Dabar 2019/20.

Задаци и решења за ученике виших разреда основне школе

– школско такмичење

(новембар, 2019.)

http://dabar.edu.rs/

Sadržaj:

Zadaci:

Draga deco i poštovane kolege,

Hvala Vam na želji, volji i entuzijazmu sa kojim pristupate ovom takmičenju! Ponosni smo na Vas i na činjenicu da se već 7. godinu zaredom družimo. U proteklom periodu na takmičenju je učestvovalo preko 250.000 takmičara, a na ovogodišnjem školskom nivou, preko 53.000. Ponovo smo, zajedno, pomerili granice i uključili veći broj dece nego prethodne takmičarske godine!

Posebno nas raduje činjenica da se sa nama družite od 1. razreda, pa do vašeg punoletstva i završetka srednje škole. DABAR je postalo TAKMIČENjE UZ KOJE ODRASTATE!

Takmičenje Dabar je namenjeno svim učenicima, ne samo talentovanim. Želja nam je, da kroz zabavne zadatke koje ste rešavali na školskom takmičenju 2019/20., ŠTO VIŠE DECE UVIDI DA SE SA INFORMATIČKIM PROBLEMIMA SUSREĆU U SVAKODNEVNOM ŽIVOTU I DA IH JE MOGUĆE SA LAKOĆOM REŠAVATI.

Priručnik je namenjen nastavnicima i učenicima kao pomoć pri bavljenju temama i intelektualnim problemima koji su predstavljeni kroz zadatke. Kako deca vole da se takmiče i vole da razmišljaju, naš posao je da ih i tokom godine podstičemo da razvijaju takmičarski duh i radoznalost.

Priručnik je i deo riznice Dabar intelektualnih problema, koja se iz godine u godinu uvećava. Pripručnik su pripremili organizatori takmičenja, kao nešto na šta smo ponosni ;).

U takmičenje je uloženo mnogo rada i energije, tako da nas posebno raduje što takmičenje postaje sve masovnije i popularnije, ne samo u našoj zemlji već i širom sveta.

Trenutno izdanje priručnika je "privremeno". Nakon narednog nivoa takmičenja, našoj riznica zadataka ćemo dodati nove. No, čak i oni će biti veoma brzo odrađeni od strane onih koji vole takve zadatke. Šta onda?

Pozivamo vas da pratite naše aktivnosti na sajtu dabar.edu.rs ili na sajtu Međunarodnog takmičenja Dabar bebras.org i da se zajedno sa nama radujete novim zadacima.

Uživajte u rešavanju zadataka!

Srdačno Vaš,

Programski odbor takmičenja Dabar

Bodovna tabela

Dabarčić

R.Br.	Zadatak	Težina	Bodovi
1.	Teren za golf	Lak	6
2.	Sef	Lak	6
3.	Čišćenje snega	Lak	6
4.	Ormarići za čuvanje stvari	Lak	6
5.	Bioskop	Srednje težine	9
6.	Red vožnje	Srednje težine	9
7.	Zastave	Srednje težine	9
8.	Prikaz slike	Srednje težine	9
9.	Transformacija reči	Težak	12
10.	Izgubljeni robot	Težak	12
11.	BenBen u kupovini	Težak	12
12.	Mašina za šivenje	Težak	12

Mladi dabar

R.Br.	Zadatak	Težina	Bodovi
1.	Prikaz slike	Lak	6
2.	Red vožnje	Lak	6
3.	Čišćnje snega	Lak	6
4.	Bioskop	Lak	6
5.	Crvenkapa	Srednje težine	9
6.	Recikliranje stakla	Srednje težine	9
7.	Transformacija reči	Srednje težine	9
8.	Izgubljeni robot	Srednje težine	9
9.	Kontrolor leta	Težak	12
10.	Video kompresija	Težak	12
11.	Kupovina cipela	Težak	12
12.	Quipu	Težak	12

Dabar

R.Br.	Zadatak	Težina	Bodovi
1.	Crvenkapa	Lak	6
2.	Reciklaža stakla	Lak	6
3.	Transformacija reči	Lak	6
4.	Popravi mog robota	Lak	6
5.	Kontrolor leta	Srednje težine	9
6.	Video kompresija	Srednje težine	9
7.	Mapa	Srednje težine	9
8.	Brojač	Srednje težine	9
9.	Presipanje vode	Težak	12
10.	Kupovina cipela	Težak	12
11.	Quipu	Težak	12
12.	Žurka	Težak	12

Stariji dabar

R.Br.	Zadatak	Težina	Bodovi
1.	Crvenkapa	Lak	6
2.	Reciklaža stakla	Lak	6
3.	Transformacija reči	Lak	6
4.	Popravi mog robota	Lak	6
5.	Kontrolor leta	Srednje težine	9
6.	Video kompresija	Srednje težine	9
7.	Mapa	Srednje težine	9
8.	Brojač	Srednje težine	9
9.	Presipanje vode	Težak	12
10.	Kupovina cipela	Težak	12
11.	Quipu	Težak	12
12.	Žurka	Težak	12

Izbor zadataka za takmičenje i prevod, Programski odbor takmičenja:

Milan Rajković (predsednik programskog odbora);
Marija Andonović Radojević (član programskog odbora);
Jelena Hadži-Purić (član programskog odbora);
Suzana Miljković (član programskog odbora);
Bojan Milosavljević (član programskog odbora);
Ivica Bekrić (član programskog odbora);
Saša Jevtić (član programskog odbora);
Nemanja Đorđević (član programskog odbora).

Tehnička podrška:

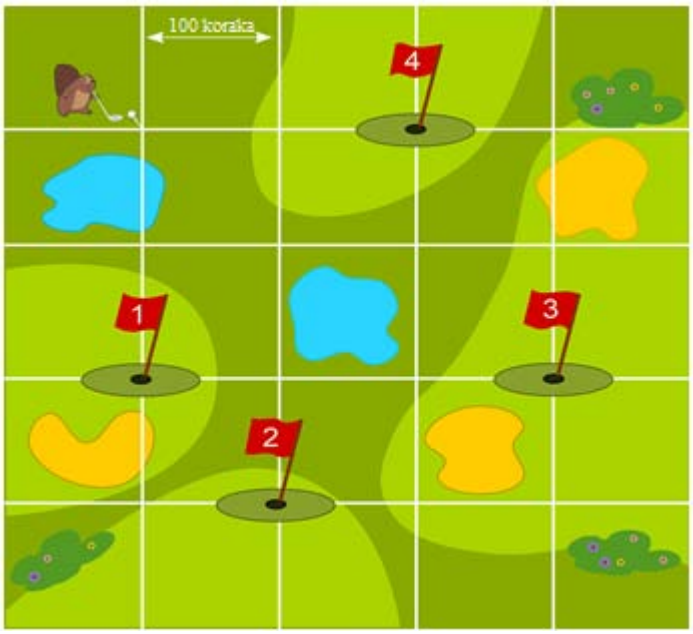
Gašper Fele-Žorž (Fakultet za računarstvo i informatiku, Univerzitet u Ljubljani)
Branislav Dolić

http://dabar.edu.rs/

Teren za golf

Dabar Sava igra golf na terenu prikazanom na slici ispod. Cilj je da dođe do svake rupice za golf, obeležene zastavicama, i to redom od prve do četvrte.

On želi da ide najkraćim putem, ali mora i da zaobiđe sve prepreke (grmlje, voda i pesak).



Pitanje / Izazov

Pretpostavimo da se Sava kreće isključivo linijama i da su sve linije dužine 100 koraka. Koliko će najmanje koraka Sava napraviti da bi ispunio svoj cilj?

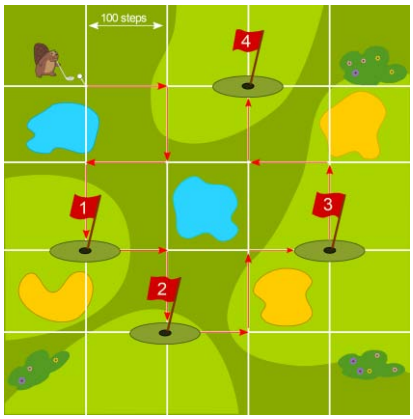
Odgovori:

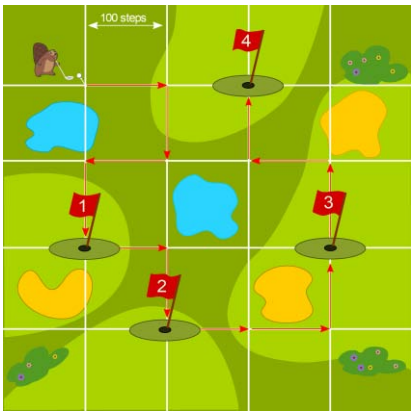
A) 1200
B) 1000
C) 1400
D) 600

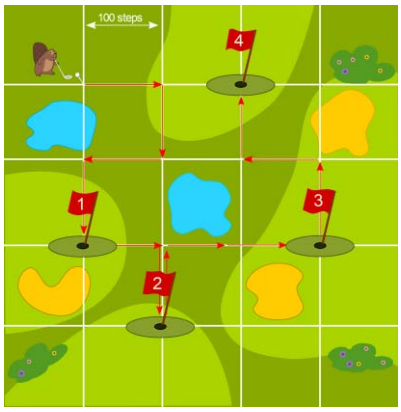
Tačan odgovor je:

Tačan odgovor je pod **A).**

Evo nekih mogućih ruta sa minimalnim brojem koraka koje bi Sava mogao da napravi, a koje ukupno traju 1200 koraka.







Informatička pozadina

Ovde postoji mehanizam za odlučivanje jer postoji niz i određena ograničenja tokom kretanja. Treba razmotriti ograničenja, a zatim odlučiti o ruti.

Evo mehanizma za odlučivanje:

1) Identifikujte najdirektniji put do sledeće zastave;

2) Ako je moguće, idite 100 koraka u tom pravcu. Ako ne, idite u jednom od dva vertikalna smera;

3) Vratite se na korak 1.

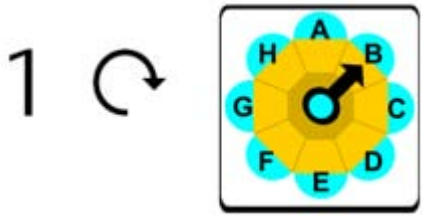
Dizajn mehanizma za odlučivanje poput ovog prilično je čest u računarskoj nauci. Zove se algoritam koji opisuje korake za postizanje rešenja ovog problema ili problema uopšte. Algoritmi su osnova računarskih programa.

Sef

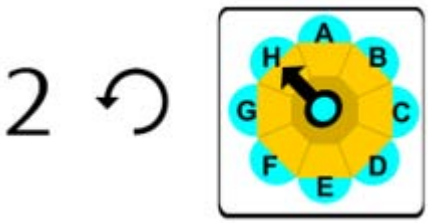
Dabar Pera je kuvar. On ima sef u kome čuva recepte. Sef se otvara pomoću kružne ručke. Ručka ima strelicu koja pokazuje na jedno od osam slova. Da bi otključao sef, Pera mora upisati lozinku koristeći strelicu. Ručka može da se okreće u smeru kazaljke na satu i u suprotnom smeru.

Početni položaj je:

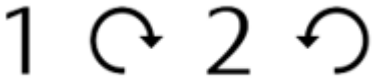
Ako pomerimo ručku za jednu poziciju u smeru kazaljke na satu, pokazivaće na slovo **B** .



Ako zatim pomerimo strelicu za dve pozicije u suprotnom smeru, pokazivaće na slovo **H** .



Brojevi označavaju broj pozicija, a strelice smer.



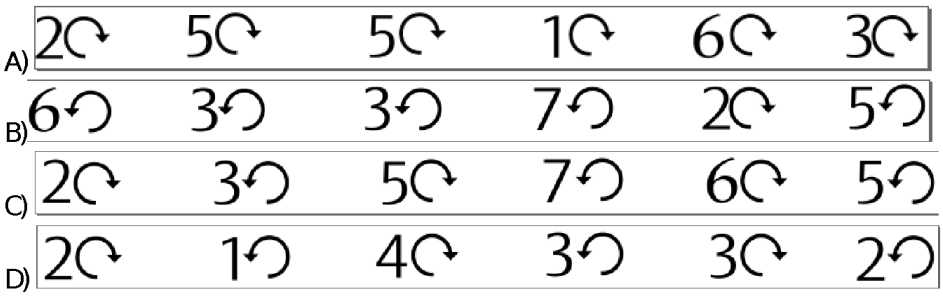
Ovaj primer unosi lozinku **BH** .

Pera želi da unese lozinku **CHEFDG** . Početni položaj ručke je prikazan na slici.

Pitanje/ Izazov

Koja od kombinacija će otvoriti sef?

Odgovori:



Tačan odgovor je:

Tačan odgovor je **C)**

Objašnjenje:

U **A)** i **B)** nije poštovano pravilo promene smerova.
C) je tačan odgovor.
D) bi dao tačnu lozinku samo kad bi se nakon svakog slova strelica vraćala u početni položaj, ali to nije slučaj.

Brava sefa počinje pamtiti položaj strelice tek nakon prvog pomicanja. Početni položaj nije uključen u lozinku.

