v naslednji tabeli:

Bober	Skrivnostni	Pustolovski	Znanstvena fantastika	Fantazijski	Zgodovinski	Biografski
Ana	✓	✓				
Bor			✓	✓		
Cvetka					✓	✓
Deni		✓	✓			
Eva	✓				✓	
Franci				✓		✓

Cvetka je podarila knjigo Evi. Kdo podari knjigo Boru in katerega žanra je ta knjiga?

A) Ana, Fantazijski
B) Franci, Fantazijski
C) Deni, Znanstvena fantastika
D) Cvetka, Znanstvena fantastika
E) Eva, Zgodovinski
F) Cvetka, Zgodovinski

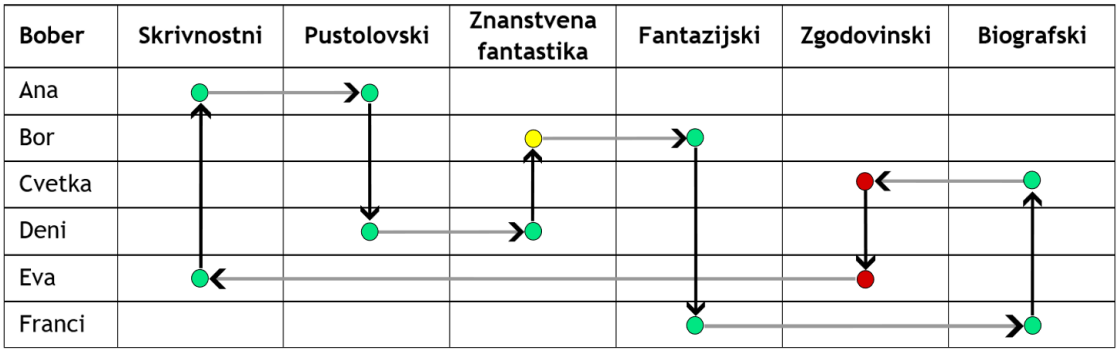
Rešitev

Pravilen odgovor je C. Deni podari Boru knjigo znanstvene fantastike.

Vemo, da je Cvetka podarila knjigo Evi. To pomeni, da je bila knjiga zgodovinskega žanra, saj je to edini žanr, ki ga imata obe radi (Cvetkina najljubša sta zgodovinski in biografski, Evina pa skrivnostni in zgodovinski).

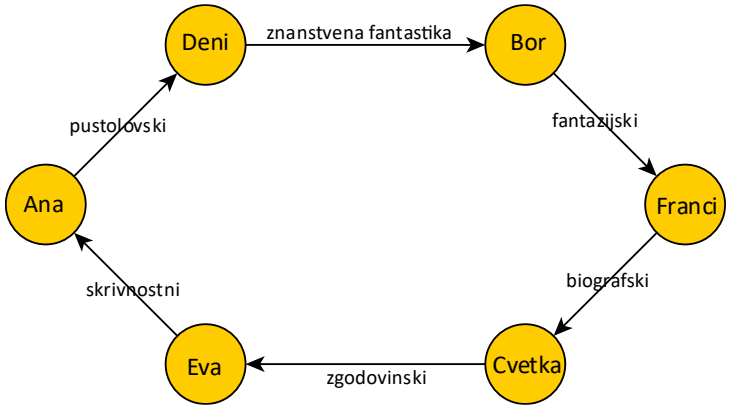
Nato sledimo izmenjavam knjig glede na priljubljene žanre. Če je Eva prejela zgodovinsko knjigo, bo naprej podarila knjigo skrivnostnega žanra, saj sta njena najljubša žanra zgodovinski in skrivnostni. Knjigo skrivnostnega žanra lahko podari le Ani, saj le Ana deli ta žanr z Evo.

Nadaljnje menjave knjig lahko prikažemo v tabeli s puščicami. Menjava, ki jo poznamo (Cvetkina zgodovinska knjiga Evi), je označena z rdečima točkama. Pri menjavi, ki nas zanima, torej od koga dobi knjigo Bor, pa je Bor označen z rumeno točko.



Iz tabele razberemo, da Ana nato podari pustolovsko knjigo Deniju, Deni pa znanstveno fantastiko Boru. Tako smo našli odgovor.

Menjave knjig v zgornji tabeli lahko poenostavljeno zapišemo v obliki grafa:



Iz grafa še lažje vidimo, da je Bor prejel znanstvenofantastično knjigo od Denija.

Računalniško ozadje

Problem lažje razumemo in rešimo, če ga predstavimo kot graf. *Graf* je sestavljen iz vozlišč (točk) in povezav (črt med njimi). V tej nalogi uporabljamo *usmerjen graf*, kar pomeni, da povezave kažejo smer s puščicami.

Graf nam pomaga iz vsega, kar je podano, izbrati le tisto, kar je pomembno za rešitev. Ko se osredotočimo samo na te informacije, je nalogo veliko lažje rešiti. Tak način poenostavljanja problema imenujemo *abstrakcija* in se pogosto uporablja v računalništvu.

Obraba segmenta

8. in 9. razred, srednja šola

Digitalna ura za prikaz časa na minuto natančno uporablja štiri prikazovalnike za številke, vsak prikazovalnik pa sestavlja sedem svetlečih segmentov.

Posamezne številke so s kombinacijo prižganih in ugasnjenih segmentov prikazane takole:

Vsak segment ima določeno življenjsko dobo in zdrži le določeno število aktivacij (tj. sprememb iz ugasnjenega v prižgano stanje). Segment, ki se največkrat aktivira, bo treba najprej zamenjati.

Kateri segment je to?

Rešitev

Pravilen odgovor je:

Vsaka številka na prikazovalniku predstavlja različno časovno obdobje: desetice ur, enice ur, desetice minut in enice minut. Skrajno desna številka (na četrtem prikazovalniku) prikazuje spremembe minut, kar se zgodi najpogosteje. Na tej skrajno desni številki se bodo namreč spremenili vsi segmenti, preden se bo spremenila katera koli druga številka (tj. na prvih treh prikazovalnikih). Torej je segment, ki se največkrat aktivira, na skrajno desnem prikazovalniku.

Poglejmo še vseh sedem segmentov na tem prikazovalniku in poskusimo ugotoviti, kolikokrat se posamezni segment aktivira, preden se zamenjajo vse številke od 0 do 0. Posamezne segmente smo označili s črkami od A do G (na sliki desno).

Slika prikazuje takšno preverjanje. Aktiviran (ravnokar prižgan) segment je tisti, ki sveti na trenutni številki, vendar ne sveti na najbližji številki levo od nje. Vsaka aktivacija segmenta je na zgornji sliki označena s črko tega segmenta. Število aktivacij je enako številu črk na celotni sliki.

Rezultate tega preverjanja povzema spodnja tabela. Srednji stolpec prikazuje številke, pri katerih je bil segment aktiviran (torej prižgan, potem ko je bil v predhodni številki ugasnjen), desni stolpec pa število takih aktivacij.

Segment	Število, ki povzroči aktivacijo	Število aktivacij
A	2, 5	2
B	7	1
C	3	1
D	2, 5, 8	3
E	2, 6, 8, 0	4
F	4, 8	2
G	2, 8	2

Iz tabele lahko razberemo, da je največkrat aktiviran segment E (štirikrat).

Računalniško ozadje

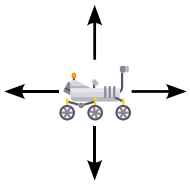
Podobne prikazovalnike, kot je ta v naši nalogi, lahko najdete tudi v različnih informacijskih panelih, na primer v dvigalih ali na peronih železniške postaje. Prikazovalnik s sedmimi segmenti je običajen način digitalnega prikaza v tehnoloških izdelkih. Številke od 0 do 9 prikaže z uporabo sedmih LED-lučk.

Rover na Marsu

8. in 9. razred, srednja šola

Marsovski rover mora doseči svojo bazo in pri tem porabiti čim manj energije. Poleg tega je tam tudi izgubljen vesoljček, ki ga lahko pobere (ni pa obvezno) in ga odpelje nazaj do njegovega letečega krožnika.

