VISMA

Bebr[a]s '25

2025. gada 1. kārtas uzdevumi ar atbildēm

11.-12. klase

INFORMATĪVIE ATBALSTĪTĀJI

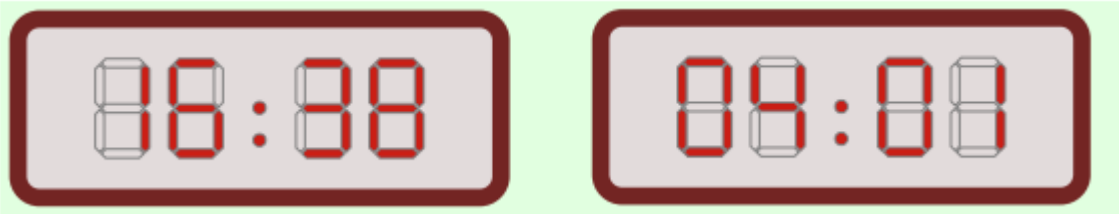
Izglītības un zinātnes ministrija

start(it)
www.startit.lv

Satura rādītājs

Pulksteņa segments

Čehija

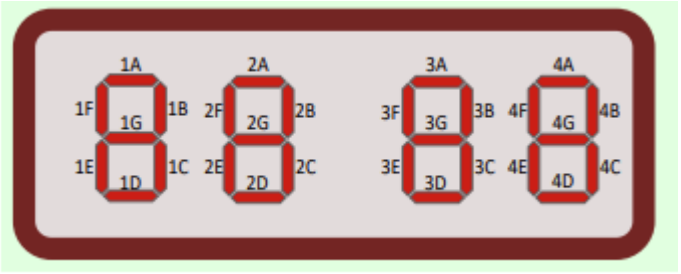


Standarta digitālais pulkstenis, kura rādījums mainās ik minūti, rāda laiku, izmantojot četrus ciparus. Katru ciparu attēlo septiņi gaismas segmenti. Lūk, kā, izmantojot segmentus, var attēlot skaitļus no 0 līdz 9:

Segmenti nolietojas katru reizi, kad tie tiek aktivizēti (pārslēgti no izslēgta stāvokļa uz ieslēgtu). Pēc kāda laika kā pirmais būs jānomaina segments, kas tiek aktivizēts visvairāk reižu.

Jautājums / Izaicinājums

Kurš no 28 pulksteņa segmentiem ir jānomaina vispirms? Norādiet segmenta kodu (cipars, kam seko burts, skatīt attēlu).



Interaktivitātes apraksts:

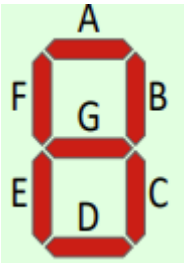
Kurš no 28 segmentiem ir jānomaina vispirms? Atzīmējiet to, noklikšķinot uz tā.

Katrs no 28 interaktīvajiem segmentiem tiek iezīmēts pēc noklikšķināšanas uz tā. Vienlaikus var atzīmēt tikai vienu segmentu.

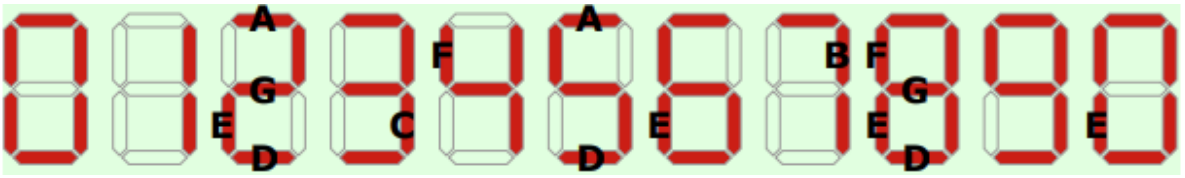
Atbildes skaidrojums

Pareizā atbilde ir: 4E (interaktīvā versija): 

Katrs cipars displejā apzīmē atšķirīgu laika periodu: desmitus stundu, stundas, desmitus minūšu un minūtes. Cipars visvairāk pa labi, t. i., ceturtais cipars jeb cipars ar kodu 4, rāda minūšu izmaiņas. Pirms jebkura cita cipara izmaiņām mainīsies visi ceturtā cipara segmenti. Visvairāk nolietotais segments būs ceturtajā ciparā, un tā segmenta kods ir E.



Mēs apskatīsim visus septiņus segmentus ceturtajā ciparā un noskaidrosim, cik reižu tie tiks aktivizēti (ieslēgti), pirms tajā tiks nomainīti visi cipari no 0 līdz 0.



Attēlā redzama šāda pārbaude. Aktivizētais (tikko ieslēgtais) segments ir segments, kas iedegas uz pašreizējā cipara, bet nedega uz tuvākā cipara pa kreisi. Katra segmenta aktivizācija ir atzīmēta ar šī segmenta burtu. Aktivizāciju skaits ir burtu skaits visā attēlā. Tabulā ir apkopota šī pārbaude. Centrālajā kolonnā ir parādīti cipari, uz kuriem segments tika aktivizēts. Labajā kolonnā ir uzskaitīti šie cipari.

Segments	Cipars, kas izraisa segmenta aktivizēšanu	Aktivizēšanas reižu skaits
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

Visvairāk mainītais segments ir E uz ceturtā cipara, un tā kods ir 4E.

Šī ir informātika

Šajā uzdevumā laika attēlošanai tiek izmantots septiņu segmentu displejs. Septiņu segmentu displejs ir izplatīts tehnoloģiskajos produktos. Tas attēlo skaitļus no 0 līdz 9, izmantojot septiņas gaismas diodes. Turklāt šis uzdevums ietver skaitīšanas sistēmas koncepciju. Skaitīšanas sistēmām ir liela nozīme datorzinātnēs. Visbiežāk izmantotā skaitīšanas sistēma ir decimālsistēma, savukārt datori datu attēlošanai izmanto bināro sistēmu. Tomēr mūsu ikdienas dzīvē pastāv arī citas skaitīšanas sistēmas. Piemēram, standarta pulkstenī minūtes tiek attēlotas sešdesmitnieku (ar bāzi 60) sistēmā, bet stundas - divdesmitčetrinieku (ar bāzi 24) sistēmā.

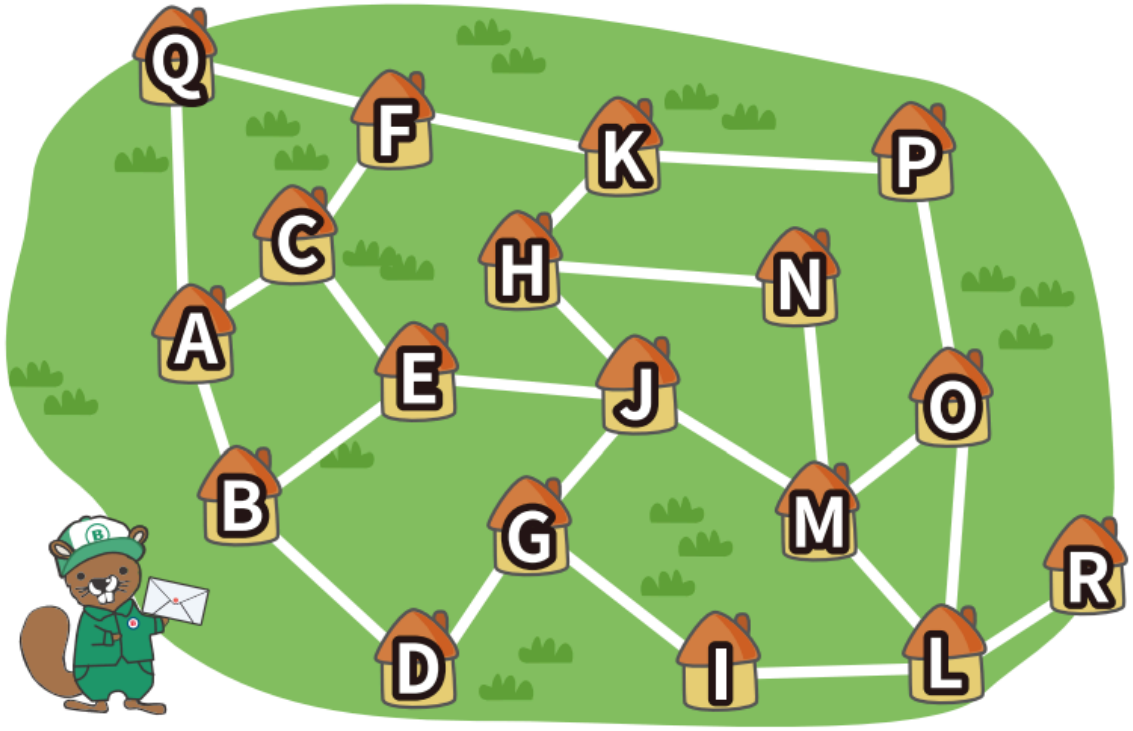
Šī ir skaitļošanas domāšana

Šis uzdevums paģēr, lai skolēni izmantotu loģisko spriešanu un veselo saprātu, lai ieraudzītu, ka standarta digitālā pulksteņa pēdējais cipars mainās visbiežāk. Tāpēc uzdevumu var sašaurināt līdz segmenta izmaiņu analīzei labajā malā esošajā ciparā. Pēc tam skolēniem jāanalizē izmaiņu skaits katrā septiņu segmentu displeja segmentā pilna izmaiņu cikla laikā. Šādas izmaiņas tiek attēlotas kā displeja segmentu pāreja no izslēgta stāvokļa uz ieslēgtu stāvokli. Šim procesam ir nepieciešama spēja atpazīt modeļus.

Bebrusalas biļetens

Taiwāna

Bebrusalā ir 18 ciemati, kā parādīts attēlā zemāk. Katrā ciematā ir daudz pastnieku - ikreiz, kad no ciemata ir jānosūta ziņojums vai tajā tiek saņemts jauns ziņojums, ciemata pastnieki nākamajā dienā nogādās ziņojumu visiem saistītajiem kaimiņu ciematiem.



Piemēram, ja no ciemata A nosūta ziņojumu, ziņojuma nonākšanai ciematos B, C un Q nepieciešama 1 diena. Lai ziņojums sasniegtu ciematus D, E un F, nepieciešamas 2 dienas, un tā tālāk, līdz visi ciemati ir saņēmuši ziņojumu.

Jautājums / Izaicinājums

Ja jauns ziņojums tiek sūtīts no ciemata J, cik dienas nepieciešamas, lai tas sasniegtu visus ciematus?

Atbilžu varianti / Interaktivitātes apraksts

Vesels skaitlis no [1, 18]

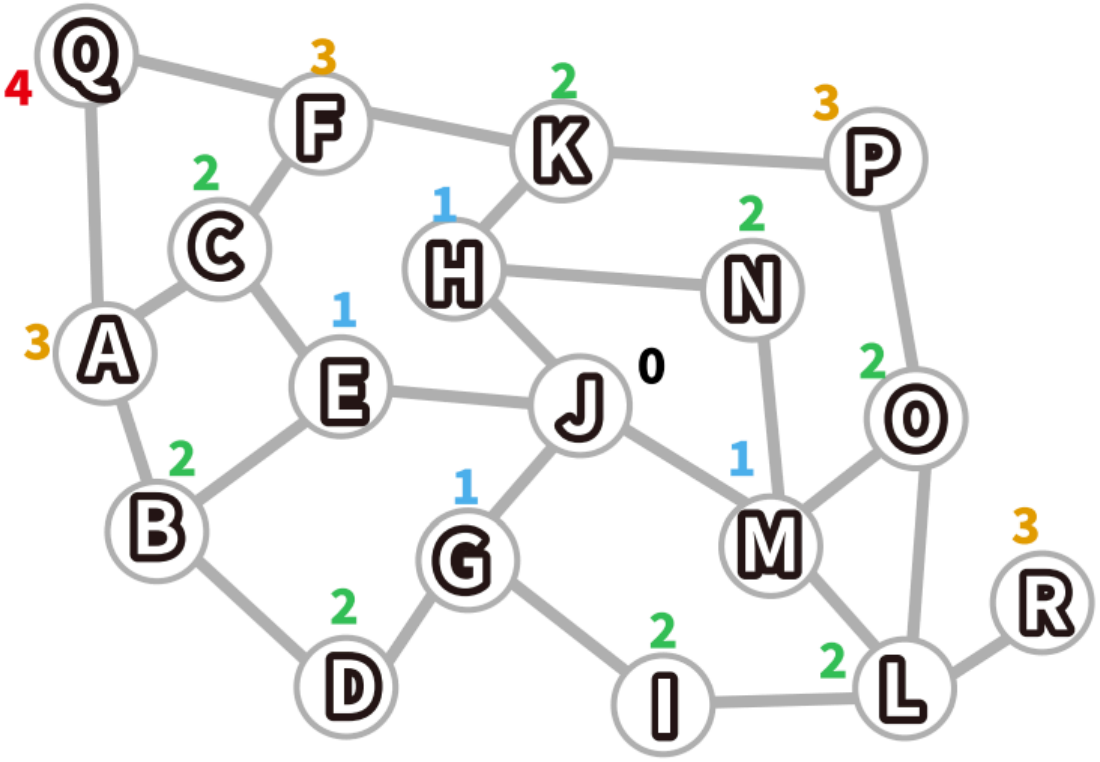
Atbildes skaidrojums

Sākot no ciemata J, mēs varam izsekot, cik dienu nepieciešams, lai ziņojums sasniegtu katru ciematu, un uzzināt atbildi:

1. diena: Ciemati E, G, H un M saņem ziņojumu.
2. diena: Ciemati B, C, D, I, K, N, O un L saņem ziņojumu, proti: ciemats B un C to saņem no E. Ciemati D un I to saņem no G. Ciemati K un N to saņem no H. Ciemati N, O un L to saņem no M.

3. diena: Līdzīgi ziņojumu saņem ciemati A, F, P un R. Šajā brīdī ziņojumu vēl nav saņēmis tikai ciemats Q.

4. diena: Ciemats Q saņem ziņojumu no F un A.



Tāpēc kopumā būs nepieciešamas četras dienas, lai nodrošinātu, ka visi ciemati ir saņēmuši ziņojumu no ciemata J.

Tā ir informātika

Šajā uzdevumā ziņojumu izplatīšanas metode ir līdzīga meklēšanas platumā (platkursija, BFS) algoritma veikšanai grafā, sistemātiski izpētot vai meklējot mezglus. Platkursijas algoritms sāk no konkrēta mezgla, vispirms apmeklējot tā blakus esošos mezglus, un pēc tam turpina apmeklēt vēl neapmeklētos blakus esošos mezglus tādā secībā, kādā tie tiek atrasti. Platkursijas pamatkoncepcija ir "paplašināšana slāni pa slānim", nodrošinot,

ka vispirms tiek izpētīti mezgli, kas atrodas tuvāk sākumam, pirms paplašināšanas uz attālākiem mezgliem.

