Lietuvos mokinių informatikos olimpiada

Respublikinis etapas • 2011 m. vasario 4 d.

Uždavinių sprendimai

Limonadas (uždavinys 8-12 klasėms). Prieš sprendžiant uždavinį reikia atkreipti dėmesį, kad akcijos metu galima pirkti ne tik akcijines pakuotes po 6 butelius limonado, bet ir atskirus limonado butelius įprasta kaina.

Jeigu Algirdas perka mažiau nei penkis butelius, jam neapsimoka naudotis akcija, nes sumokės 20 litų vietoje 0, 4, 8, 12 ar 16.

Jeigu Algirdas perka lygiai 5 butelius, jis išleis tiek pat pinigų naudodamasis akcija, kaip ir nesinaudodamas. Nors pirmuoju atveju jis nusipirks vienu buteliu daugiau, uždavinio sąlyga nieko apie tai nekalba, tad galime laikyti, kad ir perkant 5 butelius naudotis akcija Algirdui neapsimoka.

Taigi Algirdas sutaupo tik tada, kai perka 6 ar daugiau butelių. Vietoje kiekvieno butelių šešeto pirkdamas pakuotę, jis sutaupys po 4 litus. Jei perkamų butelių skaičius nesidalo iš šešių be liekanos (pvz. perka 7 ar 8 butelius), tokiu atveju likusius butelius jam apsimoka pirkti po vieną.

Jei i-ąją dieną Algirdas pirko b_i butelių limonado, tai pasinaudodamas akcija galėjo (b_i **div** 6) šešetų pakeisti pakuotėmis ir sutaupyti $(b_i \textbf{ div } 6) \cdot 4$ litus.

```
sutaupo ← 0
read N                ▷ dienų skaičius
for i ← 1 to N
      do read b
         sutaupo ← sutaupo + (b div 6) · 4
print sutaupo
```

Laisvadieniai (uždavinys 8-9 klasėms). Akivaizdu, kad Harbodolio planetos kalendorius yra gerokai paprastesnis negu Žemės, kurioje metai gali turėti skirtingą dienų skaičių ir šventės vyksta ne visuomet tomis pačiomis dienomis.

Numeruokime, kaip prideda informatikams, savaitės dienas nuo 0 iki $w-1$ (kur $w = d + s$), metų dienas — nuo 0 iki $m-1$, o metus — nuo 0 iki $n-1$. Tokia numeracija leis supaprastinti kai kurias išraiškas.

Pavyzdžiui, kadangi pirmieji (o pagal mus — nuliniai) metai prasideda nuline savaitės diena, tai praėjus x dienų, savaitės dienos numeris bus paprasčiausiai $x \textbf{ mod } w$. Nesunkiai galime pasakyti, kokia savaitės diena prasidės y-ieji metai. Kadangi bus praėję lygiai $y \cdot m$ dienų, tai savaitės diena bus $(y \cdot m) \textbf{ mod } w$. Panašiai patikriname, ar y-ųjų metų i-oji diena yra laisvadienis:

```
LAISVADIENIS(y, i)
if svente[i] or (y · m + i) mod w ≥ d      ▷ šventinė diena arba savaitgalis?
   then return TRUE
   else return FALSE
```

1-oje eilutėje patikrinama, ar i-oji metų diena yra šventinė — tam naudojamas loginis masyvas *svente*, užpildytas pagal pradinius duomenis. Savaitės diena yra savaitgalis, jeigu jos numeris nemažesnis negu d.

Norėdami suskaičiuoti, kiek laisvadienių yra konkrečiuose metuose, tiesiog iškviečiame funkciją Laisvadienis kiekvienai tų metų dienai:

```
LaisvadieniųMetuose(y)
kiek ← 0
for i ← 0 to m − 1
      do if Laisvadienis(y, i)
            then kiek ← kiek +1
return kiek
```

Visus laisvadienius per n metų, galime suskaičiuoti iškvietę LaisvadieniųMetuose kiekvieniems metams ir susumavę rezultatus:

```
LaisvadieniųIšViso
kiek ← 0
for y ← 0 to n − 1
      do kiek ← kiek +LaisvadieniųMetuose(y)
return kiek
```

Tai yra teisingas sprendimas, tačiau jis nepakankamai efektyvus, kad įveiktų visus uždavinio testus per nurodytą laiką. Šiame sprendime kiekvienai nagrinėjamo laikotarpio dienai tikrinama, ar ta diena bus laisvadienis. Taigi nagrinėjamų dienų bus $n \cdot m$, šis skaičius gali siekti iki 1 000 000 000. Užrašant formaliai, šio algoritmo sudėtingumas yra $O(n \cdot m)$.

Norint efektyviau spręsti uždavinį, reikia pastebėti svarbią Harbodolio kalendoriaus savybę: laisvadienių skaičius metuose priklauso tik nuo to, kokia savaitės diena tie metai prasideda. Pavyzdžiui, jei savaitė turi tris dienas, o metus sudaro dešimt dienų, tai pirmieji ir ketvirtieji metai turės tiek pat laisvadienių, nes abu prasideda ta pačia savaitės diena. Ir tai nepriklauso nuo šventinių dienų kalendoriaus, kadangi jis kasmet toks pat.

Kadangi skirtingų savaitės dienų yra tik w, tai vėliausiai po w metų nauji metai vėl prasidės nuline savaitės diena. Tarkime, kad praėjo lygiai y metų kol tai nutiko, tuomet per tolesnius y metų bus lygiai tiek pat laisvadienių, kaip ir per pirmuosius y. Pakeiskime funkciją LaisvadieniųIšViso, kad aptikus šį periodiškumą, visų periodų laisvadieniai būtų susumuojami iš karto:

LaisvadieniųIšViso

```
1  kiek ← 0
2  y ← 0
3  while y ≤ n
4        do kiek ← kiek + LaisvadieniųMetuose(y)
5           y ← y + 1
6           if (y · m) mod w = 0                ▷ metai vėl prasideda nuline diena
7              then kartojasi ← n div y         ▷ kiek yra tokių periodų?
8                   kiek ← kiek · kartojasi     ▷ susumuojami visi laisvadieniai
9                   y ← y · kartojasi           ▷ praleidžiami visi periodo metai
10 return kiek
```

Šis sprendimas yra gerokai efektyvesnis. Ciklas (eilutės 3–9) iš pradžių kartojamas tol, kol nauji metai vėl prasidės nuline savaitės diena (sąlyga 6-oje eilutėje). Tuomet žinoma, kiek laisvadienių yra y metų periode, ir greitai suskaičiuojami beveik visi laisvadieniai. Tik likę keli metai, kurie nebesudaro pilno periodo, vėl išnagrinėjami ciklu. Eilutės 3–9 vykdomos ne daugiau negu $2w$ kartų. Tad bendras šio algoritmo sudėtingumas yra $O(m \times (d+s))$.

Ilgintuvas (uždavinys 8-12 klasėms). Išrikiuokime kištukus jų pločių didėjimo tvarka ir pažymėkime išrikiuotų kištukų numerius $j_1, j_2, ..., j_N$. Tada $l_{j_1} \le l_{j_2} \le ... \le l_{j_N}$.

Pabandykime sudėti kištukus į ilgintuvą tokia tvarka:

$$j_N, j_1, j_{N-1}, j_2, j_{N-2}, j_3, ... \qquad (1)$$

Parodysime, kad šis išdėstymas optimalus.

Pažymėkime D mažiausią plotį tarp gretimų ilgintuvo lizdų centrų, tokį, kad kištukai (1) pateiktu išdėstymu dar sutilptų į ilgintuvą. Galime įsivaizduoti, kad pirma sudedame kištukus tvarka (1), o tada siauriname tarpą tarp ilgintuvo lizdų nuo d, kol kažkurie du gretimi kištukai susiglaudžia. Naują atstumą tarp lizdų centrų žymime D. Taigi kažkokiam $k \le N/2$ galioja $l_{j_{N-k+1}} + l_{j_k} = D$ (dviejų gretimų kištukų pločių suma lygi D).

Pavadinkime kištukus, kurių pločiai didesni arba lygūs $l_{j_{N-k+1}}$, *plačiais*, o kurių pločiai mažesni už l_{j_k} — *siaurais*. Atkreipkime dėmesį, kad plačių kištukų yra ne mažiau kaip k, o siaurų kištukų yra ne daugiau nei $k-1$.

Dabar pasinaudosime prieštaros metodu įrodyti savo teiginiui. Sakykime, kad aukščiau aprašytas išdėstymas nėra optimalus. Tai reikštų, jog egzistuoja geresnis kištukų išdėstymas, kai kištukai telpa į ilgintuvą, kuriame atstumai tarp gretimų lizdų centrų yra $D' < D$ (dar mažesnis).

Šalia plačių kištukų, kurių yra bent k, gali būti tik siauri, todėl gretimų plačių–siaurų kištukų porų yra ne mažiau kaip $2k-2$ (visi išskyrus kraštinius turi 2 gretimus kištukus). Bet siaurų kištukų yra ne daugiau kaip $k-1$, todėl iš abiejų siauro kištuko pusių turi būti platus.

Gauname vienintelį galimą išsidėstymą, kur kas antras kištukas yra siauras, kas antras platus. Bet $2k-1 < N$, taigi yra kištukų, kurie nėra nei siauri, nei platūs — jiems neliko vietos, nes jie netilps šalia plačių kištukų. Gavome prieštarą.

Taigi, išdėstymas, pateiktas (1), yra optimalus. Kadangi Linas buvo sudėjęs kištukus į ilgintuvą, tai $D \leq d$ ir mes galime juos sukišti atgal šia tvarka.

```
read N                              ▷ kištukų skaičius
read d                              ▷ atstumas tarp gretimu lizdų centrų
for i ← 1 to N
     do
        read l[i]                   ▷ kištuko plotis
        j[i] ← i                    ▷ kištuko numeris
išrikiuoti l kartu su j
for i ← 1 to N
     do
        if n mod 2 = 1
          then print j[i div 2 + 1]
          else  print j[N − i div 2]
```

Neskaitant rikiavimo, sprendimas yra tiesinis, taigi sprendimo efektyvumas priklauso tik nuo rikiavimo efektyvumo. Efektyviausias galimas sprendimas yra $O(N \log N)$.

Šis uždavinys buvo pateiktas abiems amžiaus grupėms. Skyrėsi tik ribojimai. Vyresniųjų grupės dalyviams, norint surinkti visus taškus, reikėjo parašyti efektyviausią galimą sprendimą. Tuo tarpu jaunesniems dalyviams užteko parašyti ir mažiau efektyvų ($O(N^2)$) sprendimą ir buvo galima surinkti visus taškus už testus.

Raidės (uždavinys 10–12 klasėms). Žemiau pateikiamą uždavinio sprendimą iliustruosime konkrečiu (sąlygos) pavyzdžiu.

Tegul S_M ($2 \leq M \leq N$) žymi grožio įvertį su parinkta M reikšme.

Perrinksime visas sveikas M reikšmes nuo 2 iki K, kur K yra tam tikras nedidelis sveikasis skaičius, kurį apskaičiuosime vėliau. Tai padarę žinosime, su kuria M reikšme S_M didžiausias