	5				3	2	1
5				12	10	8	2
					3	6	4
					4	4	2



Pri tem mora rover upoštevati naslednja pravila:

- Lahko se premika po kvadratih samo v smeri puščic (glej sliko zgoraj desno).
- Številka na kvadratu pove, koliko energije potrebuje, da vstopi na ta kvadrat. Če številke na kvadratu ni, energije ne potrebuje.
- Ne more se premikati čez skale.
- Kvadrat z letečim krožnikom lahko obišče le, če je že pobral vesoljčka.

Koliko najmanj energije potrebuje rover, da doseže bazo?

Rešitev

Da rover doseže svojo bazo, potrebuje najmanj 20 enot energije.

Spodnja slika prikazuje pot, ki jo rover opravi do baze. Ta pot zahteva natanko 20 enot energije.

	5		*		3	2	1
5				12	10	8	2
					3	6	4
					4	4	2

Kako preverimo, da baze ne more doseči z manj energije? Zaradi postavljenih skal mora pot roverja do baze voditi preko kvadrata, ki je na zgornji sliki označen z zvezdico. Do tega kvadrata vodita le dve poti: rover lahko na poti do kvadrata z zvezdico pobere vesoljčka, za kar porabi 10 enot energije, ali pa gre mimo vesoljčka, za kar ne porabi nobene energije.

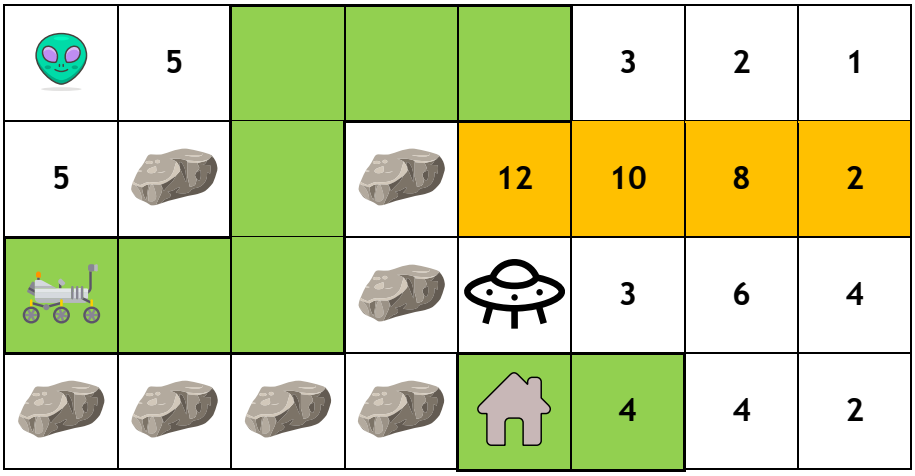
Če rover pobere vesoljčka (in za to porabi 10 enot energije), bi moral od kvadrata z zvezdico do baze porabiti največ 9 enot energije, da bi za celotno pot skupaj porabil manj kot 20 enot energije (kolikor jih potrebuje pri zgornji rešitvi). To pomeni, da na poti do baze ne sme preko kvadratov, označenih z 12 ali 10. Hitro vidimo, da bi tak obvoz terjal še več energije (dodatnih 5 enot, da pride do kvadrata z oznako 8, oziroma dodatnih 6 enot do kvadrata z oznako 2 in naprej proti bazi še najmanj 4 enote). Torej za pot, na kateri pobere tudi vesoljčka, potrebuje več kot 20 enot energije.

Druga možnost je, da vesoljčka ne pobere. V tem primeru ne more preko kvadrata z letečim krožnikom, torej se mora premikati preko kvadratov, ki so na spodnji sliki označeni zeleno.

	5				3	2	1
5				12	10	8	2
					3	6	4
					4	4	2

Da bi izboljšali predlagano rešitev (z 20 enotami energije), moramo povezati obe poti na zgornji sliki tako, da bo poraba energije na tej povezavi največ 19 – 4 = 15 enot.

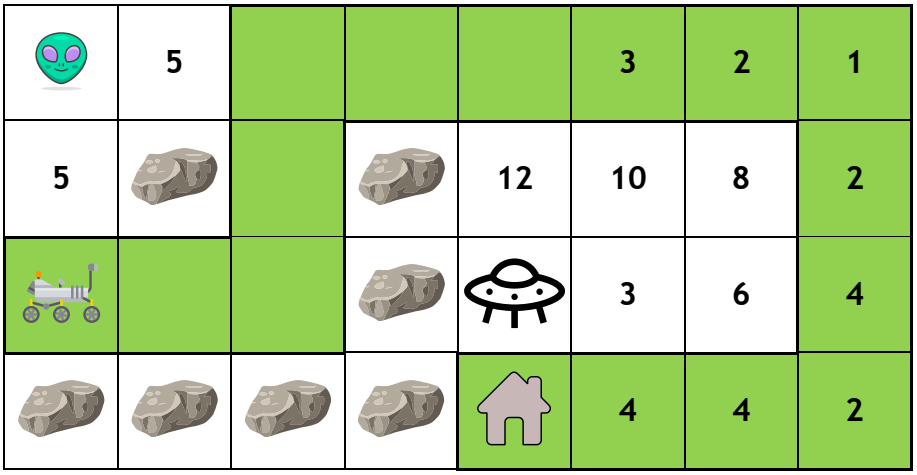
Povezava obeh delov poti mora potekati preko vsaj enega od kvadratov, ki so na spodnji sliki označeni oranžno.



Očitno povezava ne more voditi preko kvadrata z oznako 12, saj bi v tem primeru moral nadaljevati na kvadrat z oznako 10 (ker brez vesoljčka ne sme na kvadrat z letečim krožnikom), za kar že porabi preveč energije. Tudi povezava preko kvadrata z oznako 10 ni ustrezna, saj potrebuje 3 enote energije le za to, da pride do kvadrata 10, in nato še najmanj 3 enote za nadaljevanje poti, kar je skupaj več kot 15 enot (3 + 10 + 3 = 16).

Podobno lahko sklepamo za kvadrat z oznako 8: vsaj 3 + 2 = 5 enot energije potrebuje do tega kvadrata, tako da mu ostaneta le 15 – 5 – 8 = 2 enoti energije, da z njega pride do zelenega dela poti pri bazi. To očitno ni mogoče.

Preostane le še možnost, da pot vodi naokoli preko kvadrata z oznako 2, kot prikazuje spodnja slika.



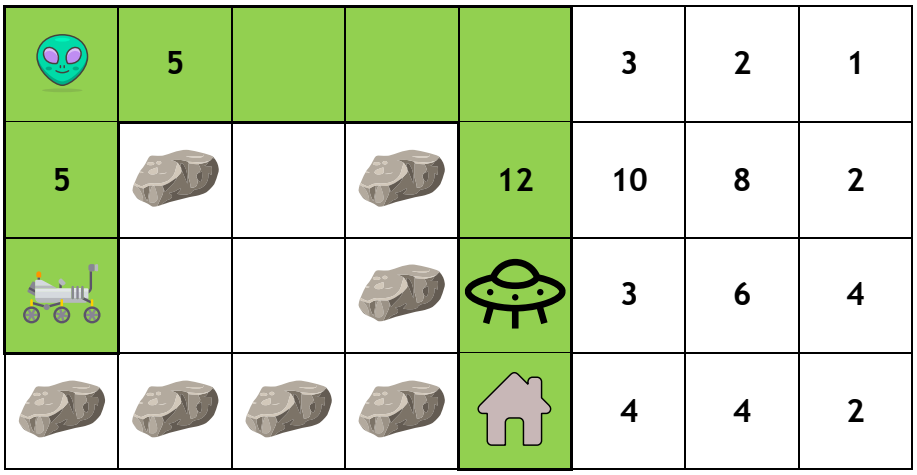
Vendar je potrebna energija za to pot 3 + 2 + 1 + 2 + 4 + 2 + 4 + 4 = 22 enot, kar je več od prve predlagane rešitve.

Računalniško ozadje

V nalogi smo prikazali problem, kjer mora rover poiskati rešitev, ki porabi najmanj energije. Podobni problemi, kjer je treba minimizirati vire (npr. energija, čas, cena, dolžina) ali maksimizirati rezultate (npr. učinkovitost, profit), so v računalništvu zelo pogosti. Imenujejo se *optimizacijski problemi*.

