Lietuvos mokinių informatikos olimpiada

2012 m. gruodžio 14 d.

Uždavinių sprendimai

Lokali dvejetainė struktūra (teorinis uždavinys VIII-IX klasėms). Įsivaizduokime LDS kodo bitų eilutę sudarytą vien tiktai iš 0. Optimalų LDS kodą gausime tada ir tik tada, kai kažkurioje eilutės vietoje dalį nulių pakeisime nepertraukiama vienetų eilutę. Pavyzdžiui, kai $K = 4$ (bandykime nulius keisti į vienetus pradedami nuo 2 pozicijos), LDS kodą `0000` galime keisti į `0100`, `0110` bei `0111` ir tokiu būdų gausime $K - 1 = 3$ optimalius LDS kodus. Kadangi LDS eilutė yra ciklinė mes galime daryti pakeitimus nesirūpindami, kad vienetų eilutė netilps į pradinį kodą. Tai reiškia, kad pradėdami nuo bet kurios pozicijos, o tokių pozicijų yra lygiai K, mes galime atlikti $K - 1$ pakeitimą ir kiekvieną kartą gausime naują optimalų kodą. Taigi, bet kokiam K, egzistuoja $K * (K - 1)$ skirtingų optimalių kodų.

Atsakymas. $32 \times 31 = 992$.

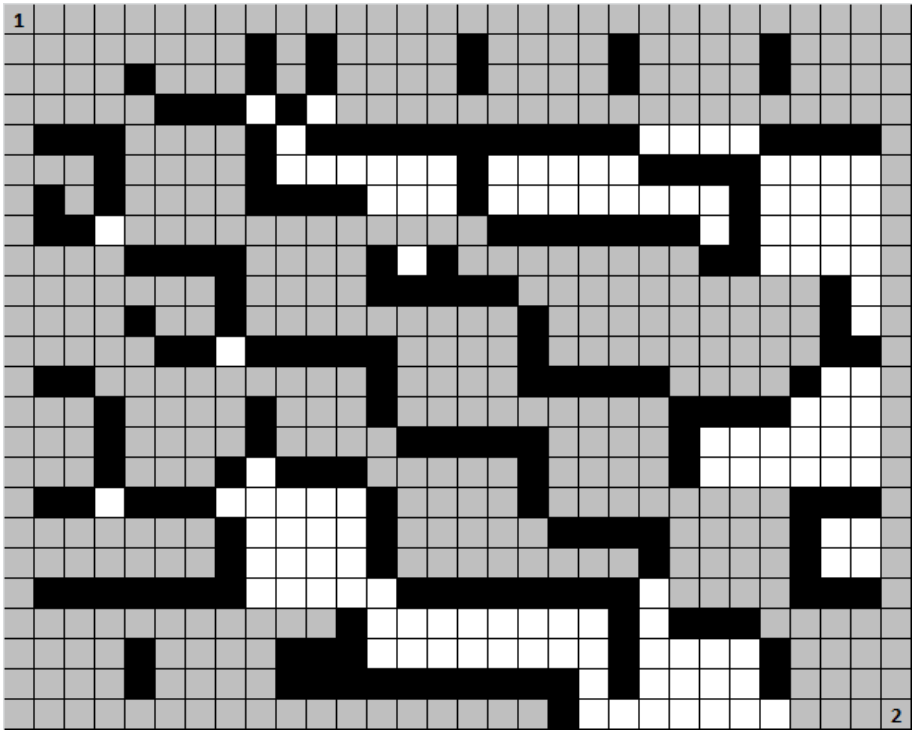
Lokali dvejetainė struktūra (uždavinys VIII-IX klasėms). Norėdami patikrinti, ar LDS kodas (duotas dešimtainėje sistemoje) yra optimalus reikia jį pasiversti į dvejetainę sistemą ir patikrinti, kiek kartų gretimi bitai skiriasi. Reikia nepamiršti dviejų dalykų — LDS kodas yra ciklinis ir jis turi lygiai K bitų. Žemiau pateikta programa tai panaudodama ir aptikrina, ar duotas LDS kodas yra optimalus:

```
 1  read k
 2  read n
 3  kartai ← 0
 4  ankstesnis ← n div 2^(k−1)
 5  for i ← 1 to k
 6        do
 7           dabartinis ← n mod 2
 8           n ← n div 2
 9           if ankstesnis ≠ dabartinis
10              then
11                   kartai ← kartai +1
12                   ankstesnis ← dabartinis
13  if kartai = 2
14     then print "TAIP"
15     else  print "NE"
```