Platkursijas praktiskos pielietojumos ietilpst meklētājprogrammas, kas pārmeklē tīmekļa lapas slāni pa slānim, izmantojot saites, sociālo mediju platformas, kas analizē, kā ziņojumi izplatās no viena lietotāja citiem, un sabiedrības veselības iestādes, kas izseko kontaktus un simulē infekcijas slimību izplatību. Saistīts jēdziens ir pludināšanas-aizpildīšanas algoritms, ko parasti izmanto, lai risinātu problēmas, kas saistītas ar savienotu apgabalu paplašināšanu vai izpēti. Piemēram, pieskaroties apgabalam krāsošanas lietotnē, algoritms aizpilda visu sadaļu ar krāsu. Līdzīgi tas tiek pielietots, kad tīrīšanas robots kartē un tīra visas savienotās telpas, sākot no sākotnējās atrašanās vietas. Iekšēji plūdu aizpildīšana parasti balstās uz meklēšanu platumā vai dziļumā (dziļkursija, DFS), lai sistemātiski un rūpīgi izpētītu visus blakus esošos savienotos mezglus.

Šī ir skaitļošanas domāšana

Lai atrisinātu šo uzdevumu, studentiem vispirms ir jāsaprot un jāpielieto noteikums par to, kā ziņojumi tiek nodoti no ciemata uz ciematu. Viņiem sistemātiski jāseko ziņojuma izplatībai dienu no dienas, kas ietver algoritmisku domāšanu. Šī procesa laikā studenti iesaistās arī simulācijā, jo viņi garīgi vai vizuāli modelē, kā ziņojums laika gaitā “plūst” tīklā. Viņiem ir jāatbrīvojas no nevajadzīgām detaļām un jākoncentrējas uz galvenajiem elementiem: savienojumiem starp ciematiem un ziņojuma izplatīšanas laiku.

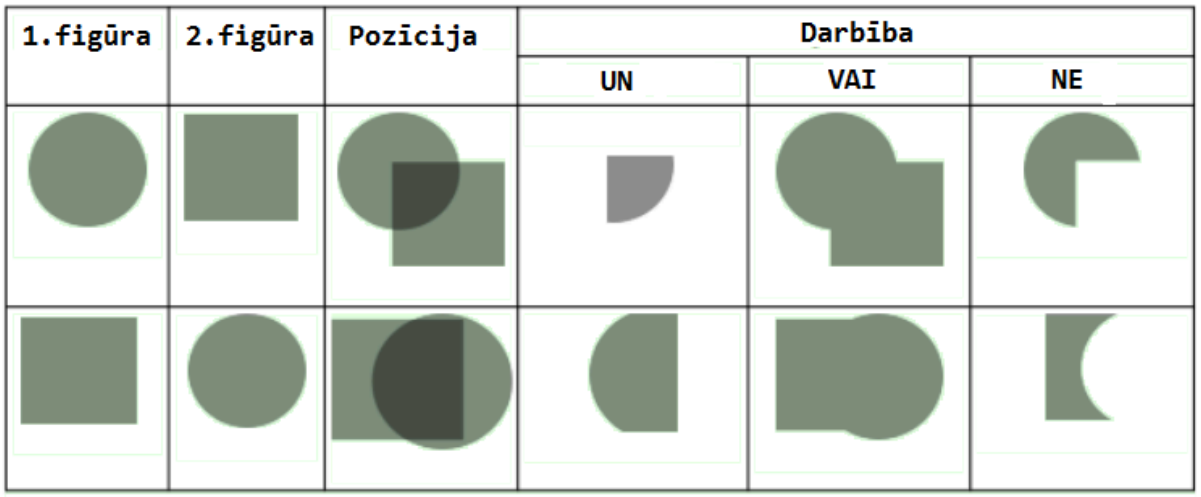
Informātikas atslēgvārdi un tīmekļa vietnes

Meklēšana plašumā, pludināšanas-aizpildīšanas algoritms

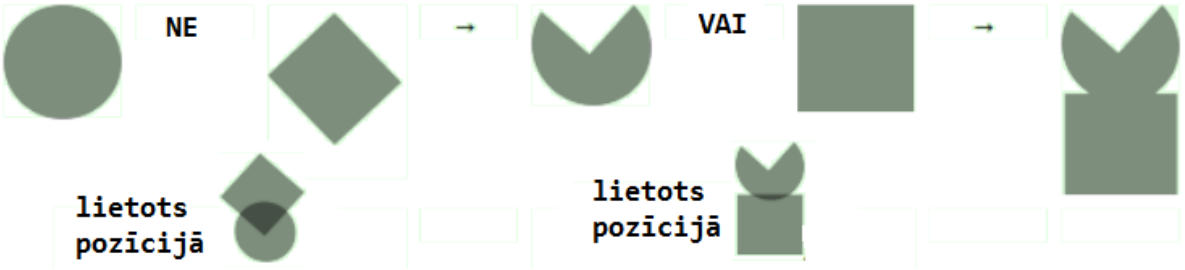
Būla figūras

Īrija

Būla darbības tiek izmantotas datorgrafikā. Šīs darbības ļauj no vienkāršākām figūrām veidot sarežģītākas. Zemāk ir sniegts trīs Būla pamatdarbību piemērs: UN (angliski AND), VAI (OR) un NE (NOT).

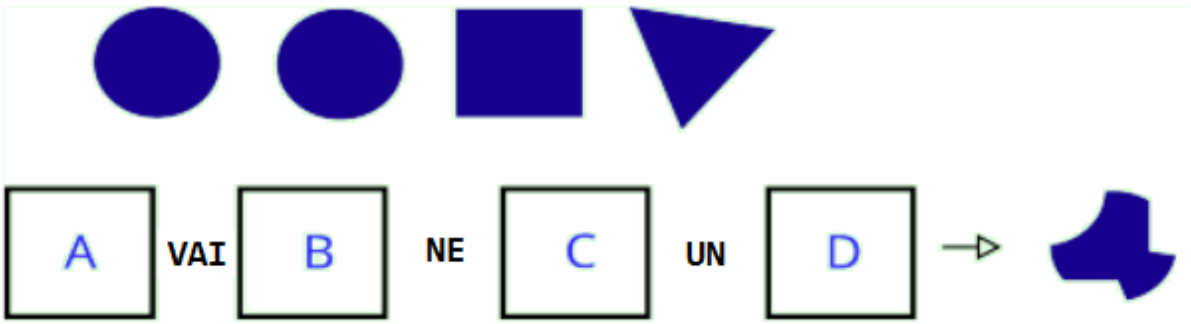


Mēs varam izmantot šīs darbības secīgi, lai uzzīmētu/izveidotu sarežģītākas figūras.

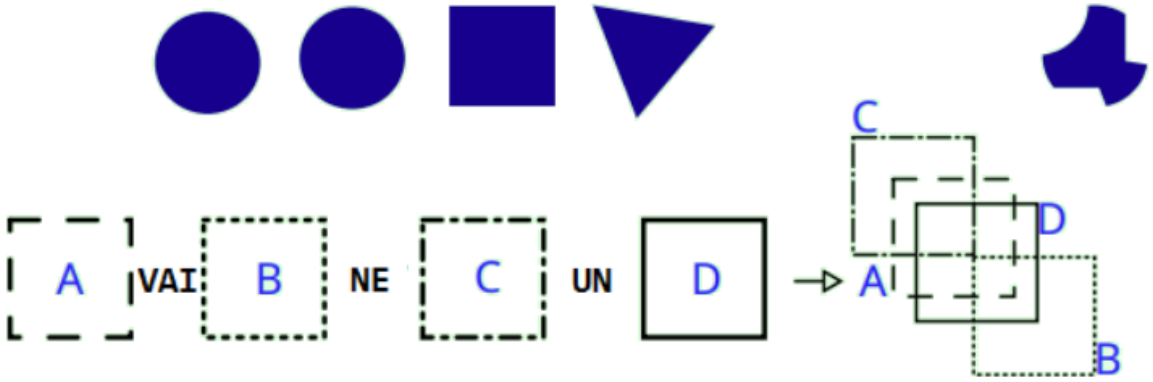


Jautājums / Izaicinājums

Jums ir dotas četras figūras un trīs Būla operācijas. Velciet un nometiet figūras pareizajās vietās tā, lai rezultāts pēc darbību izpildes būtu tāds, kāds parādīts aiz bultiņas.



Versija: Ja vēlaties sniegt skolēniem lielāku atbalstu pozīcijas noteikšanā, varat izmantot mērķa kvadrātus “pozīcija”:

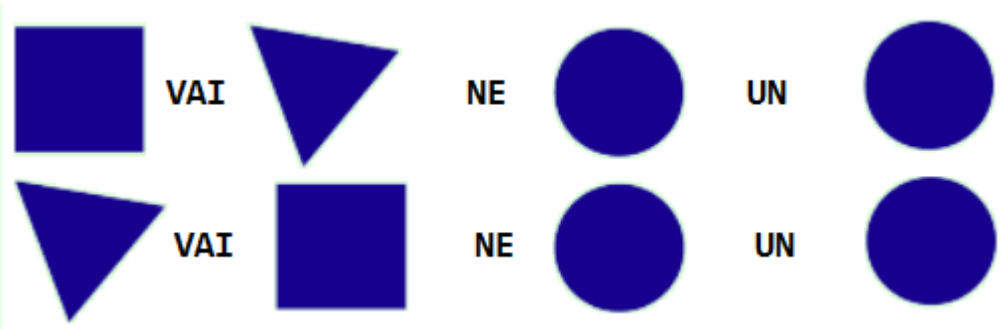


Atbilžu varianti / Interaktivitātes apraksts

Interaktivitātes apraksts: četras figūras jāvelk uz kvadrātiem, kas apzīmēti ar A, B, C, D. Ja tās atrodas tuvu atzīmētajiem kvadrātiem, tām vajadzētu “ielēkt” iekšā. Figūras var noņemt, atgriežot sākotnējā vietā (ja tās tiek pārvietotas ārpus atzīmētajiem kvadrātiem, tās atgriežas sākotnējā vietā).

Atbildes skaidrojums

Pareizā atbilde ir viena no šīm:



Lai atrisinātu šo uzdevumu, mums ir jāidentificē iegūtā attēla galvenās iezīmes.

Pirmā dotā darbība ir VAI, kas izveidos iegūtā attēla “ķermeni”, apvienojot divas figūras. Šeit jums jāizmanto kvadrāts un trīsstūris. Tā kā mēs izmantojam VAI, figūru secībai nav nozīmes. VAI rezultāts ir vienāds abos gadījumos.

Nākamā dotā darbība ir NE. Tas nozīmē, ka mēs noņemsim daļu no attēla. Mēs redzam, ka mums būs jāatņem kaut kas no mūsu figūras augšējā kreisā stūra. Tā kā tas ir apļveida, mums jāizmanto aplis, lai noņemtu daļu ar NE.

Pēdējā dotā operācija ir UN, kas saglabās visu, kas ir kopīgs abās figūrās. Rezultāta iegūšanai varam izmantot pēdējo apli.

Šī ir informātika

Būla darbības jau sen tiek izmantotas datorgrafikā. Tādas darbības kā VAI (divu figūru apvienojums), UN (divu figūru šķēlums) un NE (vienas figūras “atņemšana” no citas) ļauj no vienkāršākām figūrām izveidot sarežģītākas figūras. Šīs darbības parasti ir pieejamas gan bitkartes, gan vektoru zīmēšanas programmās. Šādas darbības, veiktspējas apsvērumu dēļ, labāk veikt ar vektoru attēlojumiem nekā ar bitkartēm. Tas, savukārt, ir novedis pie veselas algoritmu klases, slaucīšanas līnijas algoritmu, izmantošanas, lai efektīvi veiktu šīs darbības ar vektoru attēlojumiem. (IESPĒJA: Ja vēlaties sniegt skolēniem/skolotājiem vairāk infigūrācijas par terminu Būla loģika, tad tas attiecas uz ko tādu, kam var būt tikai viena no divām vērtībām: patiesa vai nepatiesa. Būla loģikai ir svarīga loma programmēšanā. Programmēšanā Būla apgalvojumi, piemēram, tiek izmantoti, lai izveidotu nosacījumus cikliem un zarošanās operatoriem: if punkti > 100: print(“Apsveicu”) Būla apgalvojums vienmēr ir patiess vai nepatiess: piemēram, “punkti > 100”, “Bobs ir vecāks par Alisi”, “Līst lietus”. Būla operatorus, piemēram, UN, VAI, NE, var izmantot, lai izveidotu sarežģītākus apgalvojumus vai izteiksmes: piemēram, “Līst lietus” UN “Māja ir dzeltena”. Šis apgalvojums ir patiess tikai tad, ja gan līst, gan māja ir dzeltena.

“Līst lietus” VAI “Māja ir dzeltena”. Šis apgalvojums ir patiess, ja līst lietus vai/un māja ir dzeltena.

NE “Māja ir dzeltena”. Šis apgalvojums ir patiess, ja māja NAV dzeltena.)

Šī ir skaitļošanas domāšana

Abstrakcija: Lai atrisinātu šo uzdevumu, skolēnam ir jāaplūko rezultātā iegūstamā figūra un jāatrod līdzības un atšķirības, lai izlemtu, kuras figūras izmantot un kādā secībā.
Sadalīšana: Mums ir jāsadala uzdevums mazākos, nosakot figūras, kas jāizmanto katrā atsevišķā darbībā.

Puķu stādīšana

Slovākija

Robots puķu dobē rindā stāda pilnībā izaugušas puķes, pamatojoties uz norādēm. Tukšā vietā nav ne norādes, ne puķes. Robots puķu stādīšanai izmanto šādas instrukcijas:

0: Doties uz vietu, kas atzīmēta ar X.

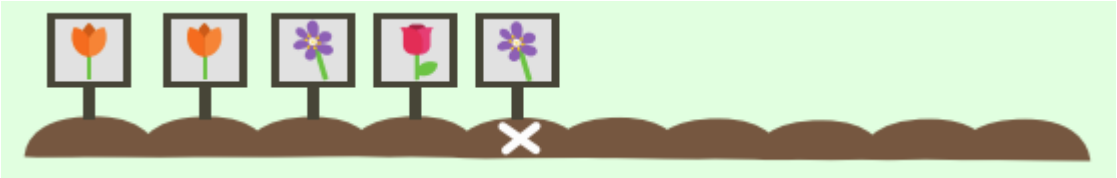
Atkārtot 1.–5. darbību, līdz vairs nav nevienas vietas ar norādi.

1: Ja šajā vietā ir norāde, iestādīt šajā vietā uz norādes attēloto puķi.

2: Iegaumēt tikko iestādīto puķi.

3: Noņemt norādi.

4: Dodieties pa labi, līdz sasniegta tukša vieta, pēc tam iestādīt tur puķi, ko iegaumējāt kā pēdējo.
5: Dodieties pa kreisi, līdz sasniegta vieta ar norādi vai līdz izkāpts ārpus puķu dobes.