Rešujemo jih lahko na različne načine, z uporabo različnih pristopov. Računalniški strokovnjaki se veliko ukvarjajo z različnimi strategijami iskanja rešitve in so razvili tudi več algoritmov, ki vodijo do rešitve. Nekatere metode vodijo do najboljše možne rešitve, medtem ko se druge raje osredotočijo na učinkovitost in vodijo do rezultata, ki ni nujno najbolj optimalen, je pa dovolj blizu, a je zato rešitev lažje in hitreje izračunati. To je zelo pomembno, ker so realni problemi lahko zelo kompleksni, rešitve pa potrebujemo hitro.

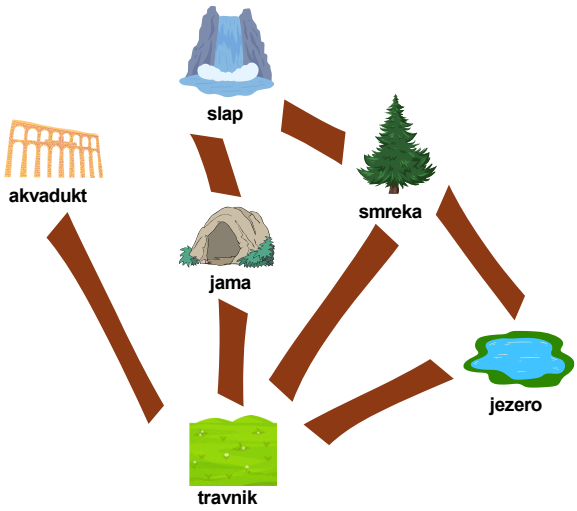
Primer take strategije je, da izberemo pot, ki gre preko najmanj kvadratov. Taka pot vodi preko kvadrata z letečim krožnikom, zato moramo predhodno pobrati vesoljčka. Pot je dolga 10 kvadratov in porabi 22 enot energije. Poraba energije je nekoliko večja od najboljše rešitve, a še vedno ni slaba.



Drug pristop je, da se na vsakem kvadratu odločimo za pot, ki zahteva najmanj energije. Rezultat take poti je prikazan na zadnji sliki v razlagi rešitve. Tudi tu nismo dobili najboljšega rezultata, a je 22 enot energije »dovolj dober« rezultat. Uporabljen algoritem se imenuje *požrešni algoritem* in pogosto pripelje do dobrih rezultatov, čeprav ne zagotavlja najboljše rešitve.

Če pa je zelo pomembno, da je rešitev resnično najboljša, se za iskanje rešitve uporabi *Dijkstrov algoritem*. To je algoritem za iskanje najkrajših poti od izhodiščnega vozlišča do vseh drugih vozlišč v uteženem grafu z nenegativnimi utežmi.

Gospod Bober je obiskal narodni park, v katerem je več znamenitosti: akvadukt, smreka, jezero, jama, slap in travnik. Vse znamenitosti so povezane s potmi, kot prikazuje slika.



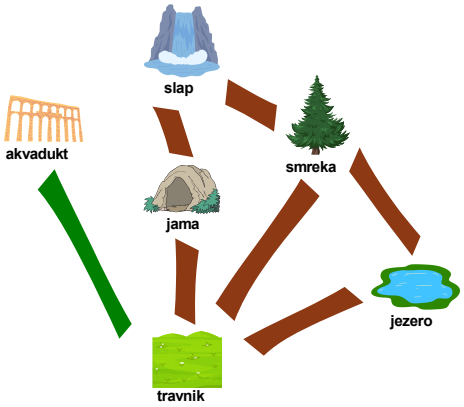
Gospod Bober lahko začne (in konča) ogled pri kateri koli znamenitosti, a želi videti vse znamenitosti in vsako obiskati le enkrat. To lahko naredi na več načinov. Če obišče znamenitosti v različnem vrstnem redu, se to šteje kot drug način obiska.

Na koliko načinov lahko gospod Bober obišče vse znamenitosti?

Rešitev

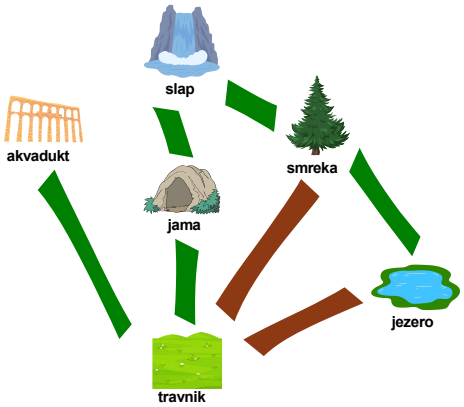
Vse znamenitosti lahko obišče na 4 različne načine.

Akvadukt je edina znamenitost, do katere vodi le ena pot in je povezana le z eno drugo znamenitostjo. To pomeni, da mora akvadukt obiskati kot prvo ali kot zadnjo znamenitost.

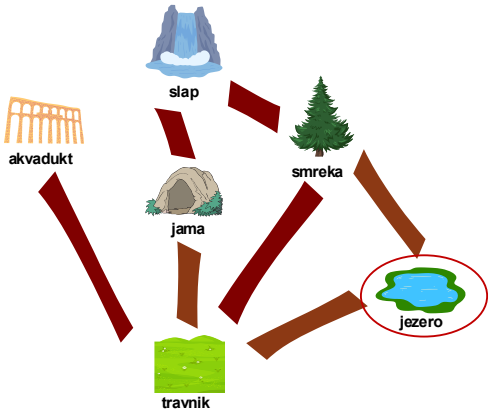


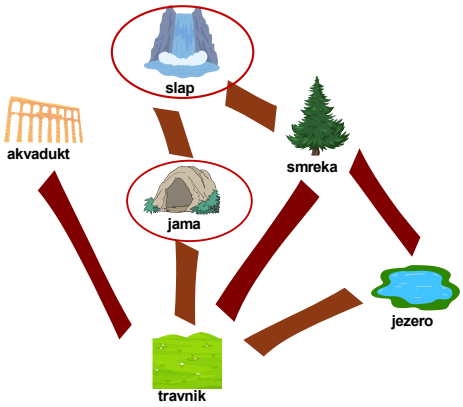
Če začne ogled z akvaduktom, lahko nadaljuje le na travnik, saj tja vodi edina pot. S travnika lahko obišče jamo, smreko ali jezero.

Če se odloči za jamo, ga pot nato vodi do slapa, nato do smreke, od tam pa lahko pride še do jezera in si tako ogleda vse znamenitosti. Njegovo pot prikazuje spodnja slika (pot je označena z zeleno). To je en način obiska vseh znamenitosti.

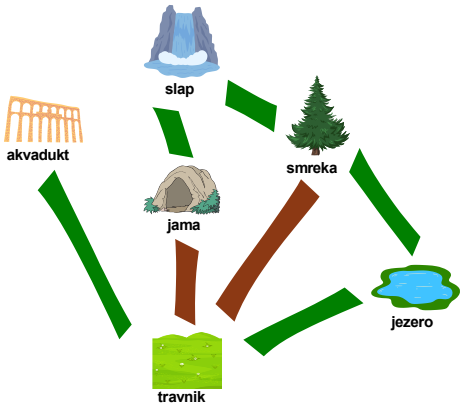


Če se po travniku odloči obiskati smreko, ne bo mogel obiskati vseh znamenitosti, ne da bi katero obiskal dvakrat. Moral bi namreč izpustiti bodisi jezero bodisi slap in jamo. Alternativi sta prikazani na spodnjih slikah.





Če pa se po travniku odloči za obisk jezera, lahko nadaljuje pot do smreke in nato do slapa in jame. Tudi v tem primeru si lahko ogleda vse znamenitosti, kar je drugi način obiska vseh znamenitosti – prikazan je na spodnji sliki z zeleno potjo.



Če začne pri akvaduktu, lahko vse znamenitosti obišče na dva načina. Ker je lahko akvadukt tudi zadnja obiskana znamenitost, lahko obe poti tudi obrnemo. Skupaj imamo štiri možne načine obiska vseh znamenitosti, če vsako obiščemo le enkrat.

Računalniško ozadje

Zemljevid znamenitosti narodnega parka lahko obravnavamo kot *graf*. Graf je sestavljen iz vozlišč in povezav med njimi. V tej nalogi so vozlišča znamenitosti narodnega parka, povezave pa poti med njimi.