Informatička pozadina

Kada radimo sa nekim objektom, često je potrebno pratiti njegov položaj ili stanje. Položaj strelice je deo stanja brave sefa u ovom zadatku.

Stanje objekta može uključivati prethodno izvedene okrete. Kako se ručka okreće, brava mora pamtiti koja slova lozinke su već upisana, a koja su deo početnog stanja brave.

Izvođenje istih radnji na objektu ne mora uvek imati isti učinak, učinci se mogu razlikovati zavisno od početnog stanja sistema. Npr. okretanje ručke na slovo **G** nakon unošenja „**CHEFD**“ će otvoriti bravu, ali ako se unese nakon „**CHE**“, brava se neće otvoriti.

Slično se može primeniti kod računara. Npr. kada crtate sliku, delovi koje ste već nacrtali su deo stanja te slike. Dodavanjem ili brisanjem linija stanje slike se menja pa računar mora pratiti te promene. Korišćenje alata za ispunjavanje bojom može promeniti boju većeg ili manjeg dela slike, zavisno od nacrtanih linija.

Primer je i hodanje uz korišćenje navigacije na mobilnom telefonu. Trenutna lokacija je stanje koje aplikacija na uređaju mora pratiti da bi dala ispravna uputstva i upozorila korisnika na grešku.

Aplikacija takođe može pamtiti koja ste mesta u prošlosti posećivali i predložiti nova mesta na osnovu tih podataka.

Kada piše u računarskom programu, programer mora odlučiti sa kojim stanjem sistema radi i napisati program tako da tačno prati promene stanja. Ako programer tu pogreši, program neće ispravno raditi.

Uočite kako je u ovom zadatku početni položaj uvek slovo **A**. Pre otključavanja sefa, ručka se uvek mora postaviti u položaj da strelica pokazuje slovo **A**. Ako nije dozvoljeno ručku okretati osam ili više puta, nemoguće je imati lozinku koja počinje slovom **A** jer, da bismo uneli **A**, najpre moramo okrenuti ručku na neko drugo slovo, što znači da lozinka počinje tim slovom.

Pri dizajnu sistema, programeri moraju izbegavati korišćenje ograničenja koja su korisniku nezgodna za rad.

Zastave

Dabrovi na svojim brodovima postavljaju zastave. Svaka zastava ima **dve trake** i **krug** između njih, kao na slici ispod:

Dabrovi na svakoj zastavi imaju neke od navedenih boja: **crvena, zelena** i **plava**. Gornja i donja traka zastave mogu biti iste ili različite boje, ali boja kruga u sredini mora imati različitu boju od boja traka.

Dabrovi su počeli da prave proračun, ali ih je loše vreme prekinulo.

Pitanje/ Izazov

Pomozite dabrovima da izračunaju koliko **različitih zastava** mogu napraviti.

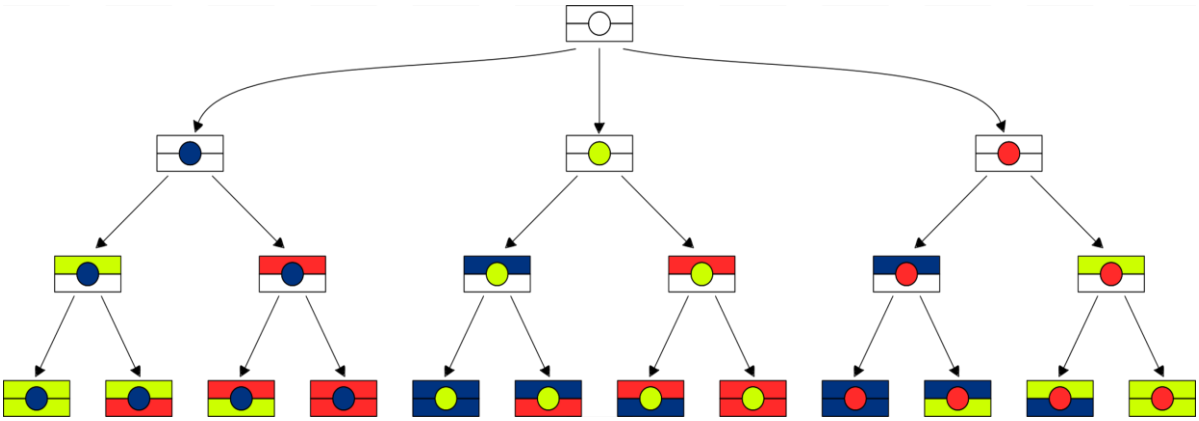
Odgovori:

A) 10
B) 12
C) 14
D) 16

Tačan odgovor je:

Tačan odgovor: **A) 12.**

Objašnjenje:



Uopšteno, svaka kombinacija boja je tačna sve dok:

- U drugom redu krajnji desni krug mora biti crvene boje.
- U trećem redu za svaku od tri boje kruga obe preostale boje su izabrane za gornju traku. Redosled nije važan.
- U četvrtom redu za svaku od boja kruga i za svaku boju gornje trake biraju se dve boje osim boje kruga. Redosled nije važan.

Informatička pozadina

U našem savremenom životu imamo različite zadatke. Neke od njih je lako rešiti, a drugima treba posvetiti više vremena. Zadaci koji zahtevaju mnogo vremena za rešenje nazivaju se teškim. Jedan od načina za rešavanje takvih zadataka je lista svih mogućih opcija i pronalaženje najboljeg rešenja. Zato je za računarskog naučnika ključno da zna kako da efikasno napiše sve opcije.

Ovo nabrajanje mora biti sistematično (tako da se nijedan obekat ili opcija ne gubi ili ponavlja) i jasno postavljeno (tako da je lako razumljivo drugim ljudima gde moraju postaviti svaku opciju, računarskim naučnicima i laicima). Ovo nije potrebno samo za proveru da li su pronađeni svi objekti ili opcije, već i da bi se procesima bilo moguće automatizovati programiranje.

Za izvođenje takvih operacija potrebno je koristiti odgovarajuće strukture podataka sa potrebnim rasporedom. Ako imamo neke karakteristike (delovi zastave) koji mogu imati različite vrednosti (boje), ove karakteristike numerišemo na način da su vrednosti karakteristika ograničene samo onima sa manjim brojem (trake zavise od boje kružnica, pa je krug prvi, numerišemo trake brojevima 2 i 3). Ako to možemo učiniti, obično koristimo "drveće" za predstavljanje svih opcija. "Drvo" je struktura koja ima koren i čvorove i krećemo se od korena do kraja grane. U svakom se čvoru krećemo različitim putanjama, u zavisnosti od vrednosti sledeće funkcije. Ovakav postupak nam pomaže da struktuiramo sve moguće vrednosti.

Više o "drveću" možete pročitati na: https://en.vikipedia.org/viki/Tree_(data_structure)

Ormarići za čuvanje stvari

U akva parku se nalazi mnogo ormarića za čuvanje stvari. Ormarići su povezani sa računarom tako da se u svakom trenutku zna da li je ormarić zauzet ili nije. Rezultati se upisuju u računar.

Tačno je **12:30** i podaci u računaru izgledaju ovako:

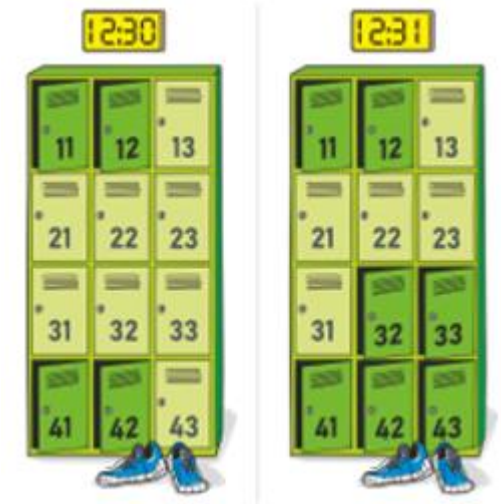
1 1 0 0 0 0 0 0 0 1 1 0 (pogledajte levu sliku ispod).

Nakon jednog minuta imaćemo sledeće podatke u računaru:

1 1 0 0 0 0 0 0 0 1 1 0 1 1 0 0 0 0 0 1 1 1 1 1 (pogledajte desnu sliku ispod).

Nakon još jednog minuta podaci u računaru izgledaju ovako:

1 1 0 0 0 0 0 0 0 1 1 0 1 1 0 0 0 0 0 1 1 1 1 1 1 1 0 0 0 0 0 0 0 1 1 0.



Pitanje/Izazov

Tačno je **12:32** . Kako izgledaju ormarići u ovom trenutku?

Ponuđeni odgovori:

A)

B)

C)

D)

http://dabar.edu.rs/

Tačan odgovor je:

Tačan je odgovor **A).**

Ormar ima dva stanja: zatvoreno ili otvoreno. Pomoću brojeva **0** i **1** možemo označiti stanja: **zatvoreno - 0**, **otvoreno - 1**.

Možemo ustanoviti da **0** znači **zatvoreno** i **1 otvoreno** gledajući početne podatke koje imamo. Upoređujući **1 1 0 0 0 0 0 0 0 1 1 0** sa stanjem ormara u 12:30 časova, možemo ustanoviti da se brojevi podudaraju sa statusom ormarića s leva na desno i od vrha do dna.

Set ormarića se sastoji od 12 ormarića. U primeru je niz od 24 broja. To znači da postoje stanja 12 ormarića pre i posle jednog minuta.

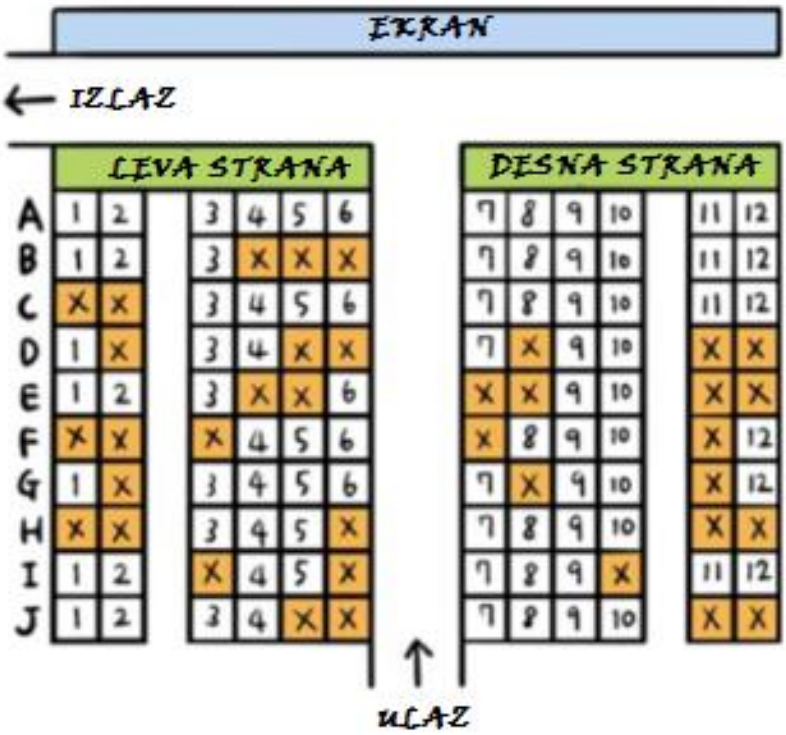
U pitanju je niz od 36 brojeva, a moramo analizirati samo poslednjih 12 brojeva**: 1 1 0 0 0 0 0 0 0 1 1 0**. To znači da su stanja ormarića: otvoreno, otvoreno, zatvoreno, zatvoreno, zatvoreno, zatvoreno, zatvoreno, zatvoreno, zatvoreno, otvoreno, otvoreno, zatvoreno. Tačan odgovor je tada A).

Informatička pozadina

Pojam predstavljanja podataka koji imaju samo dva stanja računari koriste za skladištenje svih vrsta informacija. Kompjuterski naučnici često predstavljaju ta dva stanja binarnim oznakama **1** i **0**, ali bilo koja dva druga simbola mogu se koristiti umesto njih, poput „otvoreno“ i „zatvoreno“ u ovom zadatku.

Biskop

Tri prijatelja dabra Milan, Nemanja i Saša biraju sedišta u bioskopu. Sedišta označena slovom **X** su zauzeta.



Svako od njih ima posebne želje:

- **Milan**: "Ja želim sedište na desnoj strani."
- **Nemanja**: "Želim da sedimo jedan pored drugoga u istom redu."
- **Saša**: "Ja ne želim da sedim blizu ekrana. Nećemo izabrati sedišta u prva tri reda."

Na primer, ako izaberu sedišta **G3, G4** i **G5**, tada će Milan biti nezadovoljan. Ako izaberu **D9** i **D10**, onda će Nemanja biti nezadovljan, a ako izaberu **A7, A8** i **A9**, onda će Saša biti nezadovljan.

Pitanje/Izazov

Koliko kombinacija za izbor sedišta imaju, a da svi budu zadovoljni?

Ponuđeni odgovori

A) 3
B) 4
C) 5
D) 6

Tačan odgovor je:

Odgovor je **D) 6**.