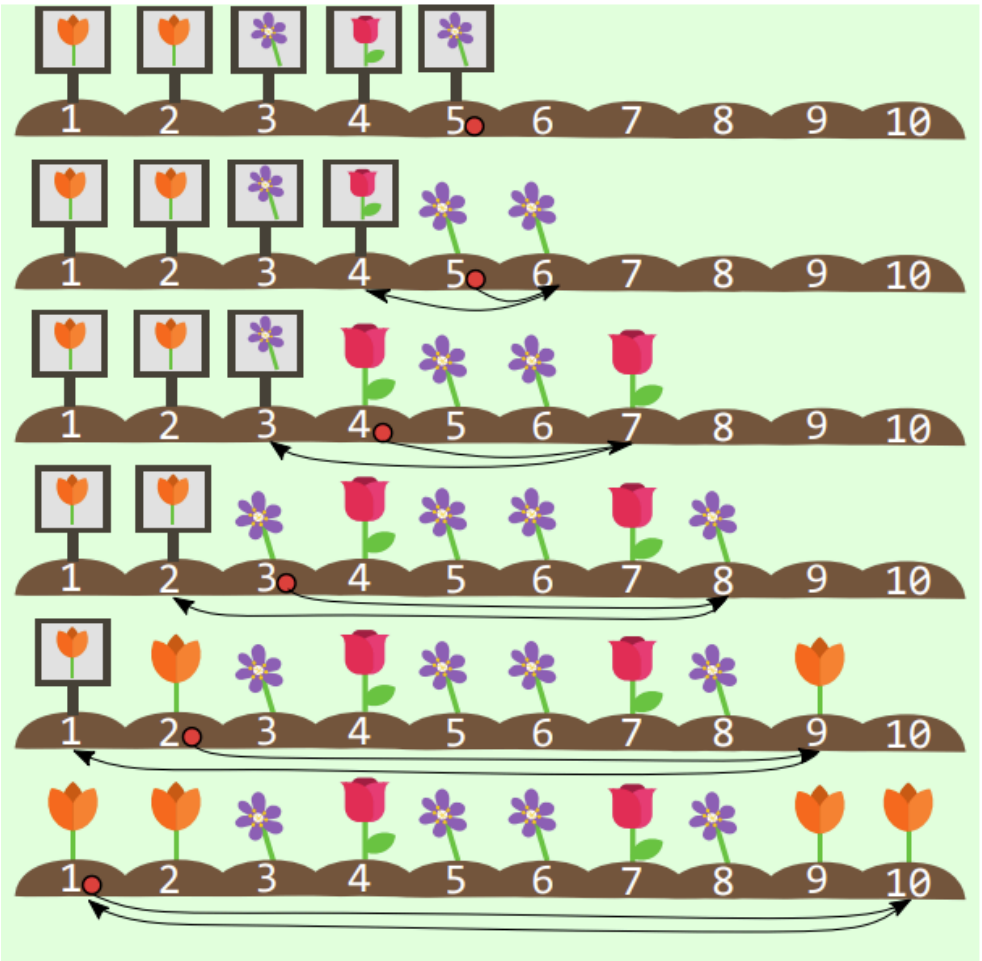
Jautājums / Izaicinājums

Kā puķu dobe izskatīsies pēc stādīšanas?

Atbilžu varianti / Interaktivitātes apraksts

Atbildes skaidrojums

Pareizā atbilde ir A.
Sanumurēsim puķu vietas. Robots sāk no pēdējās apzīmētās vietas, kas ir 5. vieta. Tur tas iestāda vijolīti, iegaumē to un iet pa labi. Tieši blakus ir tukša vieta (6. vieta), tāpēc robots iestāda pēdējo iegaumēto puķi (vijolīti) un tad iet pa kreisi, līdz atrod pirmo vietu ar norādi. Tā ir 4. vieta. Tur tas iestāda rozi un iegaumē to. Tad robots dodas pa labi uz pirmo brīvo vietu, kas ir 7. vieta, un iestāda rozi. Tālāk tas dodas pa kreisi, līdz atrod vietu ar norādi – tā ir 3. vieta. Robots tur iestāda vijolīti, iegaumē to un iet pa labi uz pirmo brīvo vietu (8. vieta), kur iestāda vēl vienu vijolīti. Gan 2. un 9. vietā, gan 1. un 10. vietā tas iestādīs tulpes. Var redzēt, ka robots tukšajās vietās stāda puķes pretējā secībā. Attēlā sarkanais punkts apzīmē robota pozīciju, un bultiņas parāda, kā tas pārvietojās.



Šī ir informātika

Šajā uzdevumā Tjūringa mašīnas koncepcija ir paslēpta aiz robota instrukcijām. Tjūringa mašīna ir mašīnas jēdziens, kas izpilda algoritmu un šajā uzdevumā tai ir viena galviņa (robots), kas pārvietojas pa lenti (puķu dobi) ar šūnām (punktiem) un var lasīt (norādes) un rakstīt (noņemt norādi un iestādīt puķes) šajos punktos. Tjūringa mašīnas atmiņa ir lente, un mašīnai var būt vairāk nekā viena lente. Šajā uzdevumā robots iegaumē, kura puķe tika iestādīta kā pēdējā, lai šo informāciju varētu ierakstīt citā lentē.

Šī ir skaitļošanas domāšana

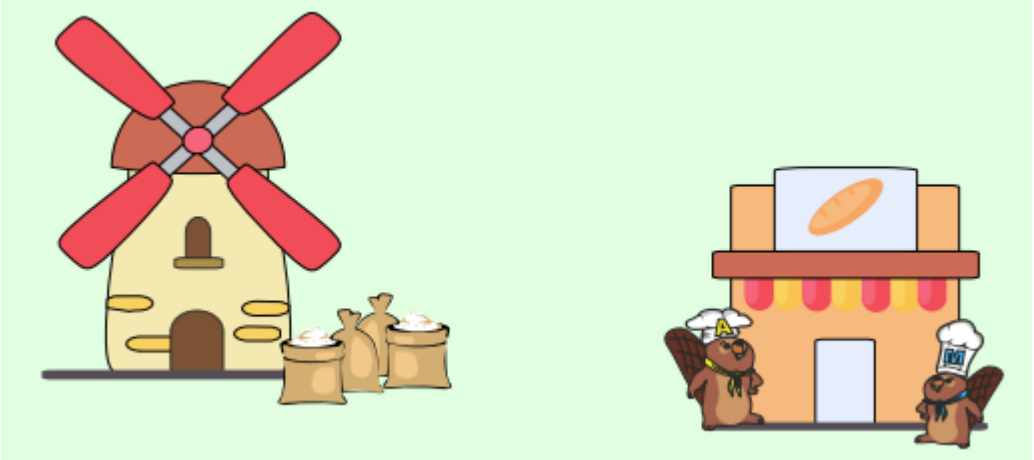
Šajā uzdevumā ir nepieciešama algoritmiskā domāšana. Jums ir jāsaprot un jāseko piedāvātajam algoritmam, kas ir uzrakstīts kā instrukciju kopums, un jums ir jāizpilda šīs instrukcijas noteiktiem ievaddatiem. Šajā uzdevumā ievaddati ir norāžu virkne, un rezultāts (izvaddati) ir iestādīto puķu virkne.

Milti

Taivāna

Divi bebri - Alberts un Mario veido leģendāru miltu pārvadāšanas komandu.
Alberts katrā braucienā pārvadā 13 kg miltu, un ceļš no maiznīcas līdz dzirnavām un atpakaļ aizņem vienu stundu. Mario pārvadā tikai 5 kg, bet katru turp un atpakaļ braucienu veic 30 minūtēs. Viņi abi nevar doties braucienā vienlaikus, jo vismaz vienam jāpaliek maiznīcā apkalpot klientus.

Tāpat kā visiem čakliem strādniekiem, viņiem kaut kad ir arī jāatpūšas. Pēc trim secīgiem braucieniem katram bebram jāpavada vismaz 30 minūtes, pirms viņš dodas nākamajā braucienā. Atpūtas laikā viņi var apkalpot klientus.



Jautājums / Izaicinājums

Alberts un Mario vēlas pārvest pēc iespējas vairāk miltu astoņu stundu laikā.
Kurš no dotajiem apgalvojumiem ir pareizs?

Atbilžu varianti / Interaktivitātes apraksts

A. Albertam jādodas braucienā pirmajam.

B. Mario jādodas braucienā pirmajam.

C. Mario jādodas pēdējā no braucieniem.

D. Alberts nedrīkst būt pēdējā no braucieniem.
E. Mario jādodas tikai vienā braucienā.

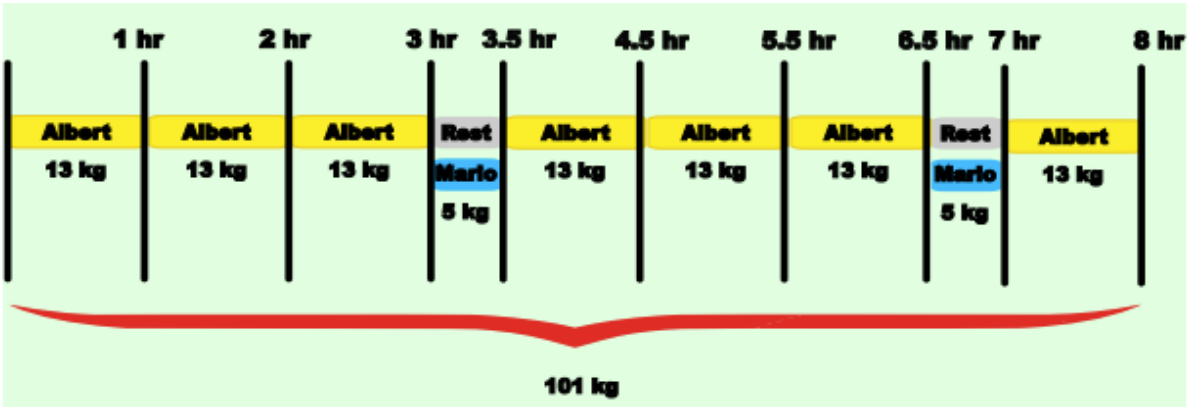
Atbildes skaidrojums

Pareizā atbilde ir A.
Galvenie fakti:
• Alberts: 13 kg vienā reizē, 1 stunda vienā reizē (13 kg/stundā)
• Mario: 5 kg vienā reizē, 30 minūtes vienā reizē (10 kg/stundā)

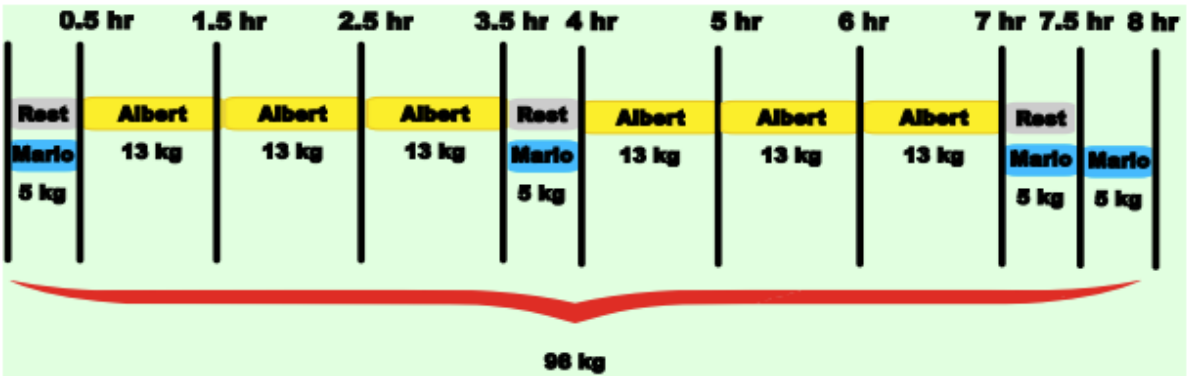
• Vienam bebram jābūt maiznīcā
• Katrs bebrs var izpildīt līdz trim braucieniem pēc kārtas
• Kamēr viens bebrs atpūšas maiznīcā, otrs turpina doties braucienos

Tā kā Alberts var piegādāt 13 kg/stundā, salīdzinot ar Mario 10 kg/stundā, Alberts ir efektīvāks. Tomēr mums ir jāizmanto Mario, kad Albertam ir jāatpūšas. Ja bebrs Alberts (A) pārvadā 3 reizes un Mario (M) vienu reizi, mēs iegūstam maksimālo miltu daudzumu 3,5 stundu laikā:

Tagad Alberts ir atpūties, tāpēc mēs varam atkārtot šo ciklu, līdz laiks ir beidzies:



Ja Mario sāk pirmais, tad pēc diviem (M, A, A) cikliem paliek tikai viena stunda. Tā kā Albertam tajā laikā ir jāatpūšas, Mario ir jāveic pēdējās divas kārtas. Tas noved pie mazāk efektīva grafika salīdzinājumā ar iepriekšējo.



Alberts nevar braukt 8 reizes, jo tas prasa 8 stundas braucienos + 2 * 30 minūšu atpūtu = 9 stundas, kas ir vairāk nekā atļauts uzdevumā. Tātad risinājums, kurā Alberts brauc 7 reizes + Mario 2 reizes, ir optimāls.

Šī ir informātika

Bebru miltu transportēšanas problēma ir plānošanas un optimizācijas problēma, jo:

• Mums ir jāplāno braucieni, ievērojot atpūtas un aizņemtības ierobežojumus.
• Mēs vēlamies maksimāli palielināt piegādāto miltu kopējo daudzumu (optimizācijas mērķis).
• Mums ir efektīvi jāsadala laiks starp Albertu un Mario.

Plānošanas un optimizācijas problēmas rodas daudzos informātikas uzdevumos, tostarp instrukciju izpildē datoru procesoros, kur aritmētiskās loģikas vienības var izpildīt tikai vienu instrukciju vienlaikus (aizņemtības ierobežojumi), instrukcijām, kas lasa datus no atmiņas, ir jāgaida, kamēr dati pienāk (līdzīgi kā atpūtai), un optimāls grafiks var izpildīt programmu daudz ātrāk nekā tad, ja grafiks ir neoptimāls.

Šī ir skaitļošanas domāšana

Šī uzdevuma risināšanā ir ieguvumi no sarežģītas problēmas sadalīšanas apakšproblēmās (dekompozīcija), atziņas, ka apakšproblēmas risinājumus var atkārtot (atkārtošana un modeļu atpazīšana), apakšproblēmas risinājumu sakārtošanas, lai atrastu optimālu risinājumu, un abstrakcijas, kas attēlo optimālo transportējamo kg/stundā. Ir nepieciešama arī loģiska domāšana, lai izslēgtu tos variantus, kas neatbilst tādiem ierobežojumiem kā atpūtas periodi un noslogojums (vismaz viens bebrs maizes ceptuvē).

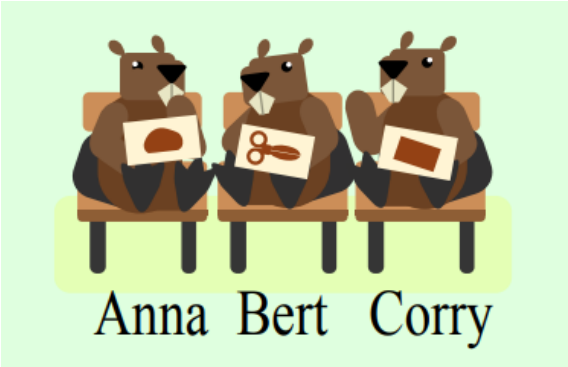
Akmens, šķēres, papīrīt’s un maiņas

Brazīlija

Anna (A), Berts (B) un Korija (C) spēlē populārās spēles “Akmens-šķēres-papīrīt’s” variantu. Atcerieties:

• akmens uzvar šķēres
• šķēres uzvar papīru
• papīrs uzvar akmeni

Anna, Berts un Korija apsēžas uz krēsliem un tur kārtis, lai visi tās varētu redzēt.