Pot v grafu je zaporedje povezanih vozlišč, kjer se vozlišča in povezave ne ponovijo. Na primer: smreka, jezero in travnik tvorijo pot (v tem vrstnem redu), ker obstaja povezava med smreko in jezerom ter povezava med jezerom in travnikom. Način, kako lahko gospod Bober obišče vse znamenitosti, lahko opišemo kot *Hamiltonovo pot*. To je pot, ki gre skozi vsako vozlišče grafa natanko enkrat. Če sta začetno in končno vozlišče poti enaki, se imenuje *Hamiltonov cikel*. Graf, ki vsebuje Hamiltonov cikel, se imenuje *Hamiltonov graf*.

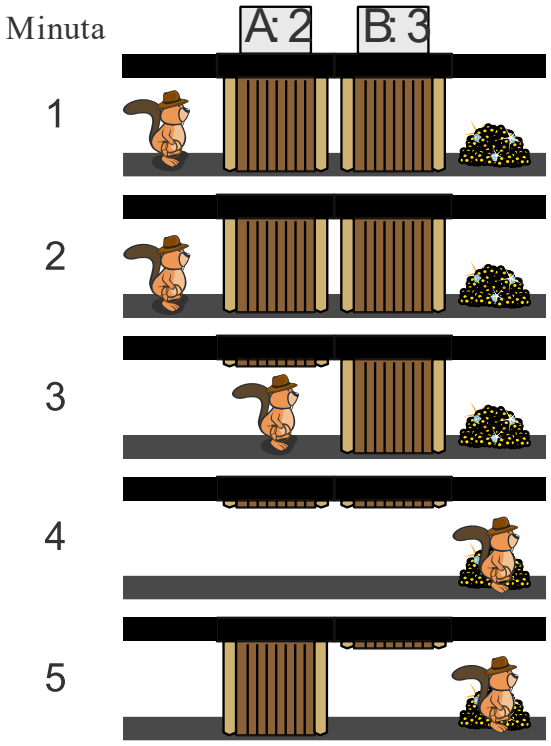
Zelo znan problem, povezan s Hamiltonovimi grafi, je problem trgovskega potnika, kjer mora trgovski potnik obiskati vsa mesta, pri tem pa vsako mesto lahko obišče le enkrat.

Bober Janez

srednja šola

Bober Janez raziskuje nevarno piramido, ki ima veliko grozljivih hodnikov. Na koncu vsakega hodnika je izjemen zaklad, ki ga Janez želi čim prej doseči.

Vsak hodnik je zavarovan z vrsto blokov. Na začetku so vsi bloki spuščeni. Takoj ko nekdo prispe do hodnika, se njegovi bloki začnejo periodično premikati. Blok s periodo 2 se po 2 minutah dvigne, po nadaljnjih 2 minutah pa se ponovno spusti navzdol in tako naprej.



Oglejmo si primer na levi. Ta hodnik ima dva bloka, označena z A in B, prvi s periodo 2 in drugi s 3. Slika prikazuje stanje hodnika v minutah od 1 do 5 po tem, ko je Janez prispel.

Da bi čim prej dosegel zaklad, Janez počaka 2 minuti, gre do bloka A, počaka še 1 minuto in nato pride mimo bloka B ter po skupno 3 minutah doseže zaklad.

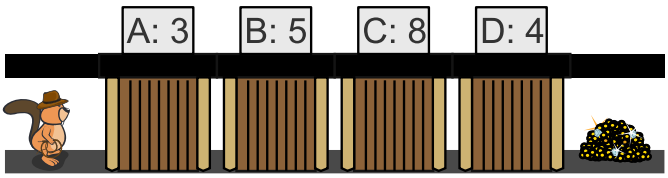
Janez v resnici sledi naslednjemu zaporedju štirih ukazov:

```
pocakaj(2)
pojdi_do_blok(A)
pocakaj(1)
pojdi_do_zaklada
```

Janez bi lahko zaklad dosegel ob istem času, vendar s krajšim zaporedjem dveh ukazov:

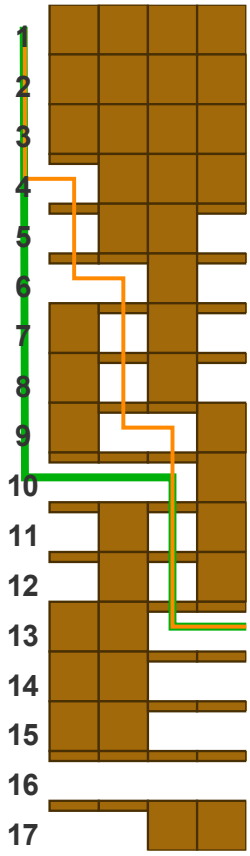
```
pocakaj(3)
pojdi_do_zaklada
```

Naslednji grozljivi hodnik ima štiri bloke s periodami 3, 5, 8 in 4. Med vsemi zaporedji ukazov, ki Janezu omogočajo najhitrejši prihod do zaklada, koliko ukazov ima najkrajše zaporedje?



Rešitev

Najkrajše zaporedje ima 4 ukaze.



Oglejmo si sliko na levi, ki prikazuje, kako se lahko Janez premika skozi ta grozljivi hodnik. Da bi se premikal čim hitreje, lahko sledi naslednjemu zaporedju ukazu in doseže zaklad po 12 minutah (oranžna črta):

```
pocakaj(3)
pojdi_do_blok(A)
pocakaj(2)
pojdi_do_blok(B)
pocakaj(3)
pojdi_do_blok(C)
pocakaj(4)
pojdi_do_zaklada
```

Vendar lahko sledi tudi manj ukazom in prav tako doseže zaklad po 12 minutah (zelena črta):

```
pocakaj(9)
pojdi_do_blok(C)
pocakaj(3)
pojdi_do_zaklada
```

Ni mogoče, da bi Janez sledil manj ukazom, ne da bi pri tem izgubil čas. Manj ukazov bi lahko uporabil samo, če bi pretekel hodnik v enem zamahu. Edina možnost za to nastopi po čakanju 15 minut. Takrat lahko zaklad doseže z le dvema ukazoma – vendar po 15 minutah, kar ni najhitreje.

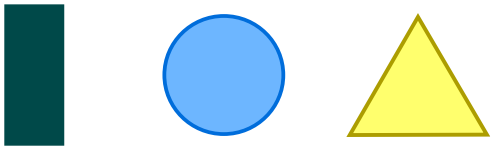
Računalniško ozadje

V tej nalogi moramo izbrati najboljši način, da Janez čim prej doseže zaklad, hkrati pa uporabimo čim manj ukazov. Takšne naloge v računalništvu imenujemo optimizacijski problemi, saj želimo minimizirati določen dejavnik (čas in število ukazov). Eden od načinov reševanja optimizacijskih problemov je dinamično programiranje. Ta pristop vključuje razdelitev problema na manjše dele in postopno reševanje večjih delov prvotnega problema z uporabo shranjenih delnih rešitev. V tej nalogi bi delni problemi lahko zajemali dosego prvega, drugega, tretjega ali četrtega bloka, preden se določi celotna pot do zaklada.

Bobrov jezik

srednja šola

V bobrovem jeziku so imena teh treh oblik:



mer ogg epp

Vsi načini, kako je lahko druga oblika za ali pod prvo obliko, so določeni na prvih štirih spodnjih slikah z uporabo končnic »x« in »z«. Tretjo obliko lahko glede na drugo obliko razporedimo po isti definiciji, kot je prikazano na zadnjih štirih slikah.

ogg merx

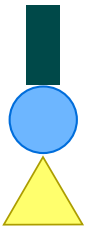
mer oggx

ogg merz

mer oggz

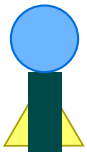
epp oggx merz

mer oggz eppz



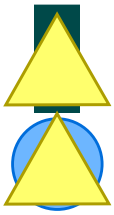
epp merx oggx

ogg merz eppx

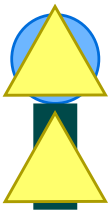


Katero od naslednjih razporeditev oblik lahko opišemo z epp eppz oggx merx?