Objašnjenje

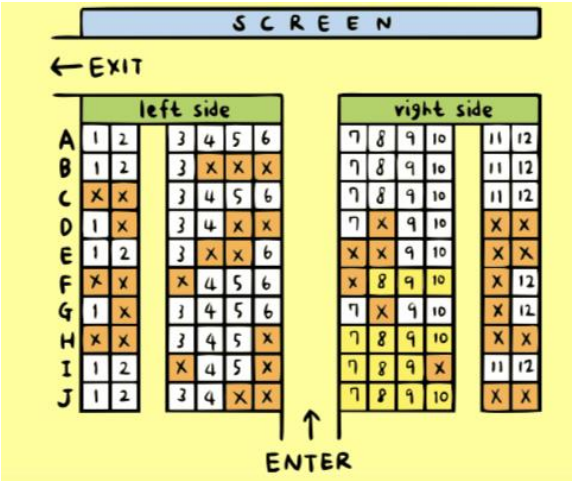
U ovom zadatku treba pronaći mesta za sedenje koja će zadovoljiti sva tri uslova (želje prijatelja).

Milan je rekao: “ Ja želim sedište na desnoj strani.” zato treba birati sedišta od 7. do 12. kolone.

Nemanja je rekao: “ Želim da sedimo jedan pored drugoga.” Zato treba pronaći tri slobodna mesta u jednom redu i birati sedišta u redovima **A**, **B**, **C**, **F**, **H**, **I** ili **J**. Ali, treba birati sedišta od 7. do 10. kolone.

“Ja ne želim da sedim blizu ekrana. Nećemo izabrati sedišta u prva tri reda”, **rekao je Saša**. Zato ne možemo izabrati sedišta od **A** do **C** reda. Treba izabrati od **D** do **J** reda.

Sedišta označena slovom **X** su zauzeta pa njih ne možemo izabrati. Ostaje nam 14 sedišta koja možemo izabrati, a da zadovoljimo sve želje prijatelja. To nam prikazuje slika ispod.



Na slici vidimo da imaju na raspolaganju 6 grupa po 3 sedišta koja zadovoljavaju sve postavljene uslove.

- (F8, F9, F10)
- (H7, H8, H9), (H8, H9, H10)
- (I7, I8, I9)
- (J7, J8, J9), (J8, J9, J10)

Informatička pozadina

Najbrži način pronalaženja rešenja u ovom zadatku je sistemom eliminacija u kojem odmah eliminišemo sedišta koja ne dolaze u obzir i od preostalih sedišta tražimo koja zadovoljavaju sva tri uslova. Ovakav način traženja rešenja koriste algoritmi brojnih aplikacija, kao na primer kod izbora sedišta u avionu nakon filtriranja želja kupaca.

Red vožnje

Sledeće tabele prikazuju kada će se autobusi zaustavljati na svakom autobuskom stajalištu:

Autobuska stanica	Linija 1	Linija 1	Linija 1
Stanica A	10:00	11:00	12:00
Stanica B	10:20	11:20	12:20
Stanica C	10:40	11:40	12:40
Stanica D	11:00	12:00	13:00
Stanica E	11:20	12:20	13:20

Autobuska stanica	Linija 2	Linija 2
Stanica A	10:10	11:10
Stanica F	10:20	11:20
Stanica C	10:30	11:30

Pitanje/Izazov

Ako je dabar Marko na **Stanici A** u **11:05**, u koliko sati će najranije stići do stanice **D** ?

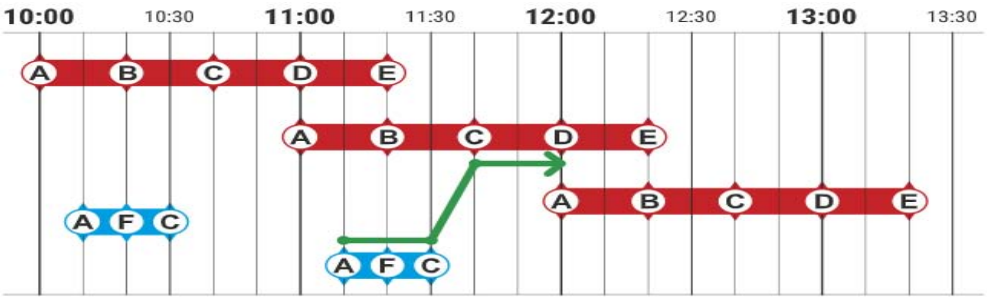
A) 12:00
B) 13:00
C) 12:20
D) 12:40

Tačan odgovor je:

Dabar Marko će se:

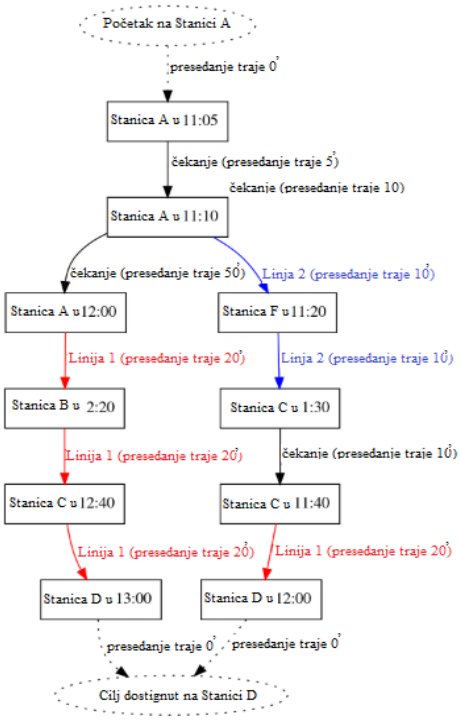
- Voziti autobusom **Linije 2** sa **Stanice A** u **11:05** do **Stanice C** u **11:30**,
- Voziti se autobusom **Linije 1** sa **Stanice C** u **11:40** do **Stanice D** u **12:00.**

Da je Marko putovao autobusom **Linije 1** sa **Stanice** A, on ne bi stigao do **13:00**.



Informatička pozadina

Uobičajeni pristup informatike je modeliranje problema kao „grafa“: dijagrama napravljenog kutijama i strelicama. Ovde je korisno koristiti okvire za beleženje doba dana i stanice na kojoj se nalazi Dabar Marko; strelica iz kutije u drugu može se koristiti za snimanje prelaznog vremena između situacije u kutiji i drugog. Ovo je rezultirajući grafikon:



Problem se sada može formulisati kao problem sa najkraćim putem, kao što je poznato u teoriji grafova: pronalaženje puta između dva vrha (polja) u grafikonu tako da je zbir težina (prelaznih vremena) njegovih sastavnih ivica (strelice) su svedene na minimum.

Od polja elipse na vrhu do elipse na dnu mogu se naći samo 2 staze: leva ima ukupnu težinu 0 + 5 + 50 + 20 + 20 + 20 + 0 = 115, a desna 0 + 5 + 10 + 10 + 10 + 20 + 0 = 55, dakle najkraći je pravi. (referenca: https://en.wikipedia.org/wiki/Shortest_path_problem)

Čišćenje snega

Mašina za čišćenje snega može se kontrolisati sledećim komandama:

FD : Mašina za čišćenje snega kreće se napred 100 metara;

RT : Mašina za čišćenje snega skreće desno;

LT : Mašina za čišćenje snega skreće levo.

Pitanje/Izazov

Koji redosled komandi treba zadati mašini za čišćenje snega da bi očistila sve ulice prikazane na slici ispod?

Ponuđeni odgovori:

A)	B)	C)	D)
FD	FD	FD	FD
RT	LT	LT	RT
FD	FD	FD	FD
RT	LT	RT	RT
FD	FD	FD	FD
RT	LT	RT	
FD	FD	FD	

Tačan odgovor:

B)

FD
LT
FD
LT
FD
LT
FD

Ako sledimo redosled naredbi, jedna za drugom lako ćemo primetiti da će samo ove komande očistiti sve ulice.

Informatička pozadina

Ako želimo da robot izvrši zadatak, moramo robotu dati skup naredbi ili uputstava. Komande su napisane na programskom jeziku. Komande moraju pratiti roboti u nizu.

Roboti obavljaju zadatke bez ljudske intervencije. Ovi zadaci se programiraju pomoću softvera Robot koji ima skup naredbi ili uputstava koji robotu govore koje zadatke treba da obavlja. Mnogi softverski sistemi i okviri predloženi su kako bi se programiranje robota olakšalo.

Skup naredbi ili uputstava u ovom slučaju napisan je na programskom jeziku LOGO.

Saznajte više na:

https://en.wikipedia.org/wiki/Logo_(programming_language)

https://en.wikipedia.org/wiki/Robot_software;

https://en.wikipedia.org/wiki/Computer_program

Prikaz slike

Slike na računaru podeljene su u mrežu malih kvadrata koji se nazivaju pikseli (elementi slike). Na crno-beloj slici svaki je piksel **crn** ili **beli**. Kada računar memoriše sliku, potrebno je memorisati koji su pikseli crne boje, a koji bele.

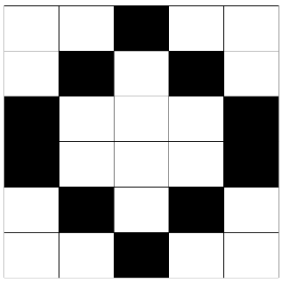
Na primer, slika slova „ **a** “ uvećana je, na slici ispod ovog teksta, pa prikazuje piksele. Jedan od načina predstavljanja ove slike je kodiranjem **1, 3, 1 – 4, 1 – 1, 4 – 0, 1, 3, 1 – 0, 1, 3, 1 – 1, 4**, gde svaki red uvek počinje kodiranjem broja belih piksela, a " – " označava kraj reda.

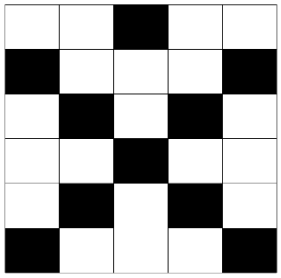
Pitanje/Izazov

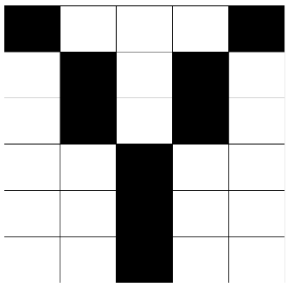
Primenjujući isto kodiranje kao u gornjem primeru, koja od sledećih slika je prikazana kodom:

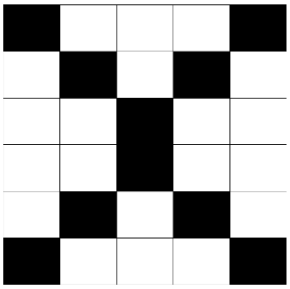
2, 1, 2 – 0, 1, 3, 1 – 1, 1, 1, 1, 1 – 2, 1, 2 – 1, 1, 1, 1, 1 – 0, 1, 3, 1 ?

Ponuđeni odgovori:

A) 

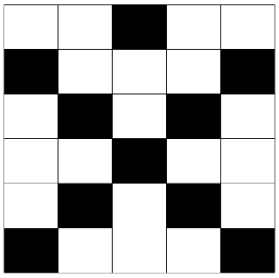
B) 

C) 

D) 

Tačan odgovor je:

b)



Objašnjenje:

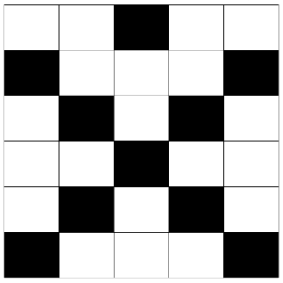
Slika u primeru pokazuje da brojevi predstavljaju broj uzastopnih piksela u istoj boji (počevši od belih piksela) unutar reda s leva na desno.

U gornjem primeru prvi skup brojeva je **1**, **3**, **1**. Zato prvi red sadrži jedan beli, tri crna, a zatim jedan belia piksel, sa leva na desno.

Možemo odrediti koji piksel sadrži svaki red, s leva na desno:

- Prvi red (2, 1, 2) znači 2 bela, 1 crni, 2 bela piksela.
- Drugi red (0 ,1, 3, 1) znači 0 belih, 1 crni, 3 bela, 1 crni piksel.
- Treći red (1, 1, 1, 1, 1) znači 1 beli, 1 crni, 1 beli, 1 crni, 1 beli piksel.
- Četvrti red (2, 1, 2) znači 2 bela , 1 crni, 2 bela piksela.
- Peti red (1, 1, 1, 1, 1) znači 1 beli, 1 crni, 1 beli, 1 crni, 1 beli piksel.
- Šesti red (0, 1, 3, 1) znači 0 belih, 1 crni, 3 bela, 1 crni piksel.

Na osnovu tih podataka možemo odrediti šta slika predstavlja:
2, 1, 2 – 0, 1 ,3, 1 – 1, 1, 1 ,1, 1 – 2, 1, 2 – 1, 1, 1, 1, 1 – 0, 1, 3, 1 predstavlja:



Informatička pozadina

U digitalnoj slici, piksel je mali element slike. Reč piksel dolazi iz kombinacije „pix“ (picture – „slika“) i „el“ (element – „element“). Količina piksela na slici predstavlja rezoluciju slike. Rezolucija se meri ukupnim brojem vertikalno i vodoravno položenih piksela na slici.