Tad spēlētājiem ir jāizlemj, cik kāršu maiņas viņi veiks. Maiņa ir kāršu apmaiņa starp diviem spēlētājiem. Pēc tam viņiem jāizlemj, kuri spēlētāji piedalīsies katrā maiņā.

Jautājums / Izaicinājums

Berta vienīgais mērķis ir pārspēt Koriju. Kāda stratēģija garantētu viņa panākumus?

Atbilžu varianti / Interaktivitātes apraksts

A) Bertam jāgarantē tikai nepāra skaits maiņu ar Koriju.

B) Neatkarīgi no tā, cik maiņas viņi nolemj veikt, Bertam nekad nevajadzētu apmainīties ar kārtīm ar Koriju.

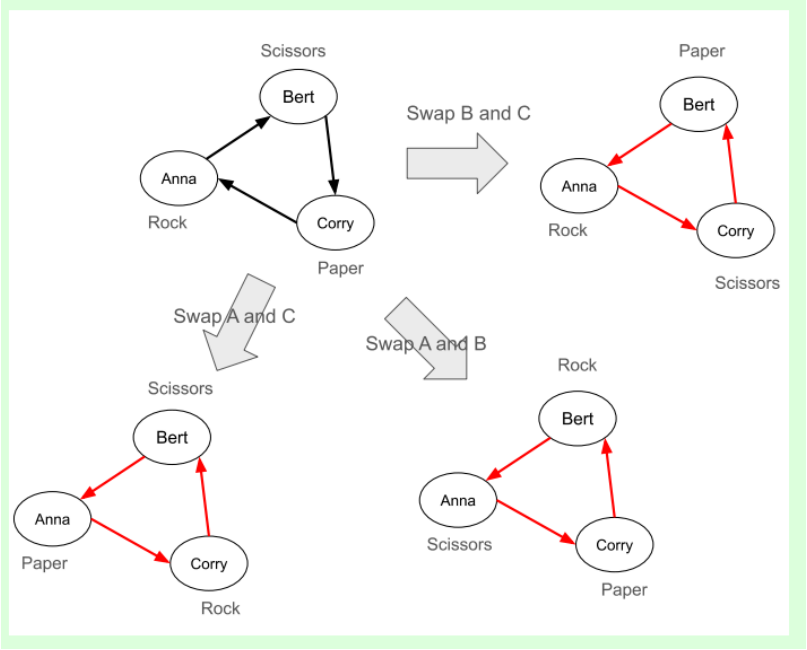
C) Neatkarīgi no tā, cik maiņas viņi nolemj veikt, Bertam vienmēr jāapmainās ar kārtīm ar Koriju.

D) Bertam jāgarantē tikai tas, ka maiņu skaits ir pāra skaits neatkarīgi no tā, kuras maiņas tiek veiktas.

Atbildes skaidrojums

Pareizā atbilde ir D.

Lai to atrisinātu, jāņem vērā, ka, veicot apmaiņu, neatkarīgi no tā, kura kārts ir katram spēlētājam, visas uzvaras un zaudējumi tiek apgriezti pretēji:



Šajā attēlā bultiņas norāda, kāds ir spēles rezultāts (kurš uzvar). Tāpēc pēc pāra skaita apmaiņu uzvaras ir tādas pašas kā sākotnējā situācijā. Un otrādi, pēc nepāra skaita apmaiņu uzvaras būs pretējas. Tā kā Berts vēlas uzvarēt Koriju, un tas notiek sākotnējā situācijā, viņam ir jāgarantē tikai pāra skaits maiņu.

Šī ir informātika

Šis uzdevums ir saistīts ar invariantiem, kas nozīmē atrast lietas, kas nemainās noteiktos apstākļos, vai identificēt apstākļus, kuros lietas ir tādas pašas kā iepriekš.

Invarianti ir būtiski matemātikā un datorzinātnēs, jo, ja mēs spējam tos atrast, mēs varam samazināt analizējamo gadījumu skaitu līdz mazākam skaitam, padarot uzdevumu vieglāk risināmu.

Šī ir skaitļošanas domāšana

Šablonu atpazīšana ir galvenā prasme šī uzdevuma risināšanā.
Sākumā uzdevums var šķist ļoti sarežģīts, jo šķiet, ka ir pārāk daudz iespējamo iznākumu, lai tos visus analizētu. Tomēr, ja sistemātiski analizē dažus gadījumus, nav grūti pamanīt, ka to nav tik daudz un ka tos var grupēt pēc maiņu skaita paritātes

Printera tīrīšana

Kongo demokrātiskā republika

Lai saglabātu kvalitāti, tintes printeriem pēc katra izdrukas cikla ir nepieciešams tīrīšanas cikls:

- Tīrīšanas cikls nejauši ilgst 1 vai 2 minūtes.
- Katrs izdrukas cikls var ilgt 1 vai 2 minūtes.

Frenkam ir piekļuve tintes printerim tikai 6 minūtes.

- Viņš var izlemt, vai sākt ar izdrukas vai tīrīšanas ciklu.
- Viņš var izlemt drukāt 1 vai 2 minūtes katrā izdrukas ciklā.
- Viņš nevar kontrolēt vai iepriekš zināt tīrīšanas ciklu ilgumu.
- Viņš vēlas maksimāli palielināt drukāšanas laiku.
- Viņam darbs jāpabeidz ar tīrīšanas ciklu. Pēdējam tīrīšanas ciklam ir jāsākas un jādarbojas vismaz 1 minūti, bet nav noteikti jābeidzas viņam atvēlēto 6 minūšu laikā.

Derīga grafika piemērs:

1. Frenks sāk ar tīrīšanas ciklu (kopējais laiks: 2 minūtes).
2. Frenks drukā 1 minūti (kopējais laiks: 3 minūtes).
3. Tīrīšanas cikls ilgst 1 minūti (kopējais laiks: 4 minūtes).
4. Frenks drukā 1 minūti (kopējais laiks: 5 minūtes).
5. Sākas tīrīšanas cikls.

Neatkarīgi no tīrīšanas cikla ilguma, Frenks šajā ciklā pabeidz savu 6 minūšu darbu.

Jautājums / Izaicinājums

Kura ir pirmā no 6 minūtēm, kad Frenkam ir jāsāk pirmais tīrīšanas cikls, lai garantētu, ka viņš pabeigs savu darbu ar tīrīšanas kārtu?

Atbilžu varianti / Interaktivitātes apraksts

A. no paša sākuma
B. pēc 1 minūtes
C. pēc 2 minūtēm
D. pēc 3 minūtēm

Atbildes skaidrojums

Pareizā atbilde ir: C
Viena no šī uzdevuma risināšanas stratēģijām ir sākt no beigām un strādāt atpakaļgaitā:

- Tā kā Frenkam jāpabeidz savs darbs ar tīrīšanas kārtu, tam beigās jārezervē vismaz 1 minūte. Un tā kā Frenks nevēlas, lai pēc šīs pēdējās tīrīšanas kārtas sāktos drukāšanas darbs, Frenkam beigās jārezervē tieši 1 minūte. Tāpēc Frenkam pēdējā tīrīšanas kārta jāsāk 5 minūtes pēc sākuma. Tas nozīmē, ka Frenkam pirms pēdējās tīrīšanas kārtas būs 5 minūtes.
- Viņš var izlemt, ka, ja iepriekšējā tīrīšanas kārta bija 1 minūti gara, viņš nosūtīs 2 minūšu drukāšanas uzdevumu, un pretējā gadījumā, ja iepriekšējā tīrīšanas kārta bija 2 minūtes gara, viņš nosūtīs 1 minūtes drukāšanas uzdevumu. Kopā drukāšanas un tīrīšanas kārtas tad aizņems tieši 3 minūtes.
- Tad paliek 2 minūtes, ko Frenks var izmantot sākumā. Lai maksimāli palielinātu drukāšanas laiku, viņš var izlemt izmantot šīs 2 minūtes drukāšanai.

Tas nozīmē, ka pirmajai tīrīšanas kārtai jāsākas ne agrāk (un patiesībā ne vēlāk) kā 2 minūtes pēc sākuma.

Šī ir informātika

Šī ir plānošanas problēma, kas darbojas ar deterministiskiem un stohastiskiem procesiem. Deterministiskie procesi ir pilnībā kontrolējami, savukārt stohastiskos procesus ietekmē nejauši notikumi, un tos nevar pilnībā paredzēt vai kontrolēt. Šādas plānošanas piemērus var atrast vides attīrīšanā, projektu vadībā, naftas izpētē, sensoru plānošanā mobilajos robotos un cikla laika modelēšanā, kā arī daudzās citās jomās.

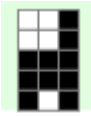
Uzdevuma idejas ir ņemta no Nim spēles. Sērkociņu vietā (visizplatītākais modelis Nim spēlēs) mums ir tīrīšanas un drukāšanas ciklu ilgums ar diviem spēlētājiem: Frenku un "Nejaušas tīrīšanas procesu". Nim spēle ir tā, ko spēļu teorijā sauc par nulles summas spēli. Nim spēles bieži izmanto, apmācot mākslīgo intelektu, īpaši ieviešot pastiprināšanas mācīšanos, lai parādītu, kā dators var iemācīties spēlēt spēli, to spēlējot.

Ciparu mikslis

Čehija

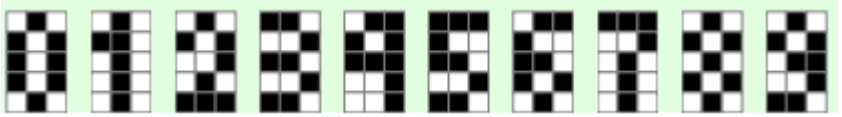
Stundas laikā Džemmai ir garlaicīgi, un viņa sāk aizpildīt kvadrātus savā lapā, lai attēlotu skaitļus. Viņa izdomā īpašu veidu, kā attēlot savu iecienītāko divciparu skaitli, 42: no diviem ciparu "4" un "2" attēlojumiem 3 × 5 režģī, kas parādīts zemāk, viņa izveido jaunu režģi, kur katrs kvadrāts ir melns tad un tikai tad, ja tieši viens no diviem atbilstošajiem kvadrātiem attēlos "4" un "2" ir melns:

Džemmai izmantoja šo pašu metodi, lai attēlotu citu divciparu skaitli:



Jautājums / Izaicinājums

Kuru skaitli Džemmai attēloja, zinot, ka ciparu sākuma attēli ir šādi? Ja ir vairākas atbildes, norādiet tikai vienu no tām.



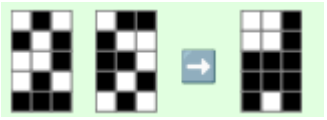
Atbilžu varianti / Interaktivitātes apraksts

Vesels skaitļi no 10 (ieskaitot) līdz 99 (ieskaitot)

Kā papildinājumu var piedāvāt divus interaktīvus 3 × 5 režģus kopā ar aprēķinātu trešo režģi ar tādu pašu izmēru, kas parāda pirmo divu režģu pikseļu XOR.

Atbildes skaidrojums

Divas pareizās atbildes ir 26 un 62. Apvienojot šos divus ciparus, iegūst vēlamo attēlu:



Ņemiet vērā, ka, apvienojot tos apgrieztā secībā, joprojām iegūsiet to pašu attēlu, jo to apvienošanas metode ir simetriska attiecībā pret tiem.

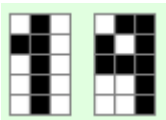
Lai atrisinātu šo uzdevumu, mums jāatrod veidi, kā noteikt divus pareizos ciparus, vienlaikus izvairoties no visu iespējamo kombināciju meklēšanas. To ir vienkārši par daudz: kopā 100, no “00” līdz “99”. Tā kā viencipara skaitļos mēs parasti izlaižam pirmo 0 (mēs rakstītu “7”, nevis “07”), tas samazina kombināciju skaitu līdz 90 vai pat 45, ja ņemam vērā iepriekš minēto simetriju, taču tas joprojām ir par daudz, lai to pilnībā pārbaudītu.

Pirmā lieta, kas jāsaprot, ir tā, ka apvienošanas darbība dod melnu pikseli dotajā pozīcijā ikreiz, kad šis pikselis atšķiras divos avota ciparos. Tas izriet no tā, ka rezultātā iegūstam melnu pikseli ikreiz, kad mums ir tikai viens melns pikselis, ja ņemam vērā avota attēlus. Ja aplūkojam vienu pikseli, apskatot visas iespējas, iegūstam šādu atbilstības tabulu:

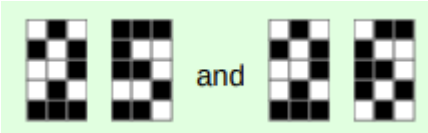
Pikselis pirmajā attēlā	Pikselis otrajā attēlā	Melno pikseļu skaits	Pikselis kombinētajā attēlā
Balts	Balts	0	Balts
Melns	Balts	1	Melns
Balts	Melns	1	Melns
Melns	Melns	2	Balts

Apzinoties to, mēs varam uzreiz izslēgt visus skaitļus, kas sastāv no viena un tā paša cipara divas reizes, piemēram, “22” vai “99”, jo visi pikseļi starp abiem ciparie vienmēr būtu vienādi, un tas radītu pilnīgi baltu kombinēto attēlu.

Viens no veidiem, kā pieiet risinājuma meklēšanai, ir saprast, ka kombinētā attēla labā malējā kolonna ir pilnīgi melna, kas norāda, ka abiem avota ciparie šajā kolonnā jābūt visiem dažādiem pikseļiem. Viens acīmredzams šāds ciparu pāris ir “1”–“4” — pēdējā kolonna ir pilnīgi tukša, ja ir “1”, un pilnībā pilna, ja ir “4”:



Diemžēl, aplūkojot, piemēram, otro rindu vai apakšējo rindu, mēs izslēdzam šo pāri, jo tas vairs neatbildīs kombinētajam attēlam. Divi citi nedaudz mazāk acīmredzami pāri, kuros visi pikseļi pēdējā kolonnā ir atšķirīgi, ir “2”–“5” un “2”–“6”:



Aplūkojot atlikušos kombinētā attēla pikseļus (piemēram, pašu pirmo pikseli augšējā rindā), mēs varam izslēgt “2”–“5”. Mums paliek pāris “2”–“6”, kas atbilst visiem pikseļiem. Ņemot vērā iepriekš minēto simetriju, mēs secinām, ka gan 26, gan 62 ir iespējamās atbildes.

Šī ir informātika

Centrālā operācija, kas tiek izmantota kombinētā attēla izveidē, faktiski ir loģiskā operācija “izslēdzošais VAI” (ExclusiveOR, XOR). Būla operācija darbojas ar patiesām/aplamām vērtībām (sauktas par loģiskām vērtībām) un iegūst citu loģisko vērtību. Šī ir loģiskās operācijas XOR patiesuma tabula, kur Z ir izvade:

A	B	Z = XOR(A, B)
0	0	0
1	0	1
0	0	1
1	1	0

Līdzību ar iepriekš parādīto pikseļu atbilstības tabulu var pamanīt, ja 0 aizstājam ar tukšiem kvadrātiem un 1 ar melniem kvadrātiem.