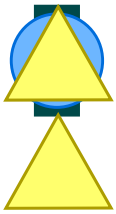
A)



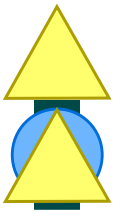
B)



C)



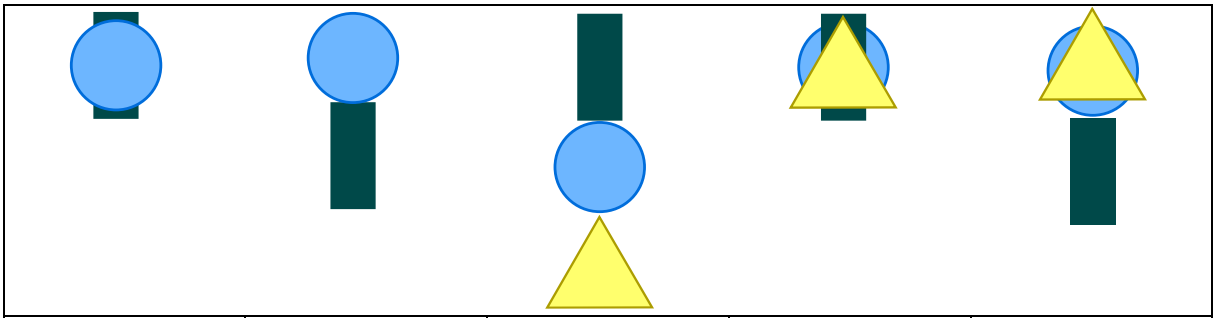
D)



Rešitev

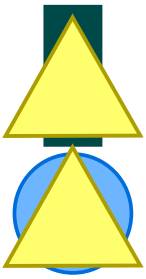
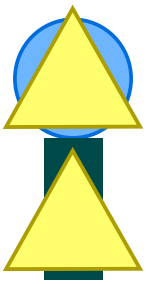
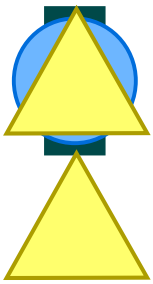
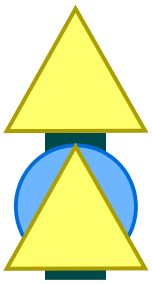
Pravilen odgovor je D.

Iz slik lahko razberemo, da so različne razporeditve oblik določene s končnicama x in z. Prav tako vemo, da lahko tretjo sliko razporedimo glede na drugo sliko z uporabo istih končnic:



ogg merx → končnica -x pomeni »za« »pravokotnik je za krogom«	ogg merz → končnica -z pomeni »pod« »pravokotnik je pod krogom«	mer oggz eppz → »krog je pod pravokotnikom« → »trikotnik je pod krogom«	epp merx oggx → »pravokotnik je za trikotnikom« → »krog je za pravokotnikom«	epp oggx merz → »krog je za trikotnikom« → »pravokotnik je pod krogom«

Torej lahko posamezne slike pri podanih odgovorih opišemo takole:

A)	B)	C)	D)
			
epp merx eppz oggx ni pravilno	epp oggx eppz merx ni pravilno	epp oggx merx eppz ni pravilno	epp eppz oggx merx pravilno

Računalniško ozadje

V računalništvu je formalni jezik podoben jeziku, ki ga posebej oblikujemo tako, da računalniki natančno razumejo, kaj mislimo, brez kakršne koli dvoumnosti. Sintaksa je kot slovnica formalnega jezika. Jasno nam določa, katere simbole lahko uporabljamo (npr. črke ali številke) in kako morajo biti ti simboli pravilno združeni, da tvorijo smiselne stavke ali ukaze, ki jih računalnik razume.

V tej nalogi je bobrov jezik zelo preprost formalni jezik. Uporablja besede mer (za pravokotnik), ogg (za krog) in epp (za trikotnik). Njegova sintaksa vključuje pravilo, da dodajanje končnic x ali z tem imenom oblik jasno pokaže, kako naj bodo oblike razporejene po določenih pravilih – na primer, če je ena za drugo ali pod drugo.

Čarobni gumb

srednja šola

Artur ima računalniško miško, kot je še ni imel nihče: ima čarobni gumb! Miška vodi števec m, ki se ob vsakem pritisku čarobnega gumba poveča za 1. A to ni edino, kar se zgodi ob pritisku: števila od m navzdol do 1 se prav tako zapišejo v posebno datoteko. Prvih 5 pritiskov na gumb ustvari naslednji vnos v datoteki:

1 2 1 3 2 1 4 3 2 1 5 4 3 2 1

Število pritiskov (**m**)	Vsebina datoteke
1	**1**
2	1 **2 1**
3	1 2 1 **3 2 1**
4	1 2 1 3 2 1 **4 3 2 1**
5	1 2 1 3 2 1 4 3 2 1 **5 4 3 2 1**

Artur je navdušen nad tem pojavom in je gumb pritisnil prevečkrat, zato ima zdaj v datoteki kup številk. Katera je 127. številka v njegovi datoteki?

Rešitev

Na 127. mestu je zapisana številka 10.

Vidimo, da se velikost datoteke (glede na število zapisanih števil) po vsakem pritisku povečuje takole: 1, 3, 6, 10, 15, 21, 28 … V splošnem, z *i*-tim pritiskom dodamo *i* novih števil. Če je *n* število pritiskov, je števil v datoteki

$$T(n) = 1 + 2 + 3 + \ldots + n = \frac{n\,(n+1)}{2}.$$

Do odgovora lahko pridemo s simulacijo. Zastavimo si vprašanje: katero je najmanjše število *n*, za katerega velja *T(n)* ≥ 127? Jasno je, da nobeno število *n*, pri katerem je *T(n)* manjše od 127, ne more biti rešitev, saj bi to pomenilo, da je v datoteko zapisanih manj kot 127 števil. Zato postopoma povečujemo *n*, dokler ne najdemo prvega ustreznega števila. Ugotovimo, da je *T(*15*)* = 120, kar je premalo, medtem ko je *T(*16*)* = 136.

Ko gumb pritisnemo šestnajstič, se v datoteko zapišejo števila

16 15 14 13 12 11 **10** 9 8 7 6 5 4 3 2 1.

Vemo, da je število 16 zapisano kot 121. število (ker je *T(*15*)* enako 120), zato lahko preprosto preštejemo števila od 16 navzdol, dokler ne pridemo do 127. števila, ki je 10.

Računalniško ozadje

Števila v zgornjem zaporedju imenujemo trikotniška števila in so zelo pogosta v računalništvu. Pojavljajo se v preprostih primerih, kot je štetje števila operacij v gnezdenih zankah, pri

razumevanju časovne zahtevnosti nekaterih algoritmov za urejanje ter tudi v zahtevnejših primerih, povezanih s podatkovnimi strukturami in analizo algoritmov.

Javni prevoz

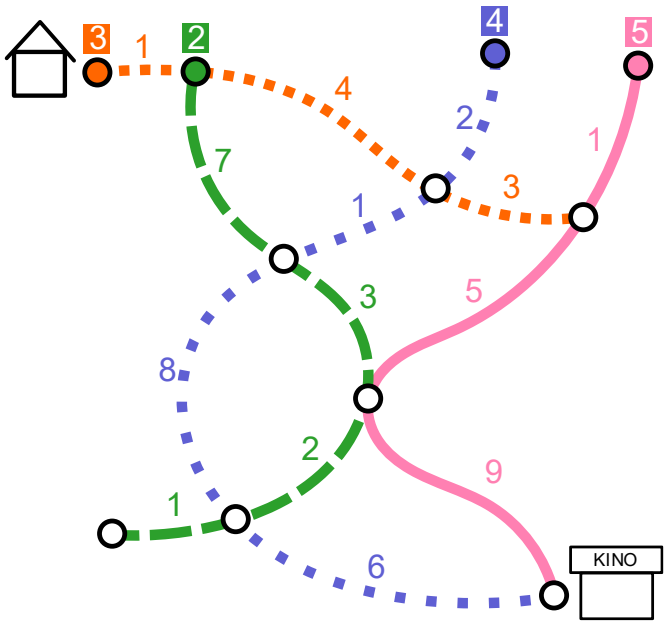
srednja šola

Marko želi priti od doma v kino z avtobusom. V mestu delujejo 4 enosmerne avtobusne linije. Avtobusne postaje so označene s krogi s črnim robom. Avtobusne linije so označene z različno obarvanimi črtami. Obarvane postaje predstavljajo začetno postajo posamezne linije.