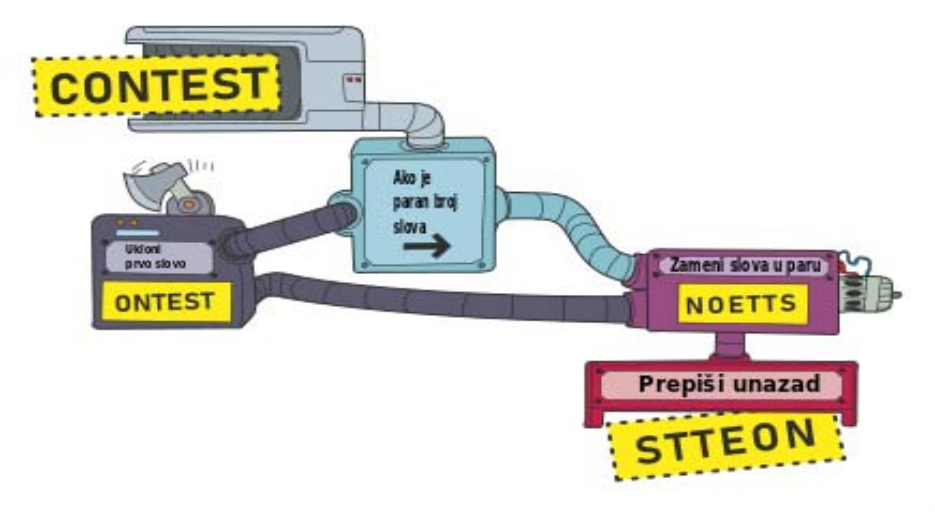
Uobičajno su slike kodirane u nekom obliku, odnosno predstavljene su na određeni način u memoriji računara. Ovaj konkretan zadatak prikazuje jedan takav primer kodiranja, poznat kao ***Run-length encoding***, gde smo izostavljali vrednosti kodiranih vrednosti, jer znamo da imamo samo naizmenične bele i crne piksele.

https://en.wikipedia.org/wiki/Run-length_encoding

Transformacija reči

Dabrovi imaju mašinu koja transformiše reči na sledeći način: ako je broj slova paran, reč se šalje dalje na obradu, u suprotnom, prvo slovo reči se uklanja. Zatim se menja svaki par susednih slova: prvo i drugo slovo, zatim treće i četvrto slovo i tako dalje. U poslednjem koraku, dobijena reč se preokreće.

Na primer, reč **CONTEST** se pretvara u **STTEON** .



Pitanje/Izazov

Ako je početna reč **STORAGE**, koju reč ćemo dobiti posle transformacije?

Ponuđeni odgovori:

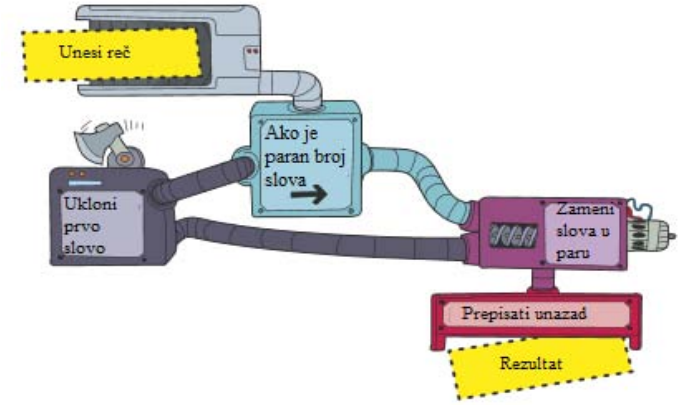
A) GERATO
B) TORAGE
C) OTAREG
D) GEARTO

Tačan odgovor je

A) GERATO.

Kako reč **STORAGE** ima neparan broj slova prvo će se ukloniti prvo slovo i dobiti reč **TORAGE**. Sledeći korak je razmenjivanje slova u paru i dobijanje **OTAREG**. Finalni rezultat se dobija obrtanjem niza **OTAREG** : **GERATO.**

Informatiča pozadina

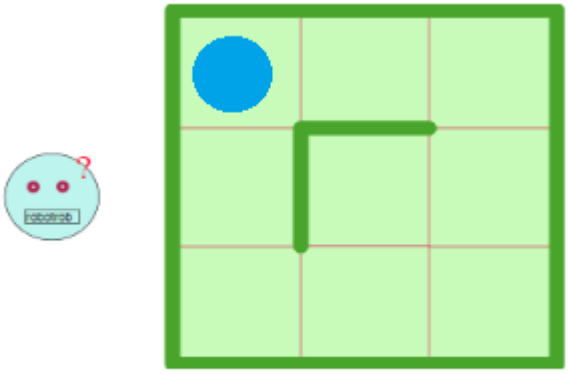


Slika (šema) u ovom zadatku deo je jednostavnog dijagrama toka, koji objašnjava kako promeniti reč korak po korak. Dijagrami toka su način da se opišu algoritmi: primetite da dijagrami toka mogu zabeležiti ideju odluka (kao u delu „AKO DALJE“ mašine) ili izvršiti akcije (poput REMOVE ili SWAP delova mašine). Imajte na umu da dijagrami toka takođe mogu omogućiti petlje/ponavljanje, što bi bilo predstavljeno pomeranjem „nazad“ na raniji deo mašine.

U ovom zadatku algoritam menja reči.

Popravi mog robota

Nataša je izgubila robota u parku. Park je predstavljen kao na slici ispod. Robot je mogao da se izgubi bilo gde u parku. Nataša može, preko daljinskog upravljača, da pošalje niz naredbi robotu. Jedna naredba može saopštiti robotu da se pomeri u susedni kvadrat gore, levo, desno ili dole. Ako robot naiđe na zid, neće moći da ide dalje, i čekaće sledeću naredbu. Zidovi su na slici obeleženi debelom (zelenom) linijom.



Pitanje / Izazov

Nataša ne zna gde je robot. Koji je najkraći niz naredbi koje ona može poslati robotu, tako da dođe do kvadrata sa plavom tačkom?

Odgovori:

A) Dole - Levo - Dole - Levo - Gore - Gore
B) Desno - Gore - Gore - Levo - Levo
C) Desno - Gore - Desno - Gore - Gore - Levo
D) Gore - Desno - Gore - Levo – Levo

Tačan odgovor je:

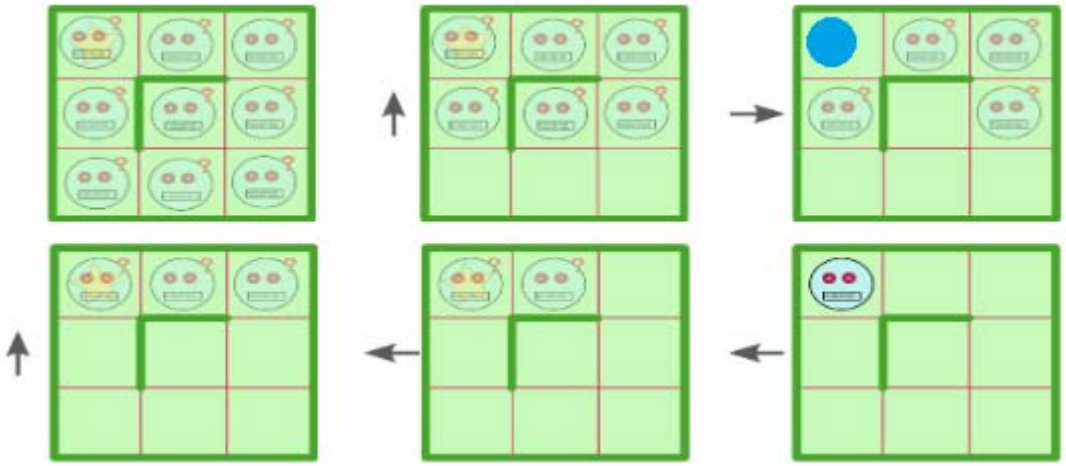
Tačan odgovor je: D) Gore - Desno - Gore - Levo – Levo.

Da potvrdimo tačan odgovor D), izvršićemo redosled naredbi, sa svih mogućih početnih pozicija. Nakon svakog koraka mogući se položaji robota menjaju i na kraju postoji samo jedan mogući položaj za robota.

Nemoguće je pronaći rešenje sa 4 koraka, jer za kvadrat u donjem desnom uglu su potrebne četiri komande (Gore - Gore - Levo - Levo ili Levo - Levo - Gore - Gore). Ove komande ne čine da robot iz centra krene do željenog kvadrata.

Iako odgovori A i C takođe čine da se robot kreće do željenog kvadrata, zahtevaju duži niz naredbi.

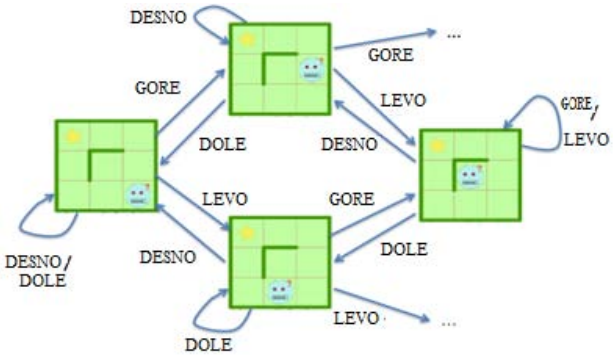
Odgovor B je pogrešan, jer robota ne usmerava da postigne željeni kvadrat ako se izgubio u donjem levom kvadratu.



Informatička pozadina

Ovo je zadatak o sinhronizaciji reči : https://en.wikipedia.org/wiki/Synchronizing_word u konačnim automatama.

U svakom trenutku, robot je na jednom kvadratu i može primiti komandu koju mu je poslala Nataša. Nakon svake naredbe ili menja kvadrat ili ostaje na istom kvadratu. Položaj robota je ono što nazivamo stanjem i zato naredba može promeniti stanje. Ove promene stanja možemo predstaviti grafički, crtežom kao sledeći (koji ne pokazuje sve moguće promene stanja):

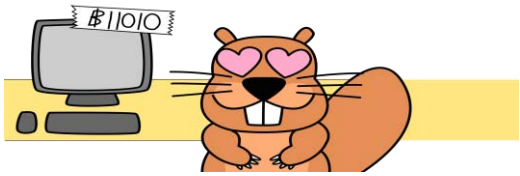


Takav prikaz je ono što nazivamo konačni automati. S obzirom na početno stanje, možemo izvršiti niz naredbi da bismo otkrili šta će biti dostignuto stanje. Takav niz je ono što nazivamo rečju. Da sumiramo, možemo izvršiti reč na ograničenom automatu, da bi prešli iz jednog u drugo stanje. Za ovaj problem ne znamo početno stanje. Ono što tražimo je reč koju treba izvršiti tako da uvek dođemo do istog konačnog stanja, bez obzira na početno stanje. To je upravo ono što se naziva sinhronizirajućom rečju. Izvođenje reči za sinhronizaciju uvek dovodi do istog konačnog stanja, bez obzira na početno stanje.

BenBen u kupovini

Dabar BenBen želi da kupi računar. On ima po jednu novčanicu sledećih vrednosti: **1 , 10 , 100 , 1000** i **10000** dinara. On želi da kupi računar ali tako da, kada plati, ne dobije kusur.

Odlučio je da kupi računar koji košta **11010** dinara.



Pitanje / Izazov

Ako poređamo sve moguće cene, počev od najveće, koje BenBen može da plati bez vraćanja kusura, koja po redu je cena od 11010 dinara?

Odgovori:

A) Na petom mestu
B) Na šestom mestu
C) Na sedmom mestu
D) Na osmom mestu

Tačan odgovor je:

B) Na šestoj poziciji.

Redosled cena koje BenBen može da plati bez kusura je:

1: 11111
2: 11110
3: 11101
4: 11100
5. 11011
6: 11010…

Dakle 11010 je 6-ti. najskuplji računar koji BenBen može kupiti bez kusura.
Cene ostalih računara u ponuđenim odgovorima su:
A) 11011
C) 10101
D) 11000

Informatička pozadina

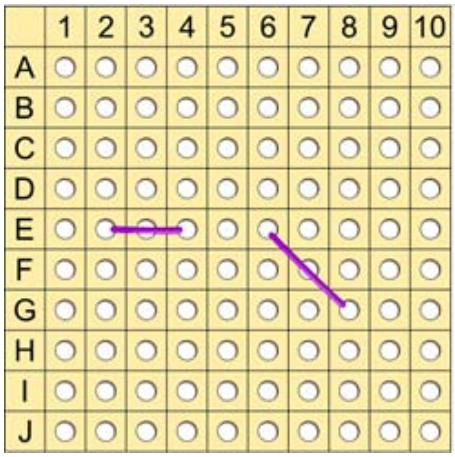
Ovaj zadatak ima zanimljivu vezu sa konceptom binarnih brojeva. Zbir BenBen-ovih plaćanja uvek se sastoji samo od broja 0 i 1, tako da cena može biti predstavljena kao binarni broj. Redosled pada cena koje može platiti je upravo silazni redosled 5-bitnih binarnih brojeva.

Mašinski vez

Dabar želi da sastavi program za svoju mašinu za šivenje kojom će šiti različite modele odeće. Za izradu programa dabar koristi naredbu **OUT(cc)-IN(dd)** , pri čemu cc i dd označavaju položaj igle.