NRobotas (teorinis uždavinys X-XII klasėms). Pirmiausiai pilkai pažymėkime tuos langelius, kuriuos gali pasiekti robotas (1 pav.). Tada atmeskime tuos langelius, iš kurių robotas negali pasiekti tikslo (2 pav.).

Matome, kad susidaro trys platūs "keliai": pirmasis eina per stačiakampio įstrižainę, o antrasis driekiasi į rytus iki kambario kampo ir leidžiasi žemyn iki tikslo.

Pirmasis yra sudarytas iš 7 nuosekliai sujungtų 4×4 dydžio salelių, o tarp gretimų salelių egzistuoja lygiai vienas perėjimas. Tai reiškia, kad jeigu įveikti vieną salelę egzistuoja N būdų, tai kiekvieną iš jų gretimoje salelėje galima pratęsti dar N skirtingų būdų ir t.t., kol bus pasiektas tikslas. Taigi, šiuo "keliu" nukeliauti iš vieno kambario kampo iki kito kambario



1 pav.: Pilka spalva pažymėti visi langeliai, kuriuos gali pasiekti robotas.

kampo galima N^7 būdų.

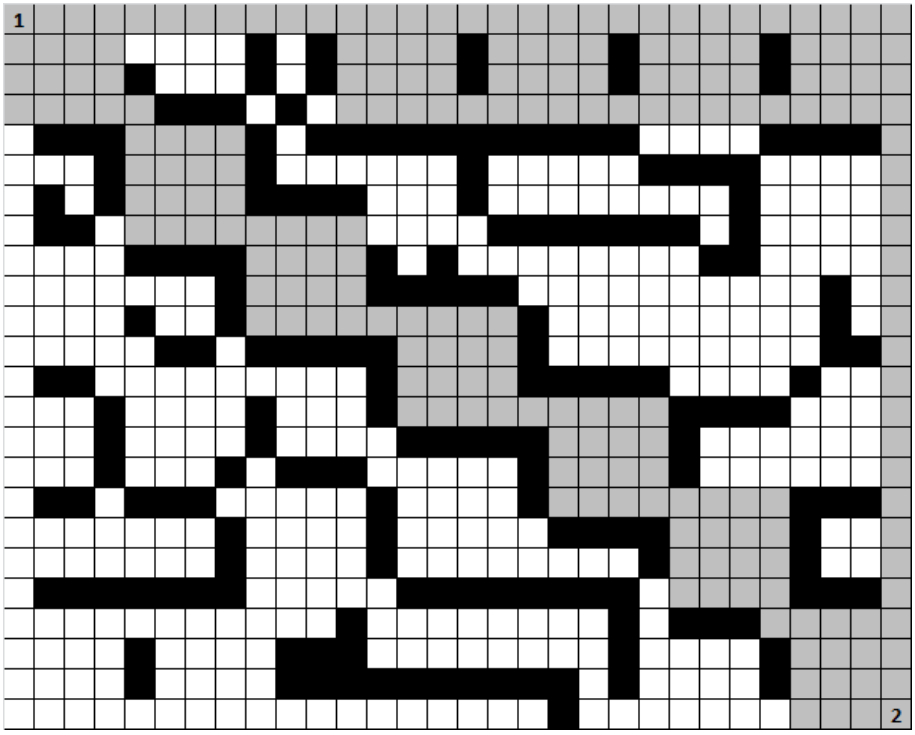
Antrasis yra sudarytas iš 4 lygiagrečiai sujungtų 4×4 dydžio salelių, o iš kiekvienos salelės į gretimą galima patekti keliaujant per dvi perėjas, iki vienos jų nusigauti galima vienu būdu, o iki kitos — N būdų, nes reikia pereiti į priešingą nagrinėjamos salelės kampą. Taigi, įveikti 4 tokias saleles galima $4 * N$ būdais. Viską susumuojame ir gauname, kad pereiti iš pradinio kambario kampo iki jam priešingo galima $N^7 + 4 * N$ būdais.