Prvi avtobusi na vsaki liniji odhajajo z začetnih postaj ob istem času. Nato vsaka linija pošilja avtobus v različnih časovnih intervalih. Številke v obarvanem ozadju predstavljajo časovni interval med odhodi avtobusov v minutah. Na primer, oranžna linija pošlje avtobus vsake 3 minute – ob minutah 0, 3, 6, 9 in tako naprej.

Številke ob delih linij prikazujejo, koliko minut avtobus potrebuje, da prevozi razdaljo med dvema postajama. Postanek na postaji in vkrcavanje potnikov ne vzame časa (0 minut).

Postaje, kjer se križajo dve ali več linij, je mogoče uporabiti za prestop avtobusa. Če Marko prispe na križišče, lahko prestopi na avtobus, ki prispe tja kasneje ali ob istem času kot on.



Marko odide s prvim oranžnim avtobusom. Najmanj koliko minut potrebuje do kina?

Rešitev

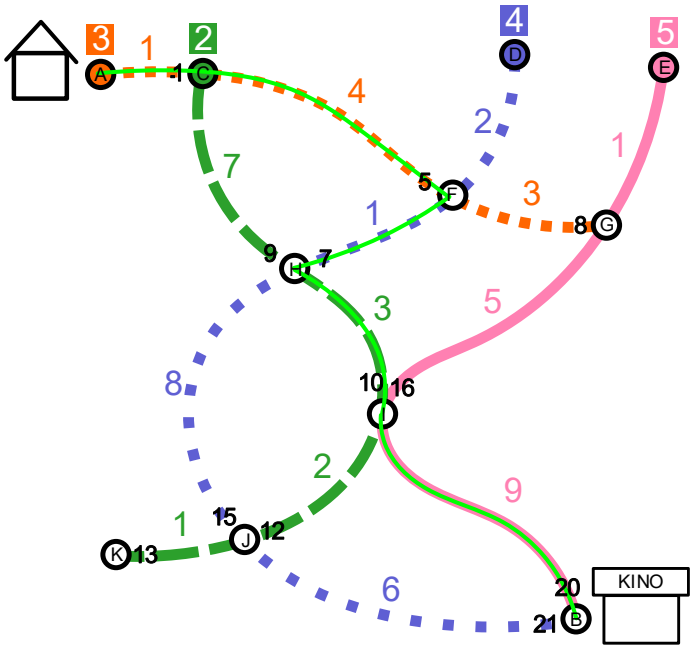
Marko potrebuje do kina najmanj 20 minut.

Marko se mora najprej peljati z oranžnim avtobusom 2 postaji (5 minut), nato z modrim avtobusom 1 postajo (1 minuta čakanja + 1 minuta vožnje), z zelenim avtobusom 1 postajo (3 minute) in z roza avtobusom 1 postajo (1 minuta čakanja + 9 minut).

Odgovor lahko najdemo tako, da izberemo postajo, za katero že poznamo najhitrejši čas do tja (na začetku je to le postaja A). Nato izračunamo čase do naslednjih postaj, dosežene od tam. Ta postopek ponavljamo z različnimi postajami, dokler ne najdemo najhitrejšega časa do cilja.

Vsaka postaja ima lahko več možnih poti z različnimi časi prihoda, vendar se upošteva le najhitrejša. Šele ko so preizkusi vseh možnosti prihoda zaključeni in je najhitrejši čas določen, postajo uporabimo za izračun časov postaj, doseženih iz nje.

Uporaba te metode je opisana spodaj.



Pravilna pot je označena. Številke ob vsaki postaji prikazujejo najkrajši čas, potreben za prihod do postaje iz vsake smeri.

Postaja C je dosegljiva v 1 minuti, če vzamemo prvi oranžni avtobus. Na enak način je postaja F dosegljiva v 5 minutah, postaja G pa v 8 minutah. Postaja I je dosegljiva s postaje G z uporabo tretjega roza avtobusa (3 minute čakanja), skupaj 8 + 3 + 5 = 16 minut. Postaja H je dosegljiva s postaje C (po prihodu ob 1. minuti) z drugim zelenim avtobusom po 1 minuti čakanja in 7 minutah vožnje, skupaj 1 + 1 + 7 = 9 minut od začetka.

Postaja H je dosegljiva tudi s postaje F (po prihodu ob 5. minuti) z drugim modrim avtobusom v 1 minuti po 1 minuti čakanja, skupaj 5 + 1 + 1 = 7 minut. To je hitreje kot prej izračunanih 9 minut, zato se ta novi čas uporabi za nadaljnje izračune. Hitrejši prihod pomeni tudi, da lahko ujamemo prvi zeleni avtobus in z njim pridemo do postaje I v skupnem času 7 + 3 = 10 minut. Novi čas do postaje I je hitrejši od prej izračunanih 16 minut, zato se 16 minut zavrže in ne uporablja naprej. Ker je to najhitrejša pot do postaje I, se isti avtobus lahko uporabi za prihod do postaje J v 10 + 2 = 12 minutah in do postaje K v 12 + 1 = 13 minutah. Postaja J je sicer dosegljiva tudi s postaje H v 7 + 8 = 15 minutah, kar je počasneje in se ne uporablja naprej.

S postaje I lahko z drugim roza avtobusom pridemo do postaje B (cilj) po 1 minuti čakanja, skupaj 10 + 1 + 9 = 20 minut. To še ni končni odgovor, ker je postaja B dosegljiva tudi s postaje J. To je mogoče z drugim modrim avtobusom po 3 minutah čakanja in 6 minutah vožnje, skupaj 12 + 3 + 6 = 21 minut. To je počasneje kot 20 minut, zato je 20 minut, izračunanih prej, dejansko pravilna rešitev.

Računalniško ozadje

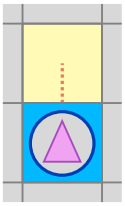
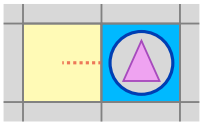
Zemljevid avtobusnih linij sestavlja več postaj, povezanih z enosmernimi povezavami, pri katerih vsaka traja določen čas. Takšna postavitev je podobna uteženemu usmerjenemu grafu. Postaje so vozlišča, linije med njimi pa povezave. Strukturo, ki jo sestavljajo vozlišča in povezave med njimi, imenujemo graf. Ker so povezave enosmerne, so usmerjene, kar pomeni, da vodi povezava le v eno smer. Vsaka povezava ima različen čas prevoza (utež povezave), zato je graf utežen. Edina razlika je, da čas med dvema postajama ni odvisen le od razdalje, temveč tudi od časa prihoda in intervalov avtobusov; v uteženem grafu pa so uteži povezav konstantne in neodvisne od časa prihoda.

Cilj problema je najti najkrajšo pot med dvema vozliščema grafa. Takšni problemi se pogosto pojavljajo v različnih računalniških nalogah in predstavljajo poenostavljeno različico resničnih algoritmov za iskanje poti. Obstaja več načinov reševanja na takem grafu, metoda, opisana v rešitvi, pa je zelo podobna dinamičnemu programiranju, kjer problem rešujemo tako, da najprej rešimo manjše dele in nato te rešitve uporabimo za večje dele, dokler ne pridemo do celotne rešitve. V tem primeru začnemo z izračunom časa do najbližjih postaj od izhodišča in postopoma gradimo do najoddaljenejše postaje – cilja.

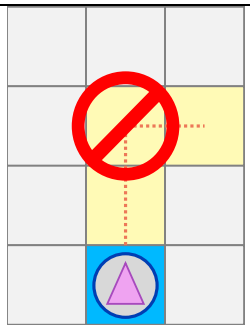
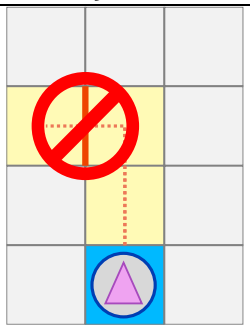
Levi

srednja šola

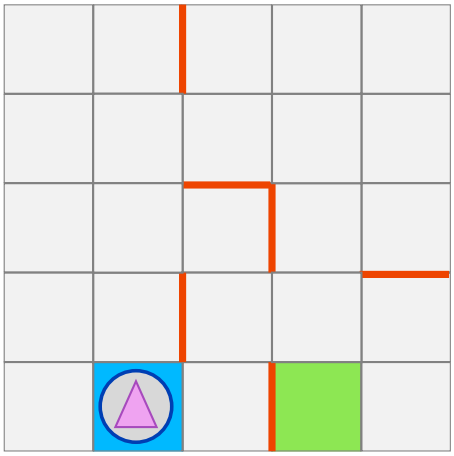
Robot Levi se premika po mreži. Na voljo sta mu le dve vrsti premika. Vsak način premikanja šteje kot en ukaz.