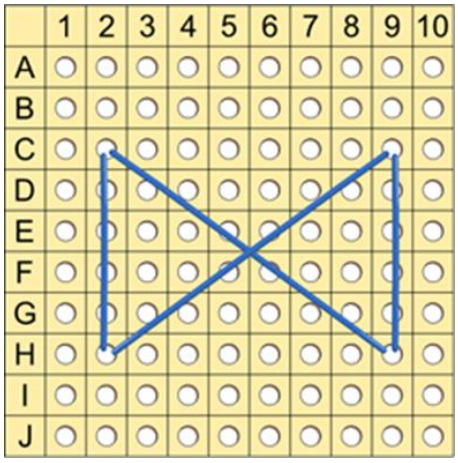
Na primer, naredba **OUT(B2)-IN(A3)** će pomeriti iglu na položaj **B2** , probosti platno i provući konac odozdo prema gore, a nakon toga se pomeriti na položaj A3 , probosti platno i provući konac odozgo prema dole.

Za izradu šablona prikazanog na slici ispod potrebne su naredbe **OUT(E6)-IN(G8);OUT(E2)-IN(E4)** .



Pitanje/Izazov

Koje naredbe će stvoriti šablon prikazan na slici ispod?



Ponuđeni odgovor:

A) OUT(H2)-IN(C2);OUT(H9)-IN(C9);OUT(C9)-IN(C2);OUT(H9)-IN(C2)
B) OUT(C2)-IN(H9);OUT(H2)-IN(C9);OUT(C2)-IN(H2);OUT(C9)-IN(H9)
C) OUT(H9)-IN(C9);OUT(H9)-IN(H2);OUT(C2)-IN(H2);OUT(C9)-IN(H2)
D) OUT(C2)-IN(C9);OUT(H2)-IN(H9);OUT(C2)-IN(H2);OUT(C9)-IN(H9)

Tačan odgovor je:

Tačan odgovor je: B

Objašnjenje:

Da bi se stvorio zadati šablon potrebne su 4 naredbe, po jedna za svaku liniju veza, pri čemu redosled veženja nije važan. Te naredbe su:

- **OUT(C2)-IN(H9) ili OUT(H9)-IN(C2)**
- **OUT(H2)-IN(C9) ili OUT(C9)-IN(H2)**
- **OUT(C2)-IN(H2) ili OUT(H2)-IN(C2)**
- **OUT(C9)-IN(H9) ili OUT(H9)-IN(C9)**

Od ponuđenih odgovora, odgovor **B** sadrži sve potrebne naredbe.

U odgovoru A nalazi se naredba **OUT(C9)-IN(C2**) koja veze liniju koja nije potrebna, dok istovremeno nedostaje naredba **OUT(H2)-IN(C9)** ili **OUT(C9)-IN(H2).**

Odgovor **B** je tačan.

U odgovoru **C** nalazi se naredba **OUT(H9)-IN(H2)** koja veze nepotrebnu liniju, a nedostaje naredba **OUT(C9)-IN(H9**) ili **OUT(H9)-IN(C9**).

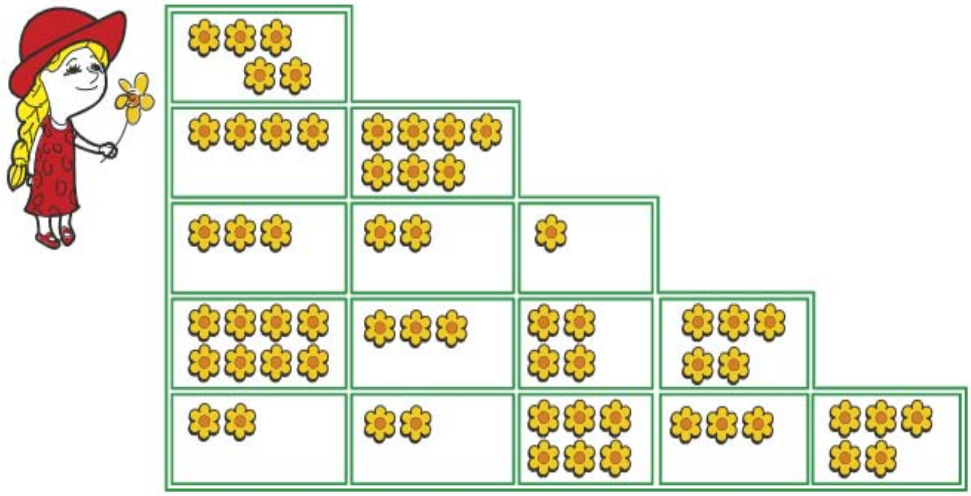
U odgovoru **D** nalaze se naredbe **OUT(C2)-IN(C9)** i **OUT(H2)-IN(H9)** koje vezu dve nepotrebne linije, a nedostaju naredbe **OUT(C2)-IN(H9)** ili **OUT(H9)-IN(C2)** i naredba **OUT(H2)-IN(C9)** ili **OUT(C9)-IN(H2)**.

Informatička pozadina

Algoritam opisuje korake koje treba slediti kako bi se izvršio neki zadatak. Algoritmi su uobičajni u informatici, ali se koriste i za rešavanje problema u svakodnevnom životu. Ovaj zadatak je primer kako se algoritam može koristiti za stvaranje šeme za veženje.

Crvenkapa

Crvenkapa želi da ubere cveće iz bašte svoje bake. Bašta je podeljena na nekoliko delova, a u svakom delu zasađen je određeni broj cvetova. Crvenkapa započinje svoj put od dela u gornjem levom uglu i kreće se do dela u donjem desnom uglu, **i kreće se samo naniže ili udesno**.



Pitanje/Izazov

Koji je maksimalni broj cvetova koje može da ubere?

Odgovori:

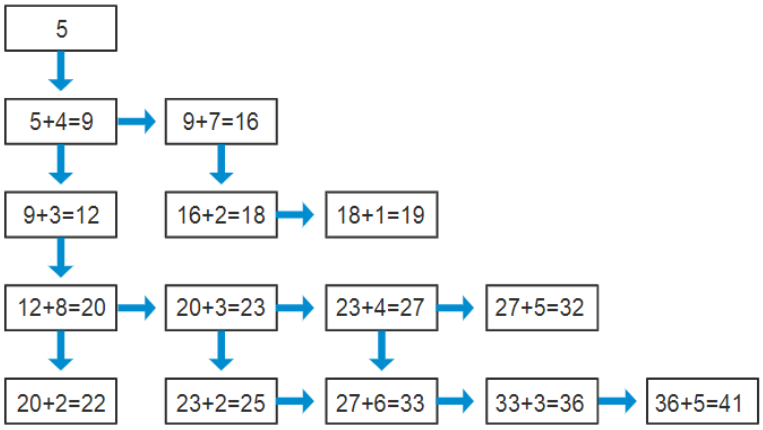
A) 35
B) 38
C) 58
D) 41

Tačan odgovor je:

D)

U svakom delu možemo izračunati maksimalni broj cvetova koje Crvenkapa može da ubere od početka do tog dela (nazovimo to "Maksimum"). Maksimum se može izračunati na sledeći način:

- Ako se do dela može doći samo samo iz jednog prethodnog dela, onda je njegov maksimum zbir cvetova u tom delu i maksimum tog prethodnog dela.
- Ako se do tog dela može doći iz dva prethodna dela, onda je njegov maksimum zbir cvetova u tom delu i veći od ta dva prethodna dela.



Informatička pozadina

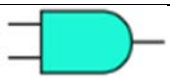
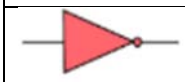
Ovo je primer primene takozvane strategije „dinamičkog programiranja" za rešavanje problema optimizacije. Princip dinamičkog programiranja je da se za podprobleme izračunavaju posredni rezultati. U našem slučaju to je najveći broj cvetova sakupljenih od početne tačke do bilo kojeg intermedijarnog dela. Trik je da se za izračunavanje novog prelaznog rezultata koristi nekoliko već izračunatih prelaznih rezultata. Primenjujući ovu ideju, "poplavni talas" već izračunatih srednjih rezultata potiče od početne tačke. Kada dostigne krajnju tačku, tada imamo željeni krajnji rezultat. Mnogi problemi sa optimizacijom mogu se rešiti ovom strategijom, koja donosi vrlo često algoritme koji su veoma vremenski efikasni.

Reckliranje stakla

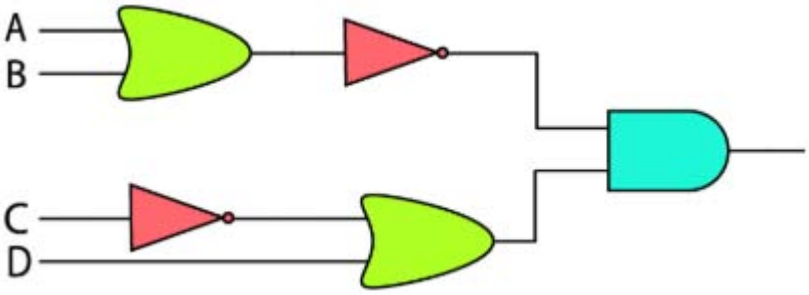
Dabrovi su veoma posvećeni recikliranju staklenih flaša. Flaše mogu biti bele ili obojene .

Oni koriste različite mašine koje na različite načine recikliraju flaše i pretvaraju ih u staklo koje takođe može biti belo ili obojeno .

Vrste mašina i načini recikliranja su prikazani u tabeli ispod:

	U ovu mašinu se unose dve staklene flaše. Belo staklo će se dobiti samo ako se unesu dve bele flaše. U ostalim slučajevima će se dobiti obojeno staklo.
	U ovu mašinu se unose dve staklene flaše. Obojeno staklo će se dobiti samo ako se unesu dve obojene flaše. U ostalim slučajevima će se dobiti belo staklo.
	Ova mašina će pretvoriti obojenu flašu u belo staklo ili belu flašu u obojeno staklo.

Napravili su sledeći sistem:



Pitanje/Izazov

Koje vrste staklenih flaša se moraju uneti u mašine na ulaze A , B , C i D , tako da se dobije belo staklo?

Odgovori:

A) A = bela, B = bela, C = obojena, D = bela
B) A = obojena, B = obojena, C = obojena, D = bela
C) A = bela, B = obojena, C = obojena, D = bela
D) A = obojena, B = obojena, C = bela, D = obojena

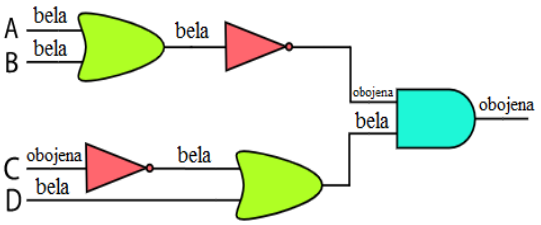
Tačan odgovor je:

B)

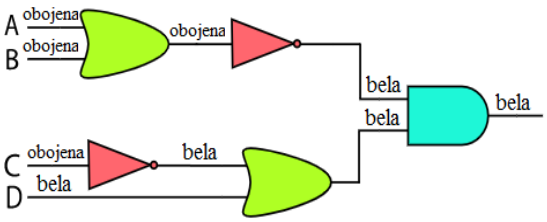
Objašnjenje:

Možemo nacrtati slike za svaku opciju i napisati vrstu stakla iznad svake od linija (unetih i proizvedenih). Tako možemo proveriti izlaz u svakoj opciji i pronaći tačan. Jedina slika ovih opcija sa belim staklom kao izlazom je: Slika 2.

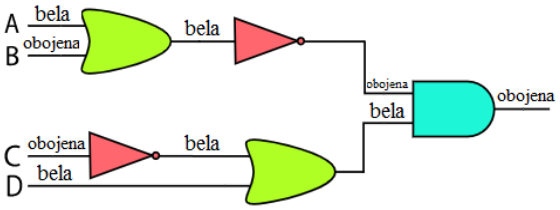
Slika 1:



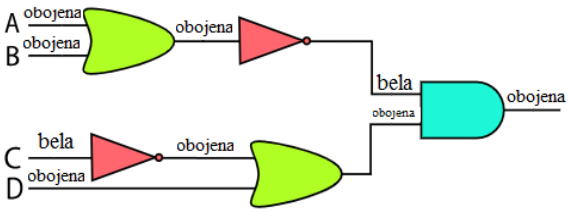
Slika 2:



Slika 3:



Slika 4:



Imajte na umu da postoje tri moguća ulaza koji proizvode belo staklo kao izlaz:

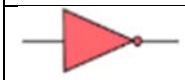
A = obojeno, B = obojeno, C = obojeno, D = obojeno
A = obojeno, B = obojeno, C = obojeno, D = bela0 (ovo je odgovor 2 gore)
A = obojeno, B = obojeno, C = belo, D = belo

Informatička pozadina

Svi računari sadrže sklopove koji se sastoje od različitih vrsta malih elemenata zvanih **logička kola**. Neke od najpopularnijih logičkih kola su **NOT**, **OR** , **AND**, **XOR**.

Ovde imamo **AND**, **OR** i **NOT** logička kola (njihov grafički prikaz u zadatku je isti kao u Inženjeringu).