Citas loģiskās operācijas ir UN un VAI. Šādas operācijas tiek plaši izmantotas aparatūras sistēmu projektēšanā loģisko vārtu veidā. Tie ir pamatelementi, kas saņem ievadā divus signālus un "aprēķina" izvadi saskaņā ar patiesuma tabulu.

Kriptogrāfijā XOR ir centrālā loma simetriskās šifrēšanas shēmās, jo tā ir atgriezeniska: divas reizes piemērojot vienu un to pašu XOR operāciju ar vienu un to pašu atslēgu, tiek atjaunota sākotnējā vērtība. (Patiešām, šeit, apvienojot apvienoto attēlu ar vienu no divu ciparu attēliem, tiks iegūts... otra ciparu attēls!) Šī īpašība padara XOR noderīgu gan datu kodēšanai, gan dekodēšanai. XOR tiek plaši izmantots arī kļūdu noteikšanas un labošanas metodēs, piemēram, kontrolsummu un cikliskās redundances pārbaudēs (CRC), kur tas palīdz identificēt neatbilstības starp pārraidītajiem un saņemtajiem datiem.

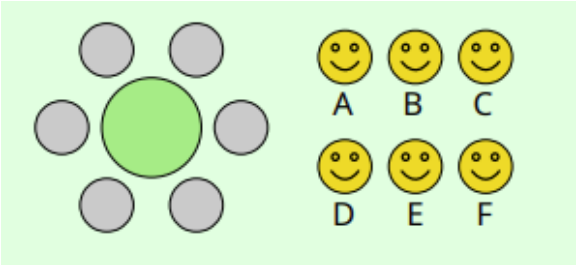
Šī ir skaitļošanas domāšana

Šis uzdevums galvenokārt apmāca paraugu saskaņošanu, pārbaudot pikseļus avota attēlos un apvienotajā attēlā.

Priecīgā aplī

Lietuva

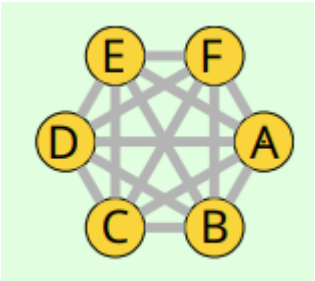
Seši draugi vēlas vakariņot pie apaļa galda. Visi no viņiem dod priekšroku sēdēt blakus saviem labākajiem draugiem un jūtas skumji, ja sēž blakus kādam, kas viņiem nepatīk.



Jautājums / Izaicinājums

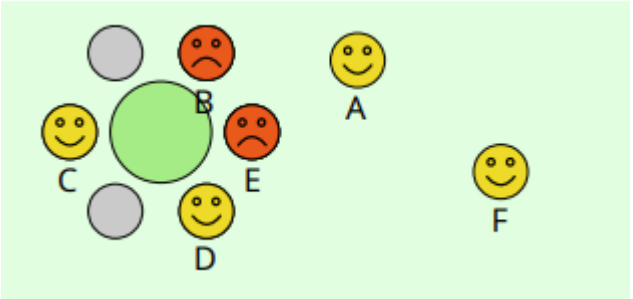
Sakārtojiet draugus ap galdu, velkot un nometot viņu simbolus uz krēsliem. Ja viņi tiek apsēdināti blakus kādam, kas viņiem nepatīk, seja kļūs sarkana un skumja. Atrodiet piemērotu sēdvietu, lai visi būtu laimīgi un smaidīgi.

Zemāk esošā interaktīvā diagramma dota, lai palīdzētu jums atrisināt uzdevumu. Atbilde nav atkarīga no tās. Varat noklikšķināt uz saitēm, lai mainītu to krāsu.

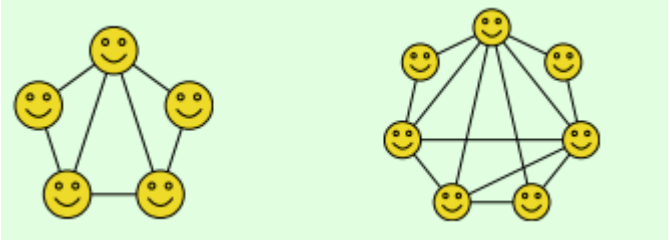


Atbilžu varianti / Interaktivitātes apraksts

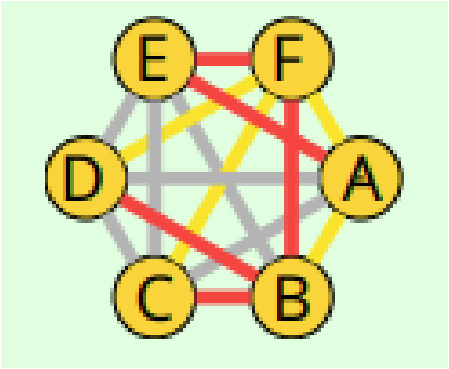
Interaktīvo versiju var pārbaudīt šeit:
https://bebras-lodge.sp.fsf.vu.lt/public/d8fa0bc9897a40b09b65



Draugus (smaidīgās sejas) var pārvietot un vilkt uz krēsliem (pelēki apļi). Sākumā visi cilvēki atrodas sānos un smaida. Kad cilvēku velk uz krēsla, ja viņam nepatīk vismaz viens blakus sēdošs cilvēks, smaids mainās uz sarauktu pieri. Cilvēku, kuriem patīk viens otram, grafs tiek ģenerēts šādi: vispirms ņemam attēlā redzamo grafu un katru tur savienoto cilvēku pāri, kas viens otram patīk, un katrā nesavienotajā pārī abi cilvēki viens otru ienīst (vienkāršības labad pietiktu ar to, ka viens cilvēks ienīst otru). Tad mēs varētu katrai personai piešķirt vārdus nejauši, bet šādā gadījumā kādam varētu paveikties un nejauši pareizi novietot cilvēkus ap galdu. Tātad sākumā visi savienojumi apzīmētajā grafā ir nezināmi jeb atrodas "kvantu stāvoklī". Tiklīdz divi cilvēki tiek nomesti blakus viens otram, stāvoklis "sabrūk" – viņi vai nu sāk viens otram patikt, vai nepatikt. Lai izvēlētos visus iespējamos $6! = 6 \cdot 5 \cdot 4 \cdot 3 \cdot 2 \cdot 1 = 720$ cilvēku permutācijas grafikā tiek pārbaudītas un vai nu atmestas (jo tās ir pretrunā ar jau zināmu informāciju), vai arī tām tiek piešķirts svars kā katra zināmā pāra vērtību summa (1 nepatikai vienam pret otru, 2 patikai diagonālēs, 0 patikai sešstūra malās dotajā attēlā). Pēc tam tiek izvēlēta šķautne ar permutāciju ar vislielāko svaru (vai nejauši, ja svari ir vienādi). Ņemiet vērā, ka attēlā redzamajam grafam ir maksimālais šķautņu skaits ar tieši vienā Hamiltona ciklā. Šeit ir grafi uzdevuma vieglākajai/grūtākajai versijai:



Uzdevums bez norādījumiem var būt milzīgs, tāpēc tiek dota interaktīva diagramma, lai atzīmētu zināmās saiknes: noklikšķinot, mainās dzeltenā/sarkanā/pelēkā krāsa, piemēram:



Atbildes skaidrojums

Viens no veidiem, kā atrisināt uzdevumu, ir secināt, kurai personai patīk kuri citi cilvēki, mēģinot novietot visus cilvēku pārus uz krēsliem blakus, tādējādi ģenerējot grafiku, kas ir homomorfs (t. i., ar tādu pašu struktūru) kā iepriekš redzamajā attēlā. Tad mēs varam

atrast Hamiltona ciklu (t. i., ciklu, kas katru virsotni apmeklē tieši vienu reizi) grafā un atbilstoši tam izvietot draugus ap galdu. Vēl viens veids ir meklēt Hamiltona ciklu, vienlaikus atvasinot grafu: sākumā novietojiet jebkuru personu uz jebkura krēsla, tad jebkuru citu blakus, līdz abi ir laimīgi, un tā tālāk, un, ja mēs nonākam pie situācijas, kad jebkuras personas novietošana kādu padara nelaimīgu, mēs noņemam vienu personu (atpakaļejoša izsekošana) un turpinām meklēšanu. Ar šo grafu, ja permutācija tiktu izvēlēta nejauši, būtu (iespējams) 7/60 varbūtība (11/30 versijā ar 5 cilvēkiem) pareizi uzminēt secību bez atpakaļejošas izsekošanas.

Šī ir informātika

Šis uzdevums ir balstīts uz informātikas pamatjēdzieniem, kas saistīti ar grafu teoriju un algoritmisku problēmu risināšanu. Konkrēti, tas pēta Hamiltona ceļus un ciklus — maršrutus grafā, kas katru mezglu apmeklē tieši vienu reizi (ceļam) vai vienu reizi un atgriežas sākuma punktā (ciklam). Šīs problēmas ir ļoti aktuālas reālās pasaules lietojumprogrammās, piemēram, maršrutu optimizēšanā transportā un loģistikā, kur ir ļoti svarīgi samazināt braucienu skaitu, vienlaikus aptverot visus galamērķus.

No skaitļošanas viedokļa Hamiltona ceļu vai ciklu atrašana pieder pie problēmu klases, kas pazīstama kā NP-pilnas, kas nozīmē, ka tās ir ļoti grūti efektīvi atrisināt, palielinoties grafa izmēram. Nav zināms algoritms, kas varētu ātri (polinomiālā laikā) atrisināt visus gadījumus, kas padara šo par labu skaitļošanas ziņā sarežģītas problēmas piemēru.

Lai risinātu šādas problēmas, bieži tiek izmantots atpakaļejošas izsekošanas algoritms — stratēģija, kas sistemātiski izpēta visas iespējamās iespējas, atmet tās, kas ved uz strupceļu, un atkāpjas, lai izmēģinātu alternatīvas. Šī pieeja parāda, kā informātika nodarbojas ne tikai ar problēmu risināšanu, bet arī ar skaitļošanas sarežģītības novērtēšanu, efektīvu algoritmu izstrādi un izpratni par to, ko var aprēķināt saprātīgā laika periodā.

Strādājot ar Hamiltona ceļiem, studenti tiek iepazīstināti ar galvenajām informātikas jomām: datu struktūrām (grafiem), algoritmisko domāšanu (atpakaļejoša izsekošana), optimizācijas un sarežģītības teoriju — tas viss ir būtiski reālās pasaules problēmu risināšanai, izmantojot skaitļošanas metodes.

Šī ir skaitļošanas domāšana

Hamiltona ceļa un cikla problēmu risināšanā ir jāpielieto vairākas būtiskas skaitļošanas domāšanas stratēģijas. Šīs problēmas ietver katra mezgla apmeklēšanu grafā tieši vienu reizi un atgriešanos sākuma punktā (cikla gadījumā), kas atspoguļo reālās pasaules

scenārijus, piemēram, maršrutēšanu vai plānošanu. Lai to panāktu, studenti izmanto grafu šķērsošanas metodes, lai sistemātiski izpētītu savienojumus starp mezgliem. Palielinoties grafa izmēram, iespējamo ceļu skaits strauji pieaug, padarot izsmeļošo meklēšanu nepraktisku. Tāpēc tiek izmantota atpakaļizsekošana, lai izpētītu ceļu, atklātu strupceļus un atgrieztos pie iepriekšējiem soļiem, lai izmēģinātu alternatīvas.

Modeļu dedukcija palīdz sašaurināt meklēšanas telpu, identificējot struktūras vai konfigurācijas, kas nekad nevar novest pie derīgiem risinājumiem. Var izmantot arī heiristiskās meklēšanas stratēģijas, lai virzītu meklēšanu uz daudzsološākām grafa zonām, prioritāri piešķirot ceļiem,
kas, pamatojoties uz daļējām zināšanām, visticamāk, novedīs pie panākumiem.

Kopā šīs pieejas atbalsta loģiskās spriešanas, stratēģiskās izpētes un efektīvas problēmu risināšanas attīstību, kas viss ir skaitļošanas domāšanas pamatā.

Maģiskā atslēga

Dominikāna

Artūram ir datorpele, kādas nav nevienam citam: tai ir taustiņš — Maģiskā atslēga! Pele glabā skaitītāju m, kas ar katru Maģiskās atslēgas nospiešanu palielinās par 1. Taču tā nav vienīgā lieta, kas notiek pēc nospiešanas: skaitļi no m līdz 1 tiek ierakstīti arī īpašā failā. Pirmie 5 taustiņa nospiešanas reižu rezultāti failā rada šādu rezultātu:

1, 2, 1, 3, 2, 1, 4, 3, 2, 1, 5, 4, 3, 2, 1.

Nospiedienu skaits (m)	Faila saturs
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

Jautājums / Izaicinājums

Artūrs ir sajūsmināts par šo iespēju un jau pārāk daudz reižu nospiedis pogu. Tagad viņam failā ir vesela virkne skaitļu, un viņš vēlētos uzzināt, kāds skaitlis ir ierakstīts 127. pozīcijā.

Atbilžu varianti / Interaktivitātes apraksts

Atvērta atbilde: vesels skaitlis no [1, 16].

Atbildes skaidrojums

Pareizā atbilde šeit ir 10. Mēs redzam, ka faila lielums (pēc tajā ierakstīto skaitļu skaita) pēc katras nospiešanas reizes ir: 1, 3, 6, 10, 15, 21, 28... Kopumā, ja n ir nospiešanas reižu skaits, faila lielums būs T(n) = n (n + 1) / 2. Lai iegūtu atbildi, mēs varam paļauties uz simulāciju. Mēs varam mēģināt atbildēt uz šādu jautājumu: kāds ir mazākais n, lai T(n) >= 127. Skaidrs, ka jebkurš n, kas rada mazāku vērtību, nevar būt atbilde, jo tas nozīmētu, ka failā ir ierakstīti mazāk nekā 127 skaitļi. Tāpēc mēs pakāpeniski palielinām n, līdz atrodam pirmo apmierinošo skaitli. Mēs redzam, ka T(15) = 120, kas ir pārāk mazs skaitlis, bet T(16) = 136. Nospiežot pogu 16. reizi, ierakstītie skaitļi būs [16, 15, 14, 13, 12, 11, 10, ..., 1]. Mēs zinām, ka 16 ir 121. ierakstītais skaitlis. Tāpēc mēs varam vienkārši skaitīt atpakaļ no 16, līdz sasniedzam 127. skaitli, kas būs 10.

Šī ir informātika

Datorzinātnēs simulācija ir reāla (vai hipotētiska) procesa vienkāršota modeļa izveide un pēc tam tā izpilde. Modeļus un simulācijas var izmantot, lai veiktu prognozes, vai nu palaižot simulāciju un novērojot rezultātu, vai ievadot modelim dažādas sākotnējās vērtības un aprēķinot rezultātu.

Šajā uzdevumā, nospiežot taustiņu pirmo reizi, tas ieraksta [1]. Nospiežot to otro reizi, tas ieraksta [2, 1]. Līdzīgi tas trešo reizi ieraksta [3, 2, 1]. Un tā tālāk. Šī ir vienkārša simulācija. Sekojot līdzi skaitļiem, kas ierakstīti pēc katra soļa, un turpinot soli tālāk, mēs varam sasniegt reizi, kas ierakstīs meklējamo atbildi.