Pojdi naprej za eno polje.	Obrni se levo in pojdi naprej za eno polje.
	

Spodaj sta primera, kako se Levi ne more premikati:

Premik v desno ni možen.	Če je na poti stena, je Levi ne more prečkati.
	

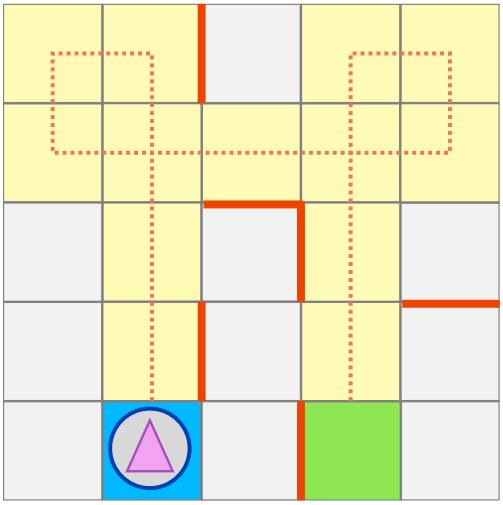
Spodaj je mreža z nekaj stenami. Koliko je najmanjše število ukazov, ki jih mora Levi narediti, da doseže zeleni cilj?



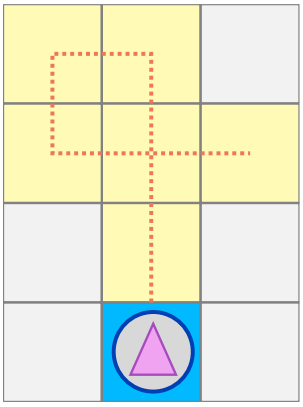
Rešitev

Levi mora narediti najmanj 16 ukazov, da doseže cilj.

Spodnja slika prikazuje, kako Levi doseže cilj, če se premika samo levo, in pri tem obišče najmanjše število polj.



Levi dvakrat uporabi postopek za spremembo smeri v desno, kot je prikazano na spodnji sliki.



Položaj sten ne omogoča nobene druge poti do cilja z manj ali enakim številom obiskanih polj mreže.

Računalniško ozadje

Levijeve možnosti premikanja so močno omejene. Če bi se robot lahko obrnil desno in morda celo preplezal stene, bi bilo njegovo gibanje po mreži veliko lažje. Za to bi potreboval bolj kompleksen nabor ukazov.

Ali je to res potrebno? Na primer, Levi se lahko obrne desno s tremi zaporednimi obrati levo. Pomembno je le, da odstranimo pravilo, da mora Levi po obratu vedno iti naprej – tako bi se lahko premikal v vse smeri. Namesto plezanja čez steno bi lahko preprosto šel okoli nje (če je

to mogoče). To pomeni, da je poenostavljen nabor ukazov prav tako dovolj. Za izvedbo bolj kompleksnega vedenja, ki se pojavlja redkeje, lahko programerji ustvarijo podprogram, ki združi osnovne ukaze v en bolj kompleksen ukaz.

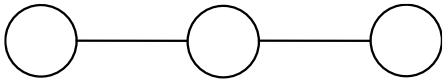
V informatiki sta prav ti dve metodi oblikovanja nabora ukazov procesorjev najbolj priljubljeni: nekateri procesorji so CISC (*Complex Instruction Set Computer* – procesor z bogatim naborom ukazov), drugi pa RISC (*Reduced Instruction Set Computer* – procesor z zmanjšanim naborom ukazov), kot je Levi v tej nalogi. Procesor CISC običajno vsebuje veliko različnih, zelo zmogljivih ukazov (npr. plezanje čez steno), medtem ko RISC vsebuje le tiste ukaze, ki so res pogosto uporabljeni, z možnostjo uporabe podprogramov za redkejše operacije. Obe vrsti arhitektur imata svoje prednosti in slabosti.

Razporejanje izpitov

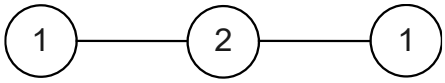
srednja šola

Maša mora več šolskih izpitov razporediti v pet dni. Nekateri dijaki opravljajo več izpitov, zato ti izpiti ne smejo biti razporejeni na isti dan. To je prikazano z diagramom, v katerem vsak krog predstavlja en izpit, črta med dvema krogoma pa pomeni, da imata izpita skupne dijake in morata potekati na različna dneva. Naloga Maše je, da vse izpite razporedi v čim manjše število dni.

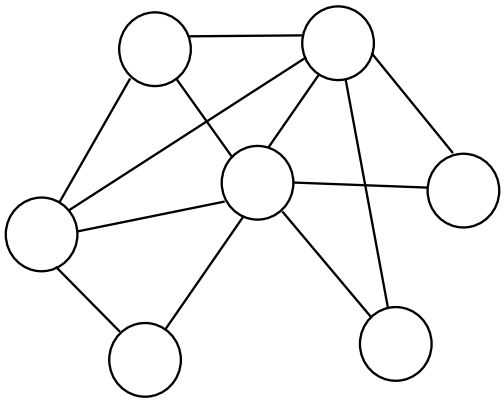
Na primer, spodnji diagram prikazuje tri izpite, kjer ima en izpit skupne dijake z drugima dvema izpitoma, kar je prikazano z dvema črtama:



Te tri izpite lahko Maša razporedi tako, da se levi in desni izpit izvedeta v prvem dnevu, izpit v sredini pa v drugem dnevu:



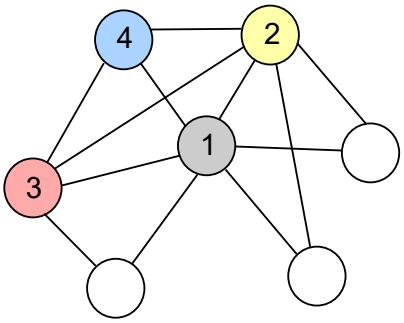
Koliko je najmanjše število dni, ki jih Maša potrebuje, da razporedi sedem izpitov na spodnjem diagramu?



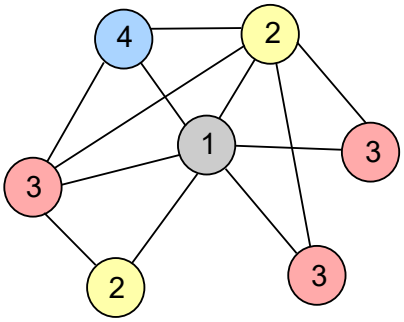
Rešitev

Maša potrebuje najmanj 4 dni, da razporedi 7 izpitov na podani diagram.

Opazimo, da v diagramu obstajajo štirje krogi, ki so med seboj vsi povezani. Zato mora biti vsak od teh štirih izpitov razporejen na svoj dan, torej na štiri različne dni:



Preostale tri izpite enostavno razporedimo na najzgodnejše razpoložljive dni:



Računalniško ozadje

V tej nalogi uporabljamo diagram, ki predstavlja graf, osnovno podatkovno strukturo v računalništvu. Graf sestavljajo vozlišča (krogi) in povezave (črte). Tak model prikazuje situacije, kjer so elementi povezani s konfliktnimi odnosi ali omejitvami.

Naloga je povezana s problemom barvanja grafa: vsakemu vozlišču je treba dodeliti oznako (»barvo«), tako da nobeni dve povezani vozlišči nimata enake oznake. Barvanje pomeni zgolj dodeljevanje različnih oznak za preprečevanje konfliktov.