U inženjerstvu elementi rade koristeći električne signale. Beležimo ih kao **1** ako postoji signal, i **0** ako nema signala. Logički označavamo signal **1** kao **TRUE**, a signal **0** kao **FALSE**. U ovom zadatku smatramo da je belo staklo **TRUE** (ili **1**), a obojeno staklo **FALSE** (ili **0**)

	**AND** logičko kolo zahteva da oba ulaza budu **TRUE** (belo staklo) da bi se dobio **TRUE** kao izlaz.
	**OR** logičko kolo zahteva da bar jedan od ulaza bude **TRUE** da bi se dobio **TRUE** kao izlaz.
	**NOT** logičko kolo preokreće vrednosti, tako da ako je ulaz **TRUE**, izlaz je **FALSE**, a ako je ulaz **FALSE**, izlaz je **TRUE**.

Logička kola se između ostalog koriste i u arhitekturi mikroprocesora kako bi se izvodile aritmetičke i logičke operacije.

Kontrolor leta

Kada avion sleće na aerodrom, njemu se dodeljuje određena pista. Na aerodromu postoji **8 različitih pista** za sletanje aviona.

Na aerodromu Bebrasland dva aviona ne mogu koristiti istu pistu ukoliko je razlika planiranih vremena sletanja **manja od 15 minuta** .

Na primer, ako **avion broj 1** sleće u **6:10** , **avion broj dva** sleće u **6:24**, a **avion broj 3** sleće u **6:26** tada avion broj 1 i avion broj 2 ne mogu da slete na istu pistu. **Avionu broj 3** može se dodeliti ista pista kao **avionu broj 1** , ali ne i ista pista kao **avionu broj 2** .

Danas ste kontrolor vazdušnog saobraćaja na aerodromu i vaš zadatak je da odredite najmanji broj pista za avione čije je vreme sletanja prikazano u donjoj tabeli.

Let	Vreme
avion 1	7:00
avion 2	7:21
avion 3	7:20
avion 4	7:18
avion 5	7:03
avion 6	7:12

Pitanje/Izazov

Koji je najmanji broj pista potreban da bi se omogućilo da svi avioni slete u skladu sa pravilima?

Ponuđeni odgovori:

A) 4
B) 3
C) 5
D) 7

Tačan odgovor je:

Tačan odgovor: A) 4

Objašnjenje:

Lista aviona na osnovu vremena sletanja:

1. Avion 17:00
2. Avion 57:03
3. Avion 67:12
4. Avion 47:18
5. Avion 37:20
6. Avion 27:21

Prvo sleće **avion 1** u 7:00, pa ga dodeljujemo pisti 1. Sledeća dva, **avion 5** i **avion** 6 sleću u roku od 15 minuta od sletanja **aviona 1**, a **avion 6** takođe sleće u roku od 15 minuta od **aviona 5**, tako da je potrebno dodeliti tri odvojene piste ovim avionima. Dakle, do sada su nam potrebne 3 piste.

Svaki sledeći avion pokušamo dodeliti već korišćenim pistama ako je moguće. **Avion 4** sleće u 7:18, njegovo vreme sletanja je više od 15 minuta udaljeno od sletanja **aviona 1** i može mu se dodeliti ista pista kao **avionu 1**. Slično tome, **avionu 3** ne može se dodeliti pista 1 jer sleće u roku od 15 minuta od **aviona 4**, ali može mu se dodeliti pista 2, jer je vreme sletanja više od 15 minuta udaljeno od sletanja **aviona 5**.

Avion 2 u 7:21, vreme sletanja manje je od 15 minuta od sletanja **aviona 6** (7:12), **aviona 4** (7:18) i **aviona 3** (7:20) koji su dosad dodeljeni pisti 3. Moramo odrediti još jedau pistu za ovaj let.

Na taj način dobili smo sledeći redosled sletanja aviona na piste

Pista 1: avion 1 (7:00), avion 4 (7:18)
Pista 2: avion 5 (7:03), avion 3 (7:20)
Pista 3: avion 6 (7:12)
Pista 4: avion 2 (7:21)

Iako možemo da dodelimo više od 4 pista, jasno je da može doći do nesreća sa manje od 4.

Informatička pozadina

U gornjem problemu imate “relacije sukoba” (dva aviona ne mogu koristiti isti pistu ako sleću u roku od 15 minuta jedan od drugog) i resurse (pista) i morate dodeliti resurse vodeći računa o “relacijama sukoba”. Postoje mnogi problemi ovog tipa. Na primer, kada treba odrediti koliko stolova je potrebno na venčanju, da bi ste goste rasporedili na takav način da oni koji nisu u dobrim odnosima ne sede za istim stolom.

Kada vaš problem ima puno parametara (avion, gosti,…), to želite da brzo rešite. U informatici su razvijeni posebni pristupi za rešavanje takvih problema.

Jedan od njih je da se graf koristi za predstavljanje. Graf ima čvorove i ivice koje povezuju čvorove. U ovom slučaju čvorovi predstavljaju objekte (letove) i ivice sukoba: ako su dva objekta međusobno u sukobu, njihovi čvorovi povezani su ivicom, a u suprotnom nisu. Problem zadatka može se smatrati problemom bojenja grafa: koji je minimalni broj boja potreban za bojenje svakog čvora u grafu na takav način da ne postoje isti obojeni čvorovi povezani ivicom

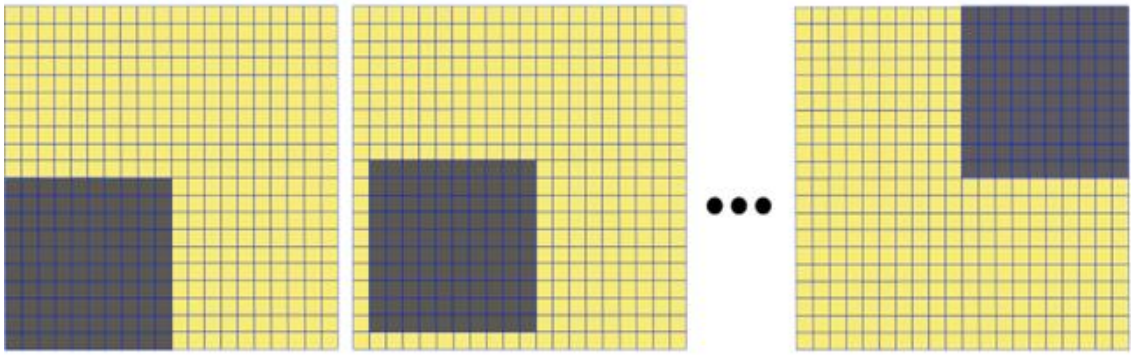
.

Kada se primeni na problem s planiranjem sletanja, svaka boja će predstavljati pistu.

Bojenje grafa ima primene poput usmeravanje aviona ka rutama letova, dodeljivanja propusne širine radio stanicama, usklađivanje obrazaca, planiranje sportskog rasporeda, dizajniranje planova za sedenje, zakazivanje ispita, pa čak i rešavanje SudoKu zagonetki.

https://en.vikipedia.org/viki/Graph_coloring

Video kmpresija



Računarska slika je pravougaona mreža obojenih kvadrata, zvanih **pikseli**. Video je niz slika, zvanih frejmovi, svaki malo drugačiji od prethodnog. Najjednostavniji način snimanja video zapisa je čuvanje svih piksela u svakom frejmu. Efikasniji način je da sačuvate celokupan prvi frejm, a zatim samo da sačuvate one piksele koji menjaju trenutni frejm od sledećeg.

Na gornjoj slici kvadrat **10×10** tamne boje pomera se s donjeg levog ugla u gornji desni ugao svetlo obojenog **20×20** polja, pomerajući se jedan piksel horizontalno i vertikalno unutra svakog frejma. Ovo traje **11** frejmova. Ako ovaj video snimimo u jednostavnom formatu, za to će biti potrebno **(20 × 20) × 11 = 4400** piksela.

Pitanje/Izazov

Ako snimimo ovaj video u efikasnijem formatu, koliko nam je potrebno piksela?

Ponuđeni odgovori:

A) 780
B) 100
C) 290
D) 380

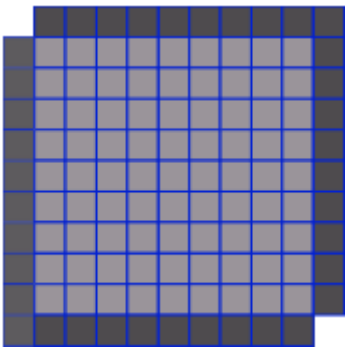
Tačan odgovor je:

Tačan odgovor je: A) 780

Objašnjenje:

Za prvi frejm moramo zapamtiti 20×20 = 400 piksela.

Iz svakog frejma u drugi menja se 38 piksela, kao što je prikazano u nastavku.



Nakon prvog frejma postoji 10 novih frejmova, tako da ukupno moramo da sačuvamo 400+ (38×10) = 780 piksela.

Informatička pozadina

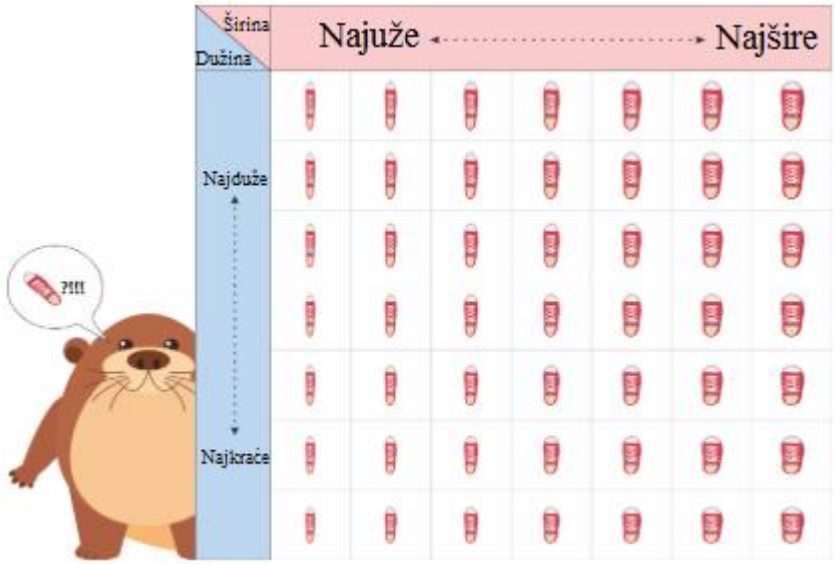
Kompresija podataka je važna tema iz informatike, posebno za video i audio podatke. Neki od poznatijih formata poput JPEG gube podatke tokom kompresije, pa se neke boje ili granice iskrivljuju. Ovaj zadatak opisuje oblik kompresije bez gubitaka, gde se ne gube nikakve informacije o slici. Ako snimamo frejmove jedan za drugim, o svakom frejmu možemo razmišljati kao o delu trodimenzionalnog niza. Promena na ekranu se dešava samo u onim pikselima koji se razlikuju u susednim frejmovima.

Saznajte više na: ttps://en.vikipedia.org/viki/Data_compression#Video

Treba imati na umu da u predstavljenom algoritmu, zapravo moramo zapamtiti ne samo boju promenjenog piksela, već i njihove koordinate.

Kupovina cipela

Dabar Vojin je otišao u prodavnicu obuće da kupi cipele. Postoji sedam različitih širina i sedam različitih dužina cipela, poređanih kao što je prikazano na slici.



Cipele su poređane prema dužini, kao i prema širini. Najkraća i najuža cipela je u levom donjem uglu, a najduža i najšira cipela u gornjem desnom uglu. Sve cipele su različitih dužina i širina.

Budući da naš dabar često zaboravlja, on se ne seća veličine obuće i moraće da isproba cipele dok ne nađe odgovarajuće.

Kada dabar proba neku cipelu, on tačno zna da li mu ta cipela odgovara po širini i da li mu odgovara po dužini.

Pitanje/Izazov

Koliko najmanje cipela dabar mora da proba da bi sa sigurnošću pronašao odgovarajuću?

Ponuđeni odgovori:

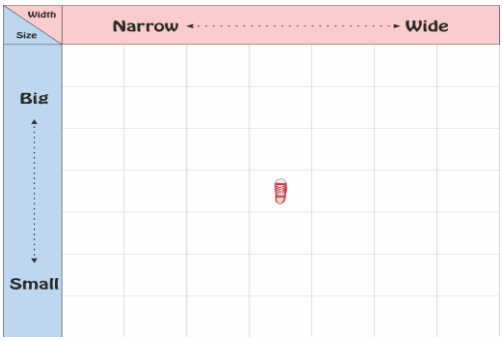
A) 2
B) 3
C) 4
D) 5

Tačan odgovor je:

Tačan odovor je A) 2.

Dabar može imati sreće i da u prvom pokušaju pronađe cipelu. Međutim, on može biti siguran da će naći odgovarajući model nakon što isproba dve cipele.

- Može početi s cipelama u sredini kao što je prikazano ispod.



- Cipela će biti odgovarajuće veličine, manje, veće i prave širine, uska ili široka za dabrova. Prema poziciji, dabar će znati da će cipela koja će mu odgovarati biti u jednoj od sledećih devet obojenih zona.

- Ako mu cipela odgovara, pronašao je pravu cipelu
- Ako je cipela manja i šira, isprobao bi cipele u zoni 1
- Ako je cipela manja, ali prave širine, isprobao bi cipele u zoni 2
- Ako je cipela veća i uska, isprobao bi cipele u zoni 9 i tako dalje

Pretpostavimo da je cipela koju je dabar probao za njega manja i šira. Tako da mora da proba veće i uže cipele, koje su u zoni 1.
Sada pokušava obuću u centru zone 1 (kutija označena kao # 1).
- Ako mu cipela odgovara, pronašao je pravu cipelu
- Ako je cipela i dalje manja i šira, cipela na poziciji A će biti odgovarajuća.