Šīs virknes skaitļus sauc par trīsstūra skaitļiem, un tie ir ļoti izplatīti datorzinātnēs. Tie parādās triviālos gadījumos, piemēram, iekļautu ciklu veikto darbību skaita skaitīšanā, dažu kārtošanas algoritmu sarežģītības noteikšanā un sarežģītākos scenārijos, kas saistīti ar datu struktūrām un algoritmu analīzi.

Šī ir skaitļošanas domāšana

Lai atrisinātu šo uzdevumu, ir būtiska modeļu atpazīšana. Virknes, par kādu palielinās faila lielums, atpazīšana, ir līdzīga modeļu identificēšanai, kas var palīdzēt izveidot labākus algoritmus. Šajā uzdevumā noderīga ir arī dekompozīcija. Uzdevuma telpu var sadalīt divās daļās: visi skaitļi, kurus var izlaist, un atbilstošā skaitļu daļa, kur atrodama atbilde. Šīs kompetences veicina dziļu skaitļošanas domāšanas izpratni, izmantojot vienkāršus scenārijus.

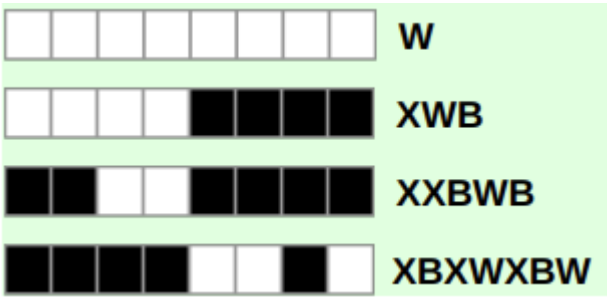
Rindas kodējums

Polija

Sāra spēlē spēli. Viņai ir kvadrātu virkne, kas ir vai nu melni, vai balti, un viņa vēlas tos attēlot noteiktā veidā saskaņā ar šādu noteikumu:

- Ja visi pašreizējās virknes kvadrāti ir balti, viņa vienkārši raksta “W”
- Ja visi pašreizējās virknes kvadrāti ir melni, viņa vienkārši raksta “B”
- Pretējā gadījumā viņa rakstīs “X”,
    - kam seko tā paša noteikuma rezultāts, kas piemērots pašreizējās virknes kreisajai pusei,
    - kam seko tā paša noteikuma rezultāts, kas piemērots pašreizējās virknes labajai pusei.

Šeit ir daži piemēri, kā noteikums darbojas 8 kvadrātu virknei:



Jautājums / Izaicinājums

Kā Sāra attēlos šo kvadrātu virkni?

Atbilžu varianti / Interaktivitātes apraksts

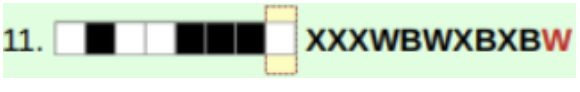
(A) XXBWBXWXWB
(B) XXXBWBXWXWB
(C) XXXWBXWWXXBBXBW
(D) XWBWXBBW
(E) XXXWBWXBXBW
(F) XXWBWXBXBW

Skatīt komentārus, lai uzzinātu par jaunu, interaktīvu versiju.

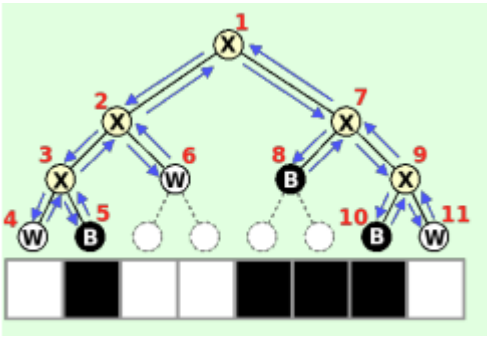
Atbildes skaidrojums

Pareizā atbilde ir (E) XXXWBWXBXBW

Pareizo atbildi varam konstruēt pa rakstzīmei šādi:

1. 	Visi kvadrāti nav vienā krāsā, tāpēc sākam ar X
2. 	Visi kreisās puses kvadrāti nav vienā krāsā, tāpēc atkal X
3. 	Visi kreisās puses kreisās puses kvadrāti nav vienā krāsā, tāpēc atkal X
4. 	Kreisās puses kreisās puses pirmais kvadrāts ir balts - W
5. 	Kreisās puses kreisās puses otrais kvadrāts ir melns - B
6. 	Visi kreisās puses labās puses kvadrāti ir balti - W
7. 	Visi labās puses kvadrāti nav vienā krāsā, tāpēc X
8. 	Visi labās puses kreisās puses kvadrāti ir melni - B
9. 	Labās puses labās puses kvadrātu krāsa atšķiras - X
10. 	Labās puses labās puses pirmais kvadrāts ir melns - B
11. 	Labās puses labās puses otrais kvadrāts ir balts - W

Vēl viens veids, kā to vizualizēt, ir parādīts šajā attēlā, kur zilās bultiņas norāda, kā mēs "pārvietojamies", un sarkanais skaitlis norāda, kādā secībā mēs "rakstām" attēlojumu:



Kā veidot koku: vispirms mēs rakstām secību ar burtiem katram kvadrātam: WBWWBBBW. Tā mēs iegūstam koka zemāko līmeni - lapas. Pēc tam mēs pārejam uz augstākiem mezgliem. Ja abi to bērni ir vienādi (tikai W vai B), mēs tos apvienojam vienā burtā, izdzēšot abus bērnus. Tas kļūst par jaunu koka lapu. Pretējā gadījumā mēs mezglā ierakstām X.

Kā no koka izveidot pareizo secību: mēs sākam no augšējā mezgla. Mēs šķērsojam koku, vispirms ejot pa kreisi katrā mezglā. Katrā mezglā, kuru mēs šķērsojam, mēs ierakstām burtu, kas parādās mezglā pirmo reizi, kad to apmeklējam. Kad mēs sasniedzam lapu, mēs atgriežamies iepriekšējā mezglā un ejam pa labo ceļu. Ja ir apmeklēti abi ceļi, mēs atgriežamies iepriekšējā mezglā.

Variants (B) ir nepareizs, jo tas dod "papildrindu" (melns, kad kvadrāts ir balts, un otrādi).

Variants (C) ir nepareizs, jo tajā ir divu vienādu kvadrātu sadaļas, kas nav reducētas līdz vienai rakstzīmei, bet gan sadalītas divās vienādās rakstzīmēs. Varianti (A), (D) un (F) ir nepareizi, cita starpā, jo tie nesākas ar trim X, kā nepieciešams, lai attēlotu pirmos kvadrātus.

Šī ir informātika

Noteikums, ko Sāra izmantoja kvadrātu secību attēlošanai, ir rekursīvs.

Rekursija ir ļoti spēcīga algoritmiska metode, ko plaši izmanto datorzinātnēs.

Tā ļauj aprakstīt risinājuma procedūru pašu par sevi: ja jums ir vienkārša situācija, jūs tieši norādāt tās risinājumu (šajā uzdevumā "W" un "B"); ja to nedarāt, jūs sadalāt problēmu līdzīgās mazākās problēmās un risināt tās tādā pašā veidā, pēc tam apvienojot risinājumus, ko iegūstat mazākām problēmām, risinājumā tai problēmai, ar kuru sākāt (šajā uzdevumā "X", kam seko kreisās un labās puses risinājumi).

Rekursiju pielieto daudzās jomās, piemēram, Fibonači analizēja trušu populācijas lielumu un atrada savu slaveno rekurento formulu. Izmantojot rekursiju, jūs varat aprēķināt trušu skaitu, kas jums būs pēc vairākiem gadiem. Vēl viens piemērs ir fraktāļi, kas ir ģeometriskas formas, kas konstruētas, izmantojot rekursiju. Lai informāciju varētu automātiski apstrādāt, tā ir jāattēlo simboliski; turklāt ir svarīgi atrast attēlojumus, kas nodrošina efektīvu apstrādi. Tāpēc attēlojumu formu un datu struktūru izpēte ir ļoti svarīga informātikas joma. Informācijas konvertēšanas procesu no viena attēlojuma uz citu sauc par kodēšanu.

Ja šajā uzdevumā ir zināms precīzs kvadrātu virknes garums, kodu var konvertēt atpakaļ uz kvadrātu virkni. Šādus kodus sauc par apgriežamiem. Ja garums ir fiksēts, šis konkrētais kods ir prefiksa kods. Tas nozīmē, ka neviens kods nav cita koda prefikss. Šādus kodus var ātri apgriezt, un tiem nav nepieciešami lieli skaitļošanas resursi.

Šī ir skaitļošanas domāšana

Šajā uzdevumā ir nepieciešama abstrakcija, lai izprastu teksta pamattekstā sniegto rekursīvo definīciju. Pēc tam mēs varam pieiet uzdevuma risināšanai soli pa solim, sadalot to mazākos uzdevumos un simulējot/novērtējot katru soli, lai iegūtu sākotnējo datu attēlojumu.

Kosmiskie stari

Čehija

Lūk, dīvains stāsts par Martinas bankas kontu. Viņa pārbaudīja savu bankas konta atlikumu savā telefonā un redzēja, ka viņai ir 8095 EUR. Bet pēkšņi skaitlis mainījās uz 7071 EUR. Viņa nebija veikusi nekādus bankas darījumus vai maksājumus, tāpēc bija apjukusi. Viņa nosūtīja ziņojumu bankai, lai noskaidrotu, kas noticis.

Banka izskatīja šo gadījumu un konstatēja, ka programmatūrā nav kļūdu un ka nav noticis kiberuzbrukums. Galu galā viņi secināja, ka atmiņā ir notikusi nejauša viena bita vērtības maiņa, ko izraisījis kosmiskais starojums.

Datorsistēmās skaitļi tiek glabāti binārā formātā. Binārā skaitļu sistēma izmanto tikai divus ciparus: 0 un 1. Pozīciju vērtības palielinās par divi, pārvietojoties pa skaitli no labās puses uz kreiso (no mazāk nozīmīgā cipara uz vairāk nozīmīgu ciparu). Zemāk esošajā tabulā parādīts, kā lasīt 8 bitu skaitli “00100011”:

128	64	32	16	8	4	2	1
0	**0**	**1**	**0**	**0**	**0**	**1**	**1**

Šis skaitlis decimālajā skaitīšanas sistēmā ir 32 + 2 + 1 = 35. Nākamajā tabulā ir mainījies viens bits. Šis ir skaitļa astotais bits (skaitot pozīcijas no labās puses uz kreiso).

128	64	32	16	8	4	2	1
1	**0**	**1**	**0**	**0**	**0**	**1**	**1**

Šī skaitļa vērtība tagad ir 128 + 32 + 2 + 1 = 163. Var redzēt, ka viena bita izmaiņas būtiski mainīja skaitļa vērtību. Bankas konta atlikums būtu jāuzglabā, izmantojot lielu bitu skaitu, jo atlikums varētu būt ļoti liels.

Jautājums / Izaicinājums

Kuru Martinas bankas konta atlikuma bitu mainīja kosmiskais starojums?

Atbilžu varianti / Interaktivitātes apraksts

A) 11. bits no 1 uz 0
B) 12. bits no 1 uz 0
C) 11. bits no 0 uz 1
D) 12. bits no 0 uz 1

Atbildes skaidrojums

Pareizā atbilde ir A

Paskaidrojums:

Noskaidrosim, cik lielā mērā ir mainījies konta atlikums, atņemot vērtības:
8095 – 7071 = 1024

No tabulām var redzēt, ka katra bita pozīcijas vērtība dubultojas, virzoties no labās uz kreiso pusi. Lai sasniegtu pozīcijas vērtību 1024, mums jāatrod 11. bits. Mēs varam pārbaudīt skaitļus un redzēt, ka binārajā formā tie atšķiras tikai par vienu ciparu 11. bitā:
8095 (bāze 10) = 1 1111 1001 1111 (bāze 2)
7071 (bāze 10) = 1 1011 1001 1111 (bāze 2)

Šī ir informātika

Datorsistēmas glabā skaitļus binārajā formā. Šie skaitļi tiek glabāti datora atmiņā, kurā var rasties kļūdas un darbības traucējumi. Tie parasti ir elektriskas dabas un saglabā vērtības 0 vai 1, uzlādējot vai izlādējot atmiņas komponenti.

Kosmiskais starojums, kas sastāv no ātri lādētām daļiņām, var sasniegt Zemes virsmu un izlidot cauri datora atmiņai. Kad tas sasniedz atmiņas vietu, kurā tiek glabāta viena informācijas bita vērtība, tas var to uzlādēt vai izlādēt.

Varbūtība ir ļoti maza, bet tā pastāv. Kosmosa kuģi ir vēl lielākās briesmās, jo kosmiskais starojums kosmosā ir daudz spēcīgāks. Tāpēc vienu un to pašu vērtību saglabāšanai kosmosa kuģos izmanto trīs atmiņas bankas. Strādājot, viņi pastāvīgi pārbauda, vai vērtības atšķiras viena no otras, un, ja tā, viņi noskaidro, kura no trim vērtībām atšķiras no pārējām, un to izlabo.

Ir dažādas specifiskas kļūdu noteikšanas un labošanas metodes (piemēram, Heminga kodi, Rīda-Solomona kodi utt.) ar dažādiem sarežģītības līmeņiem un kļūdu labošanas iespējām, lai risinātu datu kļūdas.

Šī ir skaitļošanas domāšana

Abstrakcija: atpazīt, kas ir svarīgs risinājumā, un koncentrēties tikai uz to.
Rakstu atpazīšana: bināro skaitļu pozicionālās vērtības modeļi.

Plūdu novēršana

Jaunzēlande

Bīvertaunā kalnos ir septiņi dīķi, kuros uzkrājas lietus ūdens. Spēcīgu lietusgāžu laikā šie dīķi strauji piepildās un pastāv pārplūšanas risks, kas varētu izraisīt nopietnus plūdus šajā apgabalā.

Lai to novērstu, pilsēta plāno izbūvēt jaunu cauruļvadu sistēmu, lai savienotu visus septiņus dīķus ar pilsētas galveno rezervuāru. Mērķis ir nodrošināt, lai ūdens no katra dīķa varētu droši ieplūst rezervuārā pat spēcīgu lietusgāžu laikā.

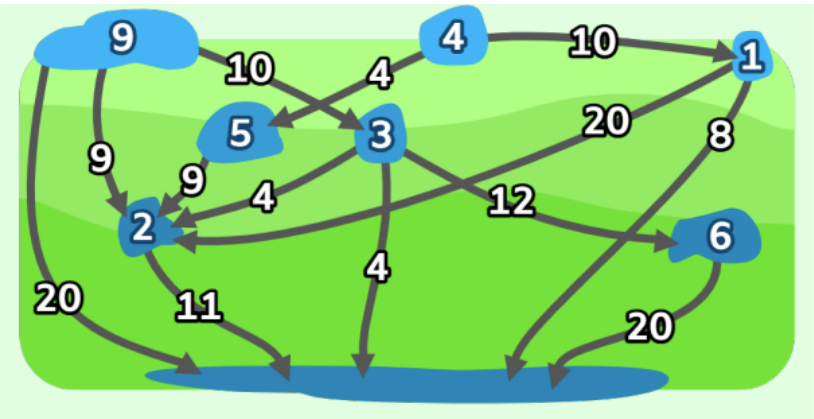
Inženieri ir identificējuši iespējamos cauruļvadu maršrutus starp dīķu pāriem (un starp dīķiem un rezervuāru). Katrai ierosinātajai caurulei ir noteikta jauda — ūdens daudzums, ko tā var pārvadīt stundā — un izmaksas, kas tiek aprēķinātas kā 1 miljons ASV dolāru, reizinātas ar tās jaudu.