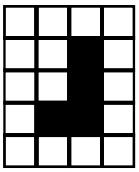
S problemom barvanja grafov se srečujemo na različnih področjih:

- razporejanje izpitov, kjer se skupine dijakov ne smejo prekrivati,
- barvanje zemljevidov, kjer morajo sosednje regije imeti različne barve,
- dodeljevanje frekvenc v omrežjih, da se prepreči interferenca,
- načrtovanje sedežev na dogodkih, da se ločijo gostje, ki se ne razumejo,
- reševanje ugank (npr. sudoku),
- dodeljevanje izhodov za letala ali peronov za vlake, kjer časovno prekrivajoči se leti ali vlaki ne smejo uporabljati istega vira.

Žigi

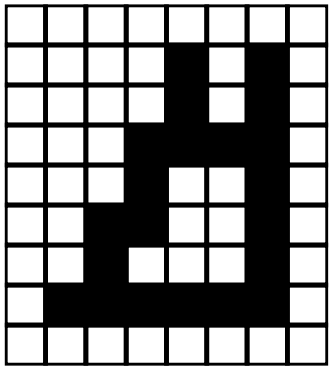
srednja šola

Borut ima gumijast žig, ki ustvarja zrcaljeno L-obliko iz štirih črnih kvadratov:



Žig večkrat uporabi na sprva praznem listu kvadratnega papirja. L-oblike se lahko delno prekrivajo, vendar le delno: Borut nikoli ne žigosa natanko na isti položaj dvakrat. Žig lahko obrne, vendar ga vedno uporabi tako, da se kvadrati žiga pokrivajo s kvadrati papirja. Naslednja slika prikazuje dve prekrivajoči se L-obliki, pri čemer je prekrivni kvadrat označen:

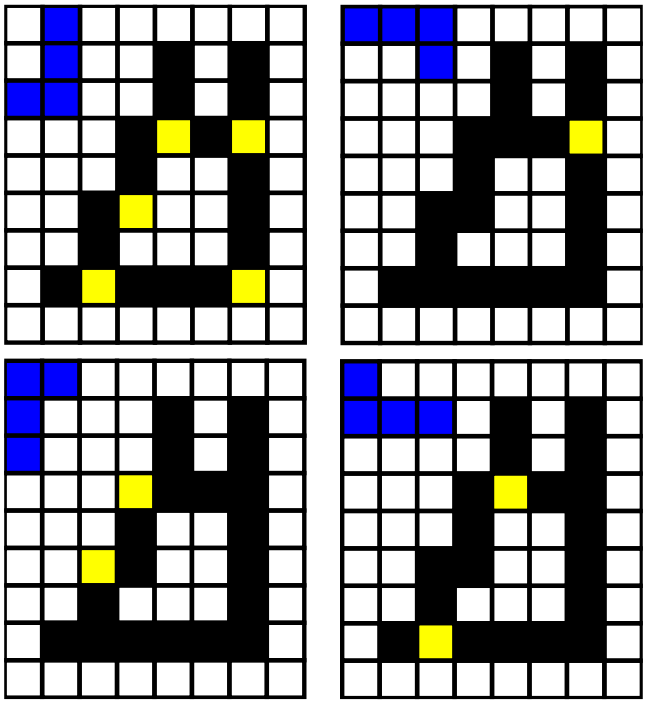
Največ koliko L-oblik je Borut lahko odtisnil na spodnji mreži?



Rešitev

Borut je lahko odtisnil največ 10 L-oblik.

Spodnje slike prikazujejo vse štiri možne rotacije žiga in pripadajoče možne položaje žigosanja vzorca. Vsak položaj žigosanja prikazuje rumeni kvadrat, ki označuje mesto, kamor je bil ob žigosanju postavljen vogalni kvadrat L-oblike. Vidimo, da je skupaj 5 + 1 + 2 + 2 = 10 možnosti.



Računalniško ozadje

Prepoznavanje vzorcev je postopek iskanja določenih elementov v sliki, na primer odtisov žigov. V resničnem svetu se prepoznavanje vzorcev uporablja v številnih vsakodnevnih situacijah, na primer pri samodejnem branju registrskih tablic ob vstopu v garažo. Prekrivanje vzorcev je tesno povezano z Boolovo operacijo OR pri bitnih vzorcih.

Pregled nalog

Državno tekmovanje

			6.	7.	8.	9.	SŠ
Cik-cak šifra	Poljska		●				
Čudežni otoki	Italija		●				
Čudovita naprava 1	Nemčija		●				
Čudovita naprava 2	Nemčija				●	●	●
Posredovanje sporočil	Kostarika		●				
Pozabljivi Albert	Združeno kraljestvo		●				
Razvrščanje vejic	Slovenija		●				
Skrivno sporočilo	Indonezija		●				
Varčna pot	Makedonija		●				
Avtonomni avtobus	Bolgarija		●		●	●	
Oskrbovalne odprave	Avstrija		●		●	●	
Robot sadi rože	Slovaška		●		●	●	
Sodobna pošta	Irska		●		●	●	
Šifriranje v dveh korakih	Slovaška		●		●	●	
Barvna polja	Saudova Arabija		●		●	●	●
Delo na cesti	Italija		●		●	●	●
Tri v vrsto	Brazilija		●		●	●	●
Oklepajoči okraski	Avstrija			●			
Robot	Slovaška			●			
Skrivnostna tipkonica	Malezija			●			
Speči bober	Japonska			●			
Stiskanje slik	Južna Koreja			●			
Nižanje stroškov	Nizozemska			●			
Zabava	Belgija			●			
Labirint	Švedska			●			
Levo-desno	Balgija			●			

			6.	7.	8.	9.	SŠ
Picerija	Švica			●			
Presedanje	Malezija			●			
Rušenje zidov	Slovaška			●			
Satovje	Švica			●			
Skakanje po štorih	Slovenija			●			
Žogice	Srbija			●			
Dure	Južna Koreja				●	●	
Izhod	Švica				●	●	
Nepravilno zaporedje	Švica						●
Črno-belo kodiranje	Portugalska				●	●	●
Izmenjava knjig	Indija				●	●	●
Obraba segmenta	Češka				●	●	●
Rover na Marsu	Belgija				●	●	●
Znamenitosti	Romunija				●	●	●
Bober Janez	Nemčija						●
Bobrov jezik	Irska						●
Čarobni gumb	Dominikanska republika						●
Javni prevoz	Litva						●
Levi	Nemčija						●
Razporejanje izpitov	Portoriko						●
Žigi	Finska						●

Avtorji nalog

Vania Natali in Husnul Hakim, Indonezija (Skrivno sporočilo)

Kirsten Schlüter, Nemčija (Čudovita naprava 1 in 2)

Maria Ines Gomez in Catalina Robles, Kostarika (Posredovanje sporočil)

Marta J. Burzanska, Poljska (Cik-cak šifra)

Carlo Bellettini, Violetta Lonati, Mattia Monga in Anna Morpurgo, Italija (Čudežni otoki)

Tina Semič, Slovenija (Razvrščanje vejic)

Združeno kraljestvo (Pozabljivi Albert)

Indonezija (Skrivno sporočilo)

Makedonija (Varčna pot)

Bolgarija (Avtonomni avtobus)

Avstrija (Oskrbovalne odprave)

Slovaška (Robot sadi rože)

Irska (Sodobna pošta)

Slovaška (Šifriranje v dveh korakih)

Saudova Arabija (Barvna polja)

Italija (Delo na cesti)

Brazilija (Tri v vrsto)

Južna Koreja (Dure)

Švica (Izhod)

Švica (Nepravilno zaporedje)

Portugalska (Črno-belo kodiranje)

Indija (Izmenjava knjig)

Češka (Obraba segmenta)

Belgija (Rover na Marsu)

Romunija (Znamenitosti)

Nemčija (Bober Janez)

Irska (Bobrov jezik)

Dominikanska republika (Čarobni gumb)

Litva (Javni prevoz)

Nemčija (Levi)

Tajvan (Razporejanje izpitov)

Finska (Žigi)

Soustvarjalci nalog

Eugenio Bravo, Španija

Špela Cerar, Slovenija

Andrew Csizmadia, Združeno kraljestvo

Susanne Datzko-Thut, Nemčija

Lucilla Dini, Italija

Husnul Hakim, Indonezija

Mette Eis-Hansen, Danska

Nont Kanungsukkasem, Tajska

Nora Anna Escherle, Švica

Yong Mao, Kitajska

Yi-Hsiu Mei, Tajvan

Hamed Mohebbi, Iran

Michael Weigend, Nemčija