- Ako je cipela manja, ali prave širine, cipela na poziciji B će biti odgovarajuća.

Kao što vidimo, dabar će morati da isproba najviše 2 cipele kako bi pronašao odgovarajuću.

Ako krene od bilo koje druge pozicije, mora će isprobati mnogo više cipela.

Informatička pozadina

U gornjem problemu, cipele koje su izložene u prodavnici obuće raspoređene su u sve većem redosledu i širini duž više dimenzija. Takav aranžman se naziva sortiranje. Algoritam binarnog pretraživanja koristi se na sortiranim podacima da bi se vrlo brzo pronašao predmet. Binarno pretraživanje svaki put smanjuje prostor za pretragu na pola da biste dobili tačan odgovor u manjem broju pokušaja.

Korišćenje algoritma binarne pretrage prilikom igranja igre da pogodi broj između recimo 1 i 100 i dovede do nagađanja broja u minimalnom broju pokušaja.

Ovaj zadatak se rešava tehnikom binarne pretrage u dvodimenzionom nizu.

Quipu

Princeza Minja koristi čvorove na visećim kanapima (zvanim **Quipu**) kako bi objavila vesti svom kraljevstvu. Svaki kanap može imati **0** , **1** , **2** ili **3** čvora.

Važan je samo redosled kanapa i broj čvorova na svakom kanapu.

Princeza ima na raspolaganju samo **50 različitih objava**.

Pitanje/Izazov

Koji je najmanji broj kanapa potreban princezi da bi mogla da objavi 50 različitih objava?

Ponuđeni odgovori:

A) 2
B) 3
C) 4
D) 5

Tačan odgovor je:

Tačan odgovor je **B).**

Objašnjenje:

Ako postoji samo jedan kanap, onda su moguće 4 različite objave jer taj kanap ima 0, 1, 2 ili 3 čvora. Povećamo li na dva kanapa dobijamo 4 mogućnosti za svaki kanap i tako 4x4 = 16 različitih objava sve zajedno. To još uvek nije dovoljno. Međutim, kad se doda treći kanap, moguće su 4x4x4 = 64 različite objave. Budući da je 64 veće od 50, to je dovoljno za prikaz svih objava.

Informatička pozadina

Ovo je problem koji zahteva da razmišljate o notaciji položaja. Na quipuu je važno mesto kanapa i broj čvorova na kanapu. Budući da postoje četiri mogućnosti za broj čvorova na kanapu, quipu je ono što nazivamo kvartarni sistem.

Kad se radi sa pozitivnim celim brojevima, obično koristimo decimalni sistem. Decimalne cifre (0 do 10) nalik su čvorovima na quipu-u, a položaj cifara koji odgovara potencijama baze 10 sličan je položaju kanapa u quipu-u. Na primjer, 427 = 4x100 + 2x10 + 7, a s tri decimalne cifre možemo sačuvati do 10x10x10 = 1000 različitih pozitivnih brojeva (0 do 999).

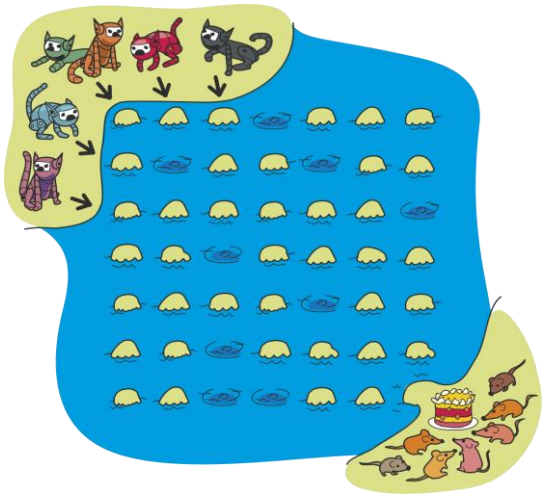
Računari koriste binarni sistem. Odnosno, postoje samo dve mogućnosti za svaku cifru (0 ili 1 nazvani bitovima). Svi podaci i instrukcije računara prikazani su pomoću binarnog sistema.

Žurka

Šest mačaka kreću na žurku sa miševima. One se nalaze na jednom, a miševi na drugom velikom ostrvu, između kojih su mala ostrva. Mačke jedva čekaju žurku i žele da što pre stignu.

One se kreću tako što skaču sa ostrva na ostrvo. Mačke mogu da skaču samo na susedno ostrvo, ne mogu da skaču dijagonalno i ne mogu da skaču na ili preko ostrva koja su pod vodom. Mačka može da skače vrlo brzo i možemo da pretpostavimo da je vreme potrebno za skok nula. Nakon skoka, mačka je veoma iscrpljena i treba joj odmor od jednog minuta. Zatim, skače na sledeće ostrvo, ukoliko sledeće ostrvo nije zauzeto, inače, mačka ostaje na ostrvu na kom se nalazi, još minut.

Mačka može da skoči na ostrvo samo ako je to ostrvo slobodno ili ako mačka, koja je na tom ostrvu, u tom trenutku odlazi. Mačke mogu da krenu sa bilo kog ostrva obeleženog strelicama.



Pitanje/Izazov

Koliko minuta treba svim mačkama da pređu od početnog mesta do odredišta (u najbržem slučaju)?

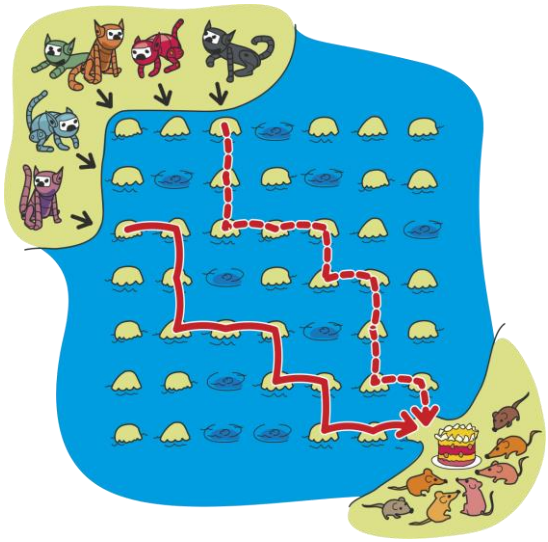
Ponuđeni odgovori:

A) 12
B) 13
C) 14
D) 15

Tačan odgovor je:

Tačan odgovor je 12.

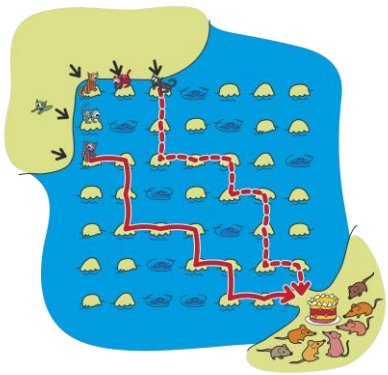
Najpre bi trebalo da pronađemo najkraće staze od svakog mogućeg ulaza: to su 10 minuta, 11 minuta i 12 minuta. Tada je važno napomenuti da postoje tačno dve različite najkraće staze (za kretanje u dva paralelna toka), kao što je prikazano na slici.

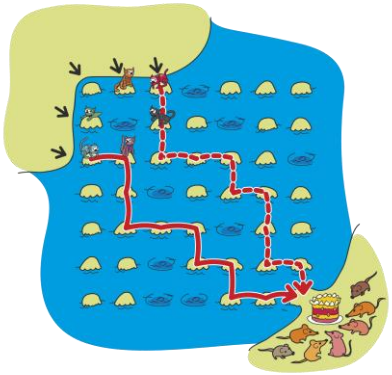


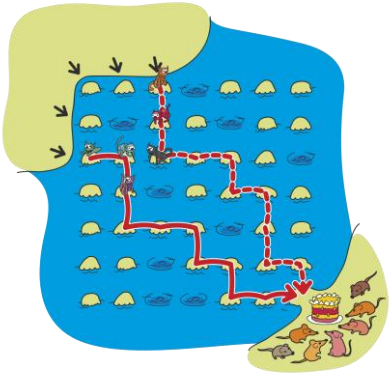
Dakle, 5 mačaka može uspeti da stigne za 12 minuta. Poslednja 6. mačka može da krene kada i peta mačka skokom na jedno od ulaznih ostrvca blizu ugla.

Potrebno je 10 minuta da mačka putuje i stigne na ostrvo za zabavu. Ako postoje dve mačke, vreme je još 10 minuta. Za tri mačke, sve ne mogu paralelno da vode najkraće staze, jer postoje samo dve najkraće staze. Dakle, treća mačka mora da putuje posle prve ili druge mačke cevovodom i to će trajati 11 minuta. Slično tome, za četiri mačke će trebati isto toliko vremena kao i za tri mačke. Logično je da će pet ili šest mačaka zahtevati 12 minuta u istom problemu.

Prvi korak	Drugi korak	Treći korak







Dalje objašnjenje:

Zašto to ne mogu brže?

Svaka mačka treba da prođe prvo kroz ulazak ili preko poslednjeg ulaza. Zatim, ako mačka krene sa nekog drugog ostrva i prođe kroz jedno od ova dva, možemo smatrati da je ova mačka tek krenula odavde i zaboravila sve prethodne delove staze. Dakle, sada smo ovaj zadatak sveli na samo dva ulaza.

Za svaki od ova dva ulaza, očigledno je najkraći put za miševe u trajanju od 10 minuta, prikazan na slici. Dakle, mačke možemo podeliti u dva toka. Kako bi u jednom potoku morale biti najmanje 3 mačke, za drugu i treću mačku moramo dodati najmanje 2 minute u svaki potok, tako da nam treba najmanje 12 minuta.

Ako tri mačke kreću s jednog od drugog našeg ulaza jedna za drugom, a tri druge mačke kreću od drugog ulaza na isti način, imamo tačno 12 minuta.

Informatička pozadina

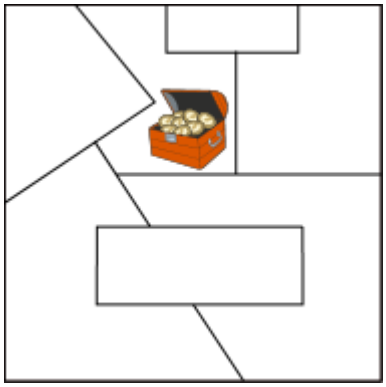
Kompjuterski stručnjaci su specijalisti za modeliranje i evaluaciju situacija, poput situacija zasnovanih na grafovima. Jedan od glavnih zadataka je otkrivanje kakvog se ponašanja može pokazati matematički modeliranim sistemima i koliko dugo može da traje matematički modeliran proces. Problem optimizacije je problem pronalaženja najboljeg rešenja iz svih izvedivih rešenja.

U računarskoj nauci važno je pretvoriti situacije iz stvarnog sveta u strukture podataka koje računari mogu razumeti. Lokacija ili položaj podataka često su predstavljeni u mrežnom prostoru. Moramo pronaći najkraću stazu dok putujemo samo vertikalno i horizontalno u mreži, a pritom izbegavamo ostrva pod vodom.

Štaviše, ovaj zadatak bi se mogao smatrati problemom teorije grafova. Ostrve skupa (uključujući ulaze i zabavu za miševe) možemo predstaviti kao vrhove grafa, sa susednim vrhovima koji odgovaraju susednim ostrvima. U takvom slučaju bismo samo ignorisali ostrva pod vodom. Dakle, naš zadatak je povezan sa problemom maksimalnog protoka: možemo primetiti da postoji „presečeni set“ dva ostrva, tako da ne može više od dve mačke da prođe kroz naš grafikon istovremeno i potrebno nam je više vremena da pustimo preostale mačke prolaze.

Mapa

Kralj dabrova je sakrio svoje blago u zemlji koju čini 7 pokrajina, kao što je prikazano na sledećoj mapi.

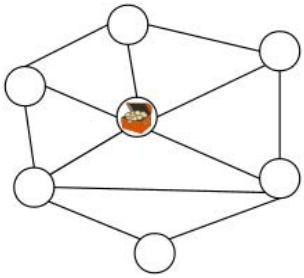


Kralj je napravio mapu. Krugovi označavaju pokrajine, a dva kruga su povezana linijom ako se odgovarajuće pokrajine graniče jedna sa drugom. Kako bi zbunio lopove, kralj je napravio još tri lažne mape.

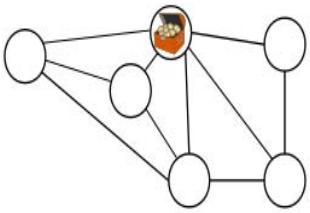
Pitanje/Izazov

Koja mapa je prava?

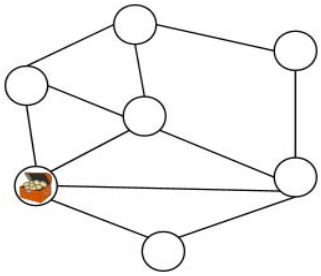
A



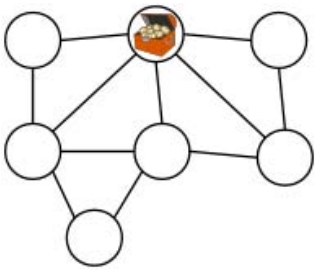
B



C



D



Tačan odgovor je:

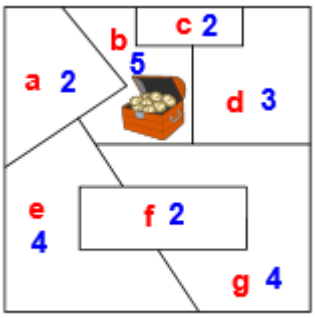
D

Mapa prikazuje 7 pokrajina. Zato B ne može biti tačan odgovor, jer sadrži samo 6 krugova.

Pokrajina u kojoj se nalazi blago ima pet suseda. Zato C ne može biti tačan odgovor, jer na mapi ne postoji pokrajina koja ima 5 suseda.

Za izbor između A i D, moramo uskladiti krugove (koji označavaju pokrajine na kodiranoj mapi) sa područjima na originalnoj mapii. Broj suseda nam može pomoći.

Obeležićemo pokrajine slovima **a** do **g** (crvena slova na slici ispod) i prebrojati njihove susede (plavi brojevi na slici ispod):



Pokrajina **a** ima 2 suseda (b i e).

Pokrajina **b** ima 5 suseda (a, c, d, e i g).

Pokrajina **c** ima 2 suseda (b i d).

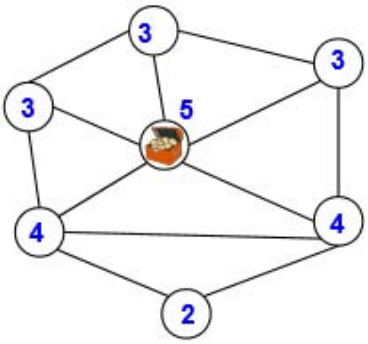
Pokrajina **d** ima 3 suseda (b, c i g).

Pokrajina **e** ima 4 suseda (a, b, f i g).

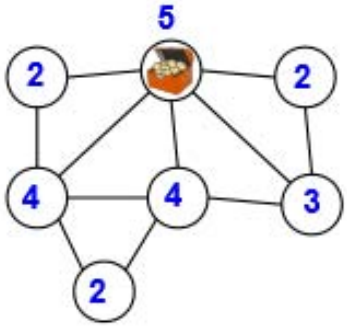
Pokrajina **f** ima 2 suseda (e i g).

Pokrajina **g** ima 4 suseda (b, d, e i f).

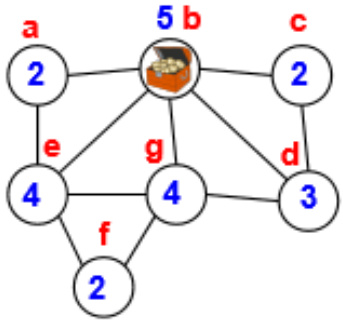
Ukupno, imamo 3 pokrajine sa 2 suseda, 1 pokrajinu sa 3 suseda, 2 pokrajine sa 4 suseda i 1 pokrajinu sa 5 suseda. Prebrojimo li susede na kodiranoj karti pod A), uočavamo 3 pokrajine sa 3 suseda i samo jednu pokrajinu sa 2 suseda. Zato A) ne može biti tačan odgovor.



Prebrojimo li broj suseda na karti D, uočavamo da odgovara našim uslovima: tri pokrajine imaju 2 suseda, jedna pokrajina ima 3 suseda, 2 pokrajine imaju 4 suseda i jedna pokrajina ima 5 suseda.



Rešenje smo pronašli eliminisanjem ostalih koji ne zadovoljavaju uslove. No da budemo sigurni ipak ćemo pridružiti krugove pokrajinama. Zapravo je moguće i ovako:



Informatička pozadina

Ovaj zadatak je primer kako grafove možemo koristiti za prikaz stvarnih situacija. Graf ima vrhove (krugove) i ivice (linije) koji ih spajaju. Linije označavaju veze između čvorova.

U ovom slučaju krugovi označavaju pokrajine, a linije označavaju vezu „je sused od“. Grafovi su odličan alat kada treba jasno opisati veze između nekih entiteta (pokrajine u ovom slučaju) u stvarnom svetu. Matematičari i informatičari razvili su više korisnih algoritama koji se mogu primeniti na grafove (jednostavni primer u ovom zadatku je brojanje krugova).

Saznajte više grafovima na primer ovde: https://en.wikipedia.org/wiki/Graph_(discrete_mathematics)

Presipanje vode

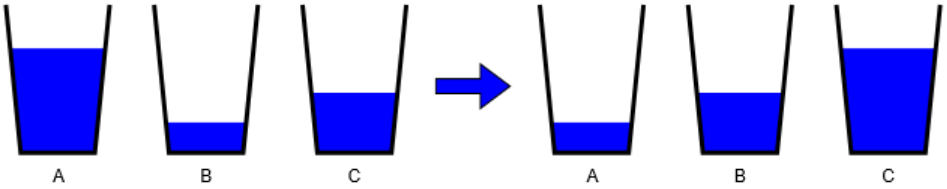
Tri čaše A, B i C sadrže različite količine vode. Nijedna čaša nije na početku puna. Na čašama nema oznaka za merenje količine vode. Međutim, moguće je uporediti količinu vode u njima. Dozvoljene su samo sledeće operacije (jedna ili više njih) koje su navedene ispod (imajte na umu da nije moguće uvek sve operacije izvršiti):

„Pražnjenje“: uzmite čašu i sav njen sadržaj prelijte u drugu čašu;

„Izjednačavanje“: uzmite čašu i sipajte onoliko vode u drugu čašu, tako da druga čaša sadrži tačno onoliko vode koliko i treća čaša;

„Punjenje“: uzmite čašu i izlijte što više vode u drugu čašu, tako da je druga čaša puna.

Zadatak je primeniti jednu ili više gore opisanih dozvoljenih operacija kako bi se promenila količina vode u svakoj čaši, kao što je prikazano na donjoj slici, bez upotrebe dodatnih čaša.



Na kraju, čaša A treba da sadrži količinu vode koja je na početku bila u čaši B, čaša B treba da sadrži količinu vode koja je na početku bila u čaši C, a čaša C treba da sadrži količinu vode koja je na početku bila u čaši A.

Pitanje/Izazov

Koja je od sledećih izjava tačna?

Ponuđeni odgovori:

A) Željeni ishod se može postići bez operacije „pražnjenja“.
B) Željeni ishod se može postići bez operacije „izjednačavanja“.
C) Željeni ishod se može postići bez operacije „punjenja“.
D) Željeni ishod se ne može postići pomoću ove 3 operacije.

Tačan odgovor:

Tačan odgovor je D. Imajte na umu da nakon izvršenja bilo kojeg pozitivnog broja tri navedene dozvoljene operacije, jedno od sledećeg se dešava:

- Jedna od čaša je prazna;
- Dve čaše imaju istu količinu vode;
- Jedna čaša je puna.

Traženo konačno stanje nema nijedno od ovih svojstava, pa je transformacija nemoguća.

Informatička pozadina

Tri operacije u zadatku su primeri operacija uništavanja informacija. Jedan od načina da se utvrdi da li su informacije sačuvane nakon operacije je može li se operacija preokrenuti. Početno stanje može se predstaviti kao tri različita broja veća od 0 i manja od 1. Ako izvršite operaciju pražnjenja, jedan od tih brojeva pretvara se u nulu. Ako izvodite operaciju izjednačavanja, dva broja će se pretvoriti u iste. Ako izvršite operaciju punjenja, jedan od brojeva pretvara se u fiksnu vrednost 1. U svakom je slučaju nemoguće preokrenuti rad da biste preuzeli početne brojeve.

Kompjuterski programeri moraju pažljivo razmisliti o efektima operacija u svojim programima kako bi bili sigurni da su sačuvane važne informacije. Dobro poznat primer gde se informacije moraju sačuvati u programiranju je zamena vrednosti dve promenljive. Pretpostavimo da promenljiva a sadrži vrednost 0,5, a promenljiva b sadrži vrednost 0,2, a naš cilj je da zamenimo vrednosti. Naivan i pogrešan algoritam je:

a: = b;

b: = a;

Nakon toga i a i b sadrže vrednost 0,2, a početna vrijednost 0,5 promenljive a se gubi. Programer može uvesti promenljivu c da bi sačuvao početnu vrednost a:

c: = b;

b: = a;

a: = c;

Alternativno, programer može algoritam prepisati na sledeći način:

a: = a - b;

b: = b + a;

a: = b - a;

U ovom slučaju se razlika između početnih vrednosti se čuva i to se može iskoristiti za ponovno izračunavanje početnih vrednosti u drugoj i trećoj programskoj liniji.

U zadatku svaka od dozvoljenih operacija uzrokuje gubitak informacija. Pored toga, dodatne promenljive su isključene nepostojanjem dodatnih čaša. Kvantno računanje je još jedno područje računarske nauke gde su operacije očuvanja informacija važne. U kvantnom računanju sve operacije moraju biti reverzibilne da bi se smatrale validnim operacijama

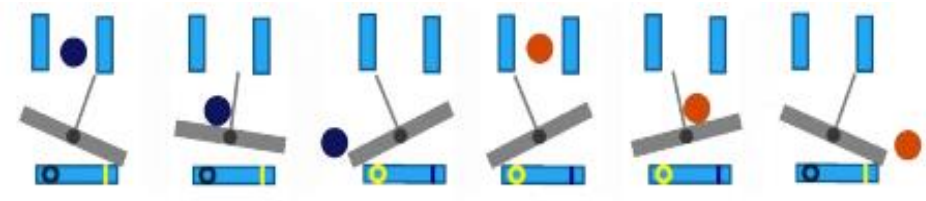
Brojač

Mašina na slici sadrži **4 poluge** koje se mogu ljuljati.

Poluga nagnuta ulevo = **0** ;

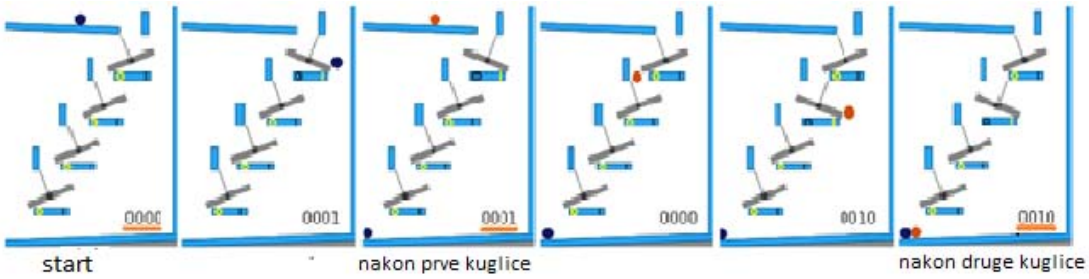
Poluga nagnuta udesno = **1** .

Poluga, na koju padne kuglica, se naginje u suprotnu stranu i kuglica se otkotrlja dalje.



Pogledajte prikaz mašine nakon ispuštanja prve **2** kuglice.

Na prvoj slici su sve poluge u položaju **0** , a brojač to prikazuje kao **0000** .



Pitanje/Izazov

Šta prikazuje brojač nakon prolaska 5. kuglice?

Ponuđeni odgovori:

A) 0101

B) 0100

C) 1010

D) 0011

Tačan odgovor:

A) 0101

Objašnjenje:

Kompletni prikaz je: 0001, 0010, 0011, 0100, 0101.
Ovo je mehanički binarni brojač.
Početna pozicija je 0000.
Prva kuglica naginje 1. polugu udesno. Brojač prikazuje 0001.
Druga kuglica naginje 1. polugu ulevo i naginje 2. polugu udesno. Brojač prikazuje 0010.
Treća kuglica naginje 1. polugu udesno i otkotrlja se van. 2. poluga ostaje nagnuta udesno. Brojač prikazuje 0011.
Četvrta kuglica naginje 1. polugu ulevo, 2. polugu ulevo i 3. polugu udesno. Brojač prikazuje 0100
Posljednja kuglica naginje 1. polugu udesno i otkotrlja se van. 2. poluga ostaje nagnuta ulevo, a 3. poluga ostaje nagnuta prema desno. Brojač prikazuje 0101.

To možemo posmatrati i drugačije:

Kuglica se kotrlja sa poluge na polugu sve dok ne padne na polugu koja se nalazi u položaju 0. Tada se poluga naginje prema udesno i loptica ispada van.

Informatička pozadina

Ako ste ograničeni na brojanje sa samo dve cifre, 0 i 1, koristite brojevni sistem sa bazom 2.

Taj se sistem se naziva binarni. Računari koriste binarni sistem jer svaki prekidač unutar njega može biti uključen (1) ili isključen (0). Današnji računari imaju puno prekidača koji upravljaju svim informacijama koje računar koristi.

Ova jednostavna mašina predstavlja mehanički binarni brojač. Ovaj brojač ima 4 cifre, pa može brojati do 15.

Ovde je tablica svih brojeva prikazanih sa 4 cifre:

Binarni broj	Broj
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	10
1011	11
1100	12
1101	13
1110	14
1111	15

Na linku pogledajte kako binarni brojač radi: Link