Inženieru piedāvātajā projektā katram dīķim pietiek ar vienu, jebkāds jaudas, izejošo cauruli. Bet, ja dīķī ienāk citi cauruļvadi, tad ienākošo cauruļvadu kopējā jauda nedrīkst pārsniegt no dīķa izejošo cauruļvadu kopējo jaudu.

Lai samazinātu būvniecības izmaksas, pilsēta vēlas izbūvēt cauruļvadu sistēmu, kas:

• Tieši vai netieši savieno visus septiņus dīķus ar galveno rezervuāru,
• Ļauj sistēmai apstrādāt paredzamo ūdens plūsmu spēcīgu lietusgāžu laikā un
• Samazina kopējās būvniecības izmaksas.

Jūsu uzdevums ir palīdzēt noteikt visrentablāko veidu, kā uzbūvēt šīs caurules, lai visi dīķi būtu droši savienoti ar rezervuāru un varētu novadīt pietiekami daudz ūdens, lai novērstu plūdus spēcīgu lietusgāžu laikā.



Jautājums / Izaicinājums

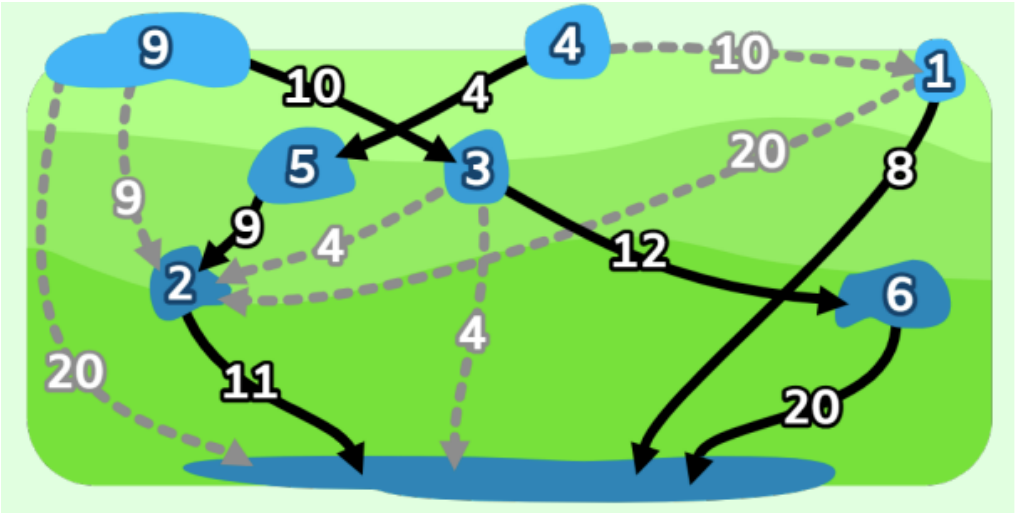
Kādas ir zemākās kopējās izmaksas, lai uzbūvētu cauruļu sistēmu, kas nodrošina, ka visi dīķi var droši novadīt savu ūdeni uz rezervuāru spēcīgu lietusgāžu laikā?

Atbilžu iespējas / Interaktivitātes apraksts

Vesels skaitlis

Atbildes skaidrojums

74



Tā kā esošo cauruļu nav, mums ir jāizbūvē jauna cauruļu sistēma no nulles, kas nodrošina, ka visi septiņi dīķi var novadīt lietus ūdeni rezervuārā bez applūšanas. Ir zināms ūdens daudzums, kas katram dīķim ir jānovada, un katrai caurulei ir ierobežota jauda un saistītās izmaksas (izmaksas = 1 miljons USD × jauda). Tā būtībā ir tīkla plūsmas problēma.

Pirmais novērojums ir tāds, ka 2., 5. un 6. mezglam ir tikai viens jauns savienojums lejup pa straumi, tāpēc tie noteikti jāiekļauj ar izmaksām 11 + 9 + 20 = 40.

Otrkārt, 1. mezglam ir divas izvēles iespējas. Ja 1. mezgls plūst uz 2. mezglu, tas neļaus 4. mezglam doties uz 5. mezglu (4+5+1+2 ir 12, kas ir > 11 ietilpība no 2. mezgla līdz rezervuāram). Tas nozīmē, ka ūdenim no 4. mezgla būtu jāplūst uz 1. mezglu, iegūstot kopējās izmaksas 20 (1→2) + 10 (4→1). Ja no 1. mezgla ūdens plūst tieši uz rezervuāru, tas ļauj ūdenim no 4. mezglam plūst uz 5. mezglu. Kopējās izmaksas ir 8 (1→rezervuārs) + 4 (4→5) = 12, kas ir mazāk nekā otrai iespējai.

Visbeidzot, 9. un 3. mezgls katru var savienot tieši ar rezervuāru ar izmaksām 20 + 4 = 24. Ūdens varētu doties arī no 9. mezgla uz 3. mezglu, apvienot plūsmu uz 12. un iet caur 6. mezglu, jo 12 + 6 ir mazāk nekā 20 ietilpība. Šis ceļš izmaksātu 10 + 12 = 22, kas ir mazāk, tāpēc šis ir ideāls maršruts. Kopējās izmaksas ir 40 + 12 + 22 = 74.

Šī ir informātika

Dīķus un savienojumus var attēlot kā grafu ar avota mezgliem un svarotām šķautnēm. Tie varētu arī attēlot tīkla joslas platumu un informācijas plūsmu izkliedētā skaitļošanas vidē, piemēram, internetā vai mobilo tālruņu tīklos. Maksimālās plūsmas atrašana ar minimālām izmaksām ir tradicionāla ierobežojumu optimizācijas problēma, ko parasti risina ar grafu šķērsošanas algoritmiem. Lai izvairītos no brutāla spēka, izmēģinot visas iespējas, lielākā daļa šo algoritmu sāk ar skaitļošanas ziņā dārgu, neoptimālu risinājumu un mēģina to iteratīvi uzlabot, lai sasniegtu optimālo risinājumu. Risinājumā kā pirmo soli var izmantot minimālo pārklājošo koku. Pārklājošie koki ir noderīgi, lai vienkāršotu daudzas grafu analīzes problēmas, kas saistītas ar mezglu savienojamību.

Šī ir skaitļošanas domāšana

Abstrakcija ir nepieciešama, lai vispirms samazinātu problēmu līdz mazākam grafa apjomam, izvēloties acīmredzamos savienojumus, kur ir tikai viena iespēja. Lai izveidotu daļēju risinājumu, kas ietver dažus mezglus un savienojumus, un virzītos uz galīgo risinājumu, ir nepieciešama rekursija vai iterācija.

Daži grafu šķērsošanas algoritmu pielietojumi ir noderīgi, īpaši tīkla plūsmas algoritmi, kur mēs sākam no "notekas" (t. i., rezervuāra) un virzāmies atpakaļ, šķērsojot grafu pa minimālās caurlaidības caurulēm.

Optimizācijai un ierobežojumu piemērošanai ir nepieciešama loģiska domāšana par to, kādas piešķires mainīgajiem ir iespējamas, lai nepārkāptu jaudas ierobežojumus (nosūtot pārāk lielu plūsmu pa cauruli), un lai salīdzinātu vairāku mainīgo piešķiršanas iespējas (piemēram, savienojot 1. un 4. mezglu).

Sabiedriskais transports

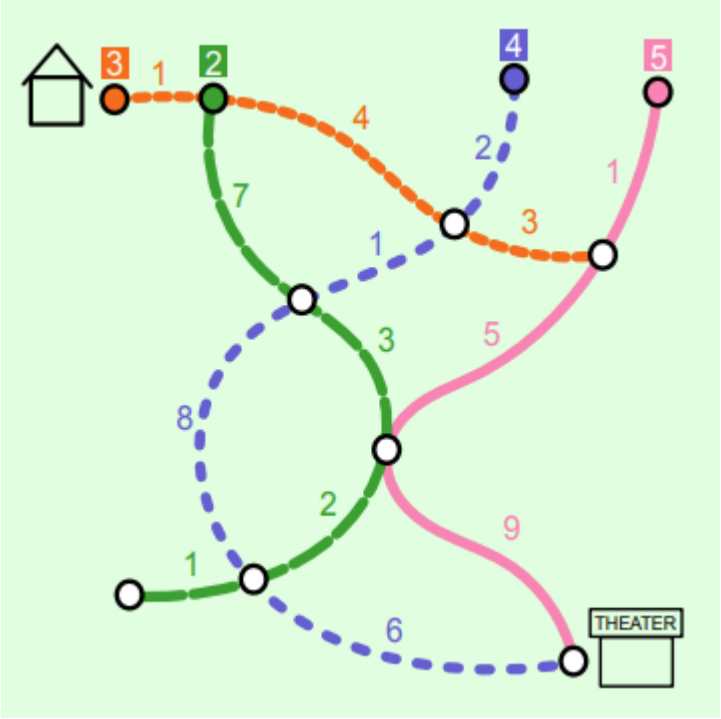
Lietuva

Markuss vēlas doties no savām mājām uz teātri ar autobusu. Pilsētā darbojas 4 vienvirziena autobusu maršruti. Autobusu pieturas ir apzīmētas ar apļiem ar melnām apmalēm . Autobusu maršruti ir apzīmēti ar dažādu krāsu līnijām. Krāsainās autobusu pieturas apzīmē katra maršruta sākuma pieturu.

Pirmie autobusi katrā maršrutā no to sākuma pieturām atiet vienlaicīgi. Pēc tam katrā maršrutā autobusi atiet ar dažādiem laika intervāliem. Skaitļi uz krāsainā fonā apzīmē laika intervālu starp autobusu atiešanu minūtēs. Piemēram, oranžā maršrutā autobusi atiet ik pēc 3 minūtēm – 0., 3., 6., 9. minūtē utt.

Skaitļi līniju segmentu tuvumā apzīmē, cik minūtes autobusam nepieciešams, lai veiktu attālumu starp divām pieturām. Apstāšanās pieturā un pasažieru iekāpšana neaizņem laiku (0 minūtes).

Pieturas, kurās krustojas divas vai vairākas autobusu līnijas, var izmantot, lai veiktu pārsēšanos no viena maršruta autobusa uz citu. Ja Markuss ierodas šādā pārsēšanās pieturā, viņš var pārsēsties uz autobusu, kas tur pienāk vienlaikus vai vēlāk par viņu.



Jautājums / Izaicinājums

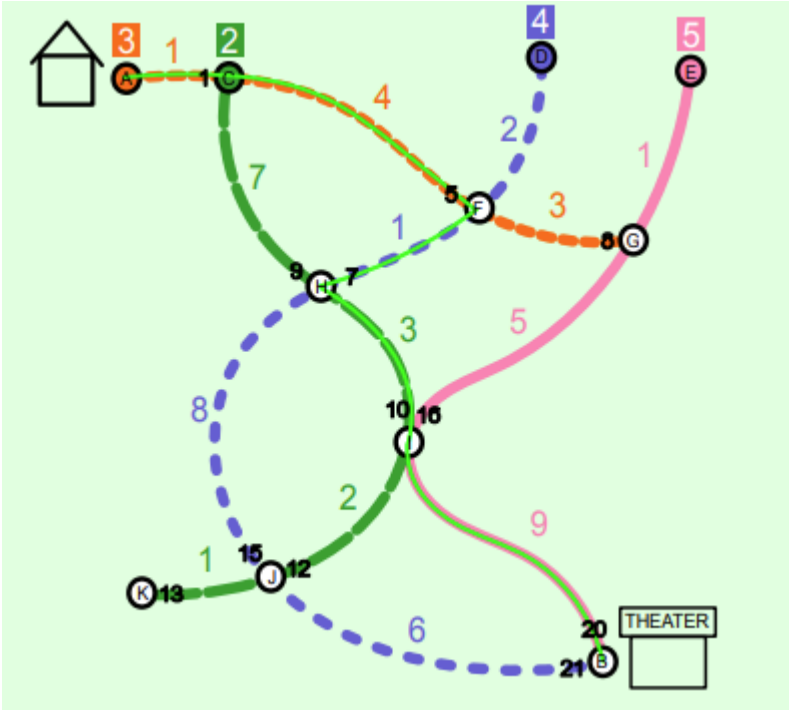
Ja Markuss dodas ceļā ar pirmo oranžā maršruta autobusu, kāds ir īsākais laiks minūtēs, kas viņam nepieciešams, lai nokļūtu teātrī?

Atbilžu varianti / Interaktivitātes apraksts

Veseli skaitļi no [0; +∞)

Atbildes skaidrojums

Pareizā atbilde ir 20 minūtes. Markusam jābrauc ar oranžo autobusu 2 pieturas (5 minūtes), tad ar zilo autobusu 1 pieturu (1 minūtes gaidīšana + 1 minūte), ar zaļo autobusu 1 pietura (3 minūtes) un ar rozā autobusu 1 pietura (1 minūtes gaidīšana + 9 minūtes). Atbildi var atrast, izvēloties pieturu ar jau zināmu ātrāko laiku, lai to sasniegtu (sākumā tā ir tikai pietura A). Pēc tam var aprēķināt sasniegto pieturu laikus. Šis process jāatkārto ar visām pieturām, līdz tiek atrasts ātrākais laiks līdz galamērķim. Katrai pieturai var būt vairāki iespējamie ierašanās maršruti ar dažādiem laikiem, bet tiek ņemts vērā tikai ātrākais. Tikai tad, kad ir izmēģinātas visas ierašanās iespējas un atrasts ātrākais laiks, pieturu var izmantot, lai aprēķinātu citu no šīs pieturas sasniegto pieturu laikus. Šīs metodes pielietojums šajā uzdevumā ir aprakstīts tālāk.



Pareizais maršruts ir iezīmēts. Skaitļi blakus katrai pieturai norāda īsāko laiku, kas nepieciešams, lai nokļūtu pieturā no katra virziena.

Pieturu C var vienkārši sasniegt 1 minūtē, braucot ar pirmo oranžo autobusu. Tāpat pieturu F var sasniegt 5 minūtēs un pieturu G 8 minūtēs. Pieturu I var sasniegt no pieturas G, braucot ar trešo rozā autobusu (3 minūšu gaidīšana), kopā 8 + 3 + 5 = 16 minūtēs. Pieturu H var sasniegt no pieturas C (pēc ierašanās 1. minūtē) ar otro zaļo autobusu pēc 1 minūtes gaidīšanas un 7 minūšu brauciena, kopā 1 + 1 + 7 = 9 minūtēs no sākuma.

Pieturu H var sasniegt arī no pieturas F (pēc ierašanās 5. minūtē) ar otro zilo autobusu 1 minūtē pēc 1 minūtes gaidīšanas, kopā 5 + 1 + 1 = 7 minūtēs. Tas ir ātrāk nekā iepriekš aprēķinātās 9 minūtes, un šis jaunais skaitlis tiek izmantots turpmākajiem aprēķiniem. Ātrāka ierašanās nozīmē arī to, ka pirmo zaļo autobusu var izmantot, lai sasniegtu pieturu I kopumā 7 + 3 = 10 minūtēs. Jaunais laiks līdz pieturai I ir ātrāks nekā iepriekš aprēķinātās 16 minūtes, tāpēc 16 minūšu laiks tiek atmests un tālāk netiek izmantots. Tā kā tas ir ātrākais maršruts līdz I, to pašu autobusu var izmantot, lai sasniegtu pieturu J kopumā 10 + 2 = 12 minūtēs un pieturu K kopumā 12 + 1 = 13 minūtēs. Pieturu J var sasniegt arī no pieturas H 7 + 8 = 15 minūtēs. Tas ir lēnāk un tālāk netiek izmantots.

No pieturas I otro rozā autobusu var izmantot, lai sasniegtu pieturu B (galamērķi) pēc 1 minūtes gaidīšanas, kopā 10 + 1 + 9 = 20 minūtes. Šī vēl nav galīgā atbilde, jo pieturu B var sasniegt arī no pieturas J. To var izdarīt ar otro zilo autobusu pēc 3 minūšu gaidīšanas un 6 minūšu brauciena kopā 12 + 3 + 6 = 21 minūtē. Tas ir lēnāk nekā 20 minūtes, tāpēc iepriekš aprēķinātās 20 minūtes patiesībā arī ir pareizā atbilde.

Šī ir informātika

Autobusu maršruta karte sastāv no vairākām autobusu pieturām, kas savienotas ar vienvirziena autobusu maršrutiem. Autobusu maršrutiem ir atsevišķi segmenti starp divām pieturām, kuru šķērsošanai nepieciešams noteikts laiks. Šāda veida izkārtojums ir līdzīgs svarotam orientētam grafam. Pieturas tiek uzskatītas par virsotnēm, bet maršrutu segmenti – par šķautnēm. Virsotņu struktūru, kas savienotas ar šķautnēm, sauc par grafu. Tā kā maršruti ir vienvirziena, savienojumi starp virsotnēm (šķautnēm) ir vienvirziena, kas padara grafu orientētu. Un, tā kā katram maršruta segmentam ir atšķirīgs tā šķērsošanas laiks (vai šķautnes svars), grafs ir svarots. Vienīgā atšķirība ir tā, ka laiks no vienas pieturas līdz otrai ir atkarīgs ne tikai no attāluma, bet arī no ierašanās laika un autobusa braukšanas laikiem. Svarotam grafikam ir nemainīgi šķautņu svari, kas nav atkarīgi no ierašanās laika. Problēma ir atrast īsāko ceļu starp divām grafika virsotnēm. Šis uzdevums bieži sastopams dažādos informātikas uzdevumos un ir vienkāršota reālu maršrutu atrašanas algoritmu versija. Ir vairāki

Šādā grafā to var izdarīt vairākos veidos, taču atbilžu sadaļā aprakstītā metode patiesībā ir līdzīga dinamiskajai programmēšanai, kas ir problēmu risināšanas veids, vispirms risinot mazākas tā daļas un izmantojot atbildes, lai atrisinātu lielākas daļas, līdz tiek izveidota pilna atbilde. Šajā gadījumā mēs sākam, aprēķinot laiku līdz sākumam tuvākajām pieturām un turpinām līdz tālākajai pieturai – galamērķim.

Šī ir skaitļošanas domāšana

Šis uzdevums veicina skaitļošanas domāšanu, uzsverot datu interpretāciju, algoritmisko spriešanu un secību. Tas ietver strukturētu datu analīzi, tostarp autobusu atiešanas intervālus, ceļojuma laikus un pārsēšanās apstākļus. Lai atrastu pareizu risinājumu, ir svarīgi identificēt modeļus šajos skaitliskajos datos, izsekot vairākām sinhronizētām darbību virknēm un noteikt derīgas pārejas, pamatojoties uz iepriekš definētiem ierobežojumiem. Efektīva datu apstrāde nodrošina precizitāti lēmumu pieņemšanā.

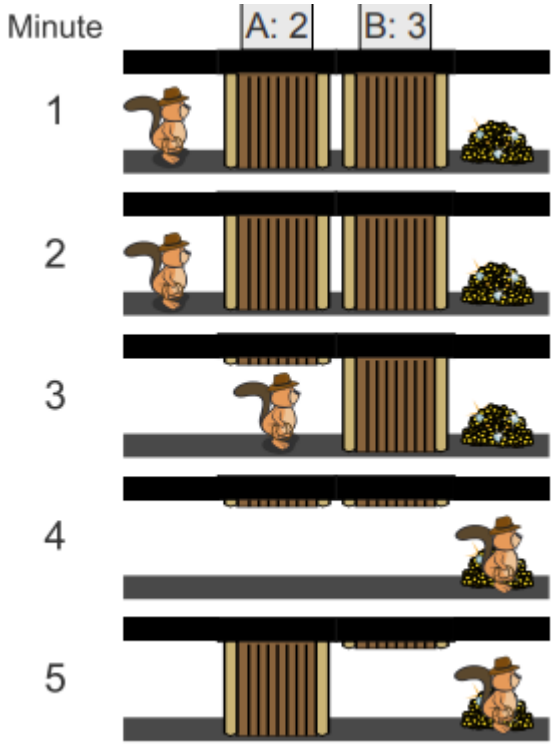
Algoritmiskā domāšana ir būtiska, jo uzdevuma risināšanai ir nepieciešams izveidot strukturētu soļu secību. Atkārtotu modeļu atpazīšana autobusu atiešanā, nosacītu noteikumu piemērošana derīgām pārsēšanās vietām un sistemātiska iespējamo maršrutu novērtēšana atspoguļo algoritmiskos pamatprincipus. Šī pieeja ir cieši saistīta ar reālās pasaules lietojumprogrammām, piemēram, plānošanas sistēmām un ceļa meklēšanas algoritmiem.

Sakārtojums ir ļoti svarīgs, jo ir jāuztur pareiza notikumu secība. Dažādu laika intervālu un ceļojuma ilguma saskaņošana strukturētā veidā nodrošina, ka risinājums atbilst loģiskai plūsmai. Integrējot datu interpretāciju ar algoritmisku spriešanu, uzdevums pastiprina to informātikas principu, kas ir būtiski automatizācijas, loģistikas un optimizācijas problēmām, pielietošanas praksi.

Bebrs Džons

Vācija

Bebrs Džons atrodas bīstamajā piramīdā, kurā ir daudz šausmīgu gaiteņu. Katra gaiteņa galā atrodas milzīgs dārgums. Džons vēlas pēc iespējas ātrāk sasniegt jebkuru dārgumu. Katru gaiteni nostiprina bloku rinda. Sākumā visi bloki ir nolaisti. Tiklīdz kāds nonāk gaitenī, tā bloki sāk periodiski kustēties. Bloks ar periodu 2 paceļas pēc 2 minūtēm, pēc vēl 2 minūtēm atkal nolaižas utt.



Skatīt piemēru kreisajā pusē. Šajā gaitenī ir divi bloki,
apzīmēti ar A un B, ar attiecīgi periodiem 2 un 3.

Attēlā parādīta gaitenī esošā situācija minūtēs
no 1 līdz 5 pēc Džona ierašanās.

Lai pēc iespējas ātrāk sasniegtu dārgumu, Džons gaida 2 minūtes, dodas uz bloku A, gaida vēl 1 minūti un pēc tam paiet garām blokam
B, lai pēc kopumā 3 minūtēm sasniegtu dārgumu.

Džons rīkojas tā, it kā sekotu šādai instrukciju virknei:

wait(2)
goto_block(A)
wait(1)
Goto_treasure

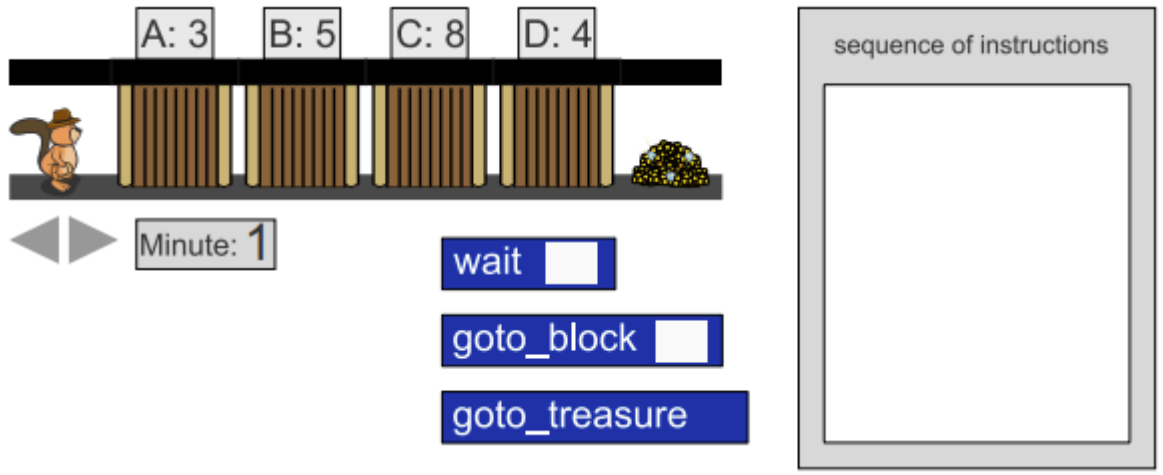
Džons varēja sasniegt dārgumu tādā pat laikā, izmantojot īsāku instrukciju virkni:

wait(3)
Goto_treasure

Nākamajā briesmīgajā gaitenī ir četri bloki ar periodiem 3, 5, 8 un 4.

Jautājums / Izaicinājums

Kāda ir īsākā instrukciju virkne, lai Džons sasniegtu dārgumu pēc iespējas ātrāk?



Salieciet šo instrukciju virkni labajā pusē. Aizpildiet instrukciju tukšumus ar pareizajām vērtībām. Paturiet prātā, ka (a) pāreja no viena bloka uz otru neaizņem laiku un (b) pāreja no viena bloka uz nākamo ir droša tikai tad, ja abi bloki ir pacelti.

Zem gaiteņa varat noklikšķināt uz pogām uz priekšu un atpakaļ, lai simulētu bloku kustību minūti pēc minūtes.

Atbilžu varianti / Interaktivitātes apraksts

Instrukcijām jābūt pieejamām kā *Blockly* stila blokiem, kurus var salikt secībā un kuros var pielāgot parametrus. Nepieciešamas šādas (lielākoties parametrizētas) instrukcijas: wait(n), goto_block(l), goto_treasure

Jābūt pieejamam arī interaktīvam rīkam bloku uzvedības simulēšanai. Lietotājam jābūt iespējai pārvietoties uz priekšu un atpakaļ pa minūtei un redzēt atbilstošos gaitenī/blokos esošos stāvokļus. Minūšu skaitītājs parāda, cik minūtes ir pagājušas.

Atbildes skaidrojums

Skatiet attēlu, lai redzētu, kā Džons varētu pārvietoties pa šo briesmīgo gaiteni. Lai pārvietotos pēc iespējas ātrāk, viņš var izpildīt šo instrukciju secību un sasniegt dārgumu pēc 12 minūtēm (oranžā līnija):

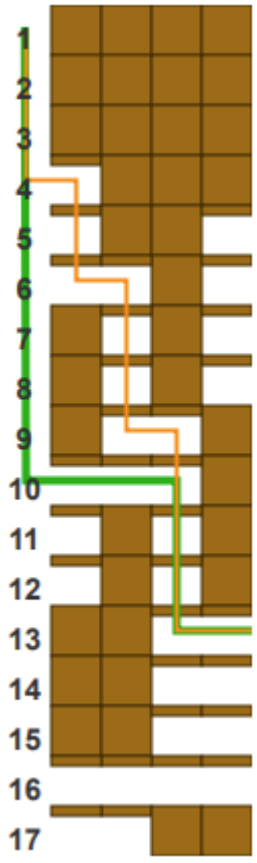
wait(3)
goto_block(A)
wait(2)

goto_block(B)
wait(3)
goto_block(C)
wait(4)
Goto_treasure

Tomēr viņš var izpildīt mazāk instrukciju un sasniegt dārgumu arī pēc 12 minūtēm (zaļā līnija):

wait(9)
goto_block(C)
wait(3)
Goto_treasure

Džons nekādi nevar izpildīt mazāk instrukciju , nezaudējot laiku. Viņš varētu izpildīt mazāk instrukciju tikai tad, ja vienā piegājienā steigtos cauri gaitenim. Vienīgā iespēja to izdarīt ir pēc 15 minūšu gaidīšanas. Tad viņš var sasniegt dārgumu tikai ar diviem norādījumiem – bet pēc 15 minūtēm, kas nav *pēc iespējas ātrāk*.



Šī ir informātika

Šis uzdevums aptver trīs svarīgus informātikas aspektus:

Algoritms

Algoritma jēdziens, iespējams, ir datorzinātnes centrālais jēdziens; tas koncentrējas uz jautājumu par to, kā var atrisināt problēmu un līdz ar to, kā risinājumu var nepārprotami aprakstīt. Algoritma atrašana prasa radošumu, atšķirībā no algoritma izpildes: Kad algoritms ir precīzi definēts, piemēram, kā programmēšanas valodas instrukciju virkne, dators var sekot instrukcijām un izpildīt tās bez jebkādas izpratnes.

Optimizācija

Problēma

Šis Bebras uzdevums ietver dažādu pieejas problēmai apsvēršanu, to salīdzināšanu savā starpā un labākās pieejas izvēli atbilstoši kādam kvalitātes rādītājam. Datorzinātnē šāda veida problēmas sauc par optimizācijas problēmām, jo mērķis ir līdz minimumam samazināt noteiktu faktoru (šajā gadījumā laiku, kas nepieciešams Džonam, lai sasniegtu dārgumu, kā arī instrukciju skaitu, ko viņš izpilda). Viena no metodēm, ko var izmantot optimizācijas problēmu risināšanai, ir dinamiskā programmēšana. Šī pieeja ietver problēmas sadalīšanu daļās un tad lielāku sākotnējās problēmas daļu risināšanu, izmantojot saglabātus daļējos risinājumus. Šeit daļējas problēmas varētu būt tikai pirmā, otrā, trešā vai ceturtā bloka sasniegšana, līdz beidzot tiek atrasts ceļš uz dārgumu.

Programmēšana

Lai algoritmus varētu izpildīt datori, tie ir jāizsaka kādā programmēšanas valodā. Šāda valoda nodrošina pamatdarbību (vai komandu) kopu, piemēram, šajā gadījumā komandas 'wait(n)' vai 'goto_block(b)'. Sarežģītu algoritmu var aprakstīt, apvienojot pamatkomandas ar valodas konstrukcijām, kas ļauj veikt secīgu, atkārtotu un nosacījumu vadītu izpildi.