Beliko išsiaiškinti, kam lygus N. Nagrinėkime 4×4 dydžio salelę, kurią bandome įveikti nuo viršutinio kairiojo kampo iki apatinio dešiniojo. Pasižymėkime langelius, iki kurių nukeliauti galima tik vienu būdu. Į kiekvieną iš neužpildytų langelių galima ateiti tik iš viršaus ir iš kairės, taigi susumuojame juose įrašytus skaičius ir turime atsakymą, keliais būdais galima ateiti iki nagrinėjamo langelio (žr. 3 pav.). Gauname, kad 4×4 dydžio salelę galima įveikti 20 skirtingų būdų.

Iš tiesų, šį metodą galime pritaikyti ir visam kambario žemėlapiui. Užpildykime pirmąją eilutę ir pirmąjį stulpelį vienetais, o kiekvieną kitą langelį $x_{i,j}$ pildykime iš viršaus pagal formulę:

$$x_{i,j} = \begin{cases} 0, & \text{jeigu šiame langelyje yra kliūtis;} \\ x_{i-1,j} + x_{i,j-1}, & \text{priešingu atveju.} \end{cases} \quad (1)$$

Atsakymas. $N^7 + 4 * N = 20^7 + 4 * 20 = 1280000080$.



2 pav.: Pilka spalva pažymėti tik tie langeliai, iš kurių robotas gali nukeliauti iki tikslo.

1	1	1	1
1			
1			
1			

1	1	1	1
1	2		
1			
1			

1	1	1	1
1	2	3	4
1	3	6	
1	4		

1	1	1	1
1	2	3	4
1	3	6	10
1	4	10	20

3 pav.: Kiekviename langelyje įrašyta keliais būdais čia galima patekti iš viršutinio kairiojo kampo.

Filmukai (uždavinys VIII-IX klasėms). Šio uždavinio sprendimo užuomina duota pavyzdžio paaiškinime – reikia rasti pertraukų trukmes (kurių yra $M-1$) ir į jas iš eilės bandyti talpinti filmukus. Jeigu į likusį pertraukos laiką filmukas nebetelpa, to laiko perkelti į kitą pertrauką negalima, nes Jaunėlis filmukus nori žiūrėti be pertrūkių. Toliau pateiktas pseudokodo fragmentas iliustruoja algoritmą (atsakymas išsaugomas kintamajame *visoPerziureta*).

```
1  visoPerziureta ← 0
2  kitasFilmukas ← 1
3  for i ← 1 to M −1
4        do pertraukosTrukme ← S_{i+1} − E_i
5           while kitasFilmukas ≤ N and L_{kitasFilmukas} ≤ pertraukosTrukme
6                 do pertraukosTrukme ← pertraukosTrukme − L_{kitasFilmukas}
7                    kitasFilmukas ← kitasFilmukas +1
8                    visoPerziureta ← visoPerziureta +1
```

Filmukai (uždavinys X-XII klasėms). Šioje uždavinio versijoje Jaunėlis filmukus gali žiūrėti su pertrūkiais, todėl individualios pertraukos nėra svarbios — svarbi tik jų trukmių suma. Ją galima suskaičiuoti taip:

```
1  laisvasLaikas ← 0
2  for i ← 1 to M −1
3        do laisvasLaikas ← laisvasLaikas + (S_{i+1} − E_i)
```

Uždavinys supaprastėja — dabar tiesiog reikia į fiksuotą laiką sutalpinti kuo daugiau filmukų. Nesunku pastebėti, kad didžiausią filmukų skaičių gausime tuomet, kai žiūrėsime trumpiausius filmukus. Taigi, reikia rasti, kelių trumpiausių filmukų trukmių suma neviršija turimo laisvo laiko. Šis pseudokodo fragmentas užbaigia algoritmą:

```
4  Rikiuoti(L)                ▷ Filmukų trukmės surikiuojamos nedidėjimo tvarka
5  perziuretaFilmuku ← 0
6  for i ← 1 to N
7        do if L_i ≤ laisvasLaikas
8              then laisvasLaikas ← laisvasLaikas − L_i
9                   perziuretaFilmuku ← perziuretaFilmuku + 1
```

Kryžiukai-nuliukai (uždavinys X-XII klasėms). Šiame uždavinyje turime nustatyti, kokiais atvejais lentelė yra sugadinta, tad apibrėžkime, kas yra teisinga lentelė. Iš sąlygos:

> „…du žaidėjai paeiliui deda kryžiukus ir nuliukus (pradeda kryžiukai).“

Pirmas pastebėjimas — teisingoje lentelėje kryžiukų skaičius visada bus arba lygus nuliukų skaičiui, arba vienetu didesnis. Teiginio teisingumu nesunku įsitikinti įsivaizduojant kaip kinta simbolių skaičius lentelėje žaidėjams darant ėjimus. Taigi, jeigu ši sąlyga duotai lentelei negalioja, iš karto žinome, kad lentelė yra sugadinta. Taip pat pastebėkime, kad teisingoje lentelėje visuomet galime nustatyti, kuris žaidėjas padarė paskutinį ėjimą — tai bus žaidėjas X, jei kryžiukų skaičius yra vienetu didesnis nei nuliukų, arba žaidėjas O, jeigu kryžiukų skaičius yra lygus nuliukų skaičiui. Skaitome toliau:

> „Laimi pirmas žaidėjas, iš savo simbolių sudaręs eilutę, stulpelį arba įstrižainę. Jei užpildoma visa lentelė ir nė vienas žaidėjas nelaimėjo, tuomet yra lygiosios. Žaidimas baigiamas kai kuris nors žaidėjas laimi, arba pasiekiamos lygiosios.“

Nustatyti, ar žaidėjas yra sudaręs bent vieną eilutę, stulpelį arba įstrižainę, nesunku. Yra 12 atvejų (5 eilutės, 5 stulpeliai, 2 įstrižainės), kuriuos galima tikrinti su 12 sąlygų arba ciklu. Gana kompaktiškas būdas yra pateiktas pavyzdiniame sprendime. Taigi, laikykime, kad jau turime parašę funkciją ArLaimi(Ž), kuri pasako, ar žaidėjas Ž yra sudėjęs bent vieną eilutę, stulpelį arba įstrižainę.

Lengviausias atvejis yra kai abu žaidėjai yra laimintys (= ArLaimi(X) ∧ ArLaimi(O)) — tuomet lentelė yra tikrai sugadinta. Jeigu nė vienas žaidėjas nėra laimintis (= ¬ArLaimi(X)∧

¬ArLaimi(O)), turime du atvejus. Jei tuščių langelių nėra, tuomet yra lygiosios, o jei yra bent vienas tuščias langelis, žaidimas yra nebaigtas.

Liko išnagrinėti atvejus, kai lygiai vienas žaidėjas yra laimintis. Kadangi žinome, kuris žaidėjas paėjo paskutinis, galime patikrinti, ar laimi tas pats žaidėjas. Jei ši sąlyga negalioja — lentelė sugadinta.

Atrodytų — atsakymas jau aiškus, bet dar liko viena smulkmena. Kas, jeigu žaidėjas yra sudaręs dvi laiminčias eilutes? Šiuo atveju, lentelė yra sugadinta, nes žaidimas turėjo pasibaigti po pirmos sudarytos eilutės. Sugalvoti šios problemos sprendimą kiek sunkiau, tačiau prisiminkime, kad žaidimas baigiasi iškart, kai kuris nors žaidėjas tampa laimintis. Tai reiškia, kad prieš paskutinį ėjimą nė vienas žaidėjas nebuvo laimintis. Deja, paskutinio ėjimo nežinome, tad turime išbandyti visus galimus variantus. Jeigu nėra tokio ėjimo, kurį atšaukus žaidėjas taptų nelaimintis, lentelė vėlgi yra sugadinta, o jeigu toks ėjimas yra, pagaliau galime pasakyti, kad žaidėjas, sudaręs laiminčią eilutę, stulpelį arba įstrižainę, ir yra žaidimo laimėtojas.

Pseudokodo čia nepateiksime, nes algoritmo veikimą lengviau suprasti skaitant pavyzdinio sprendimo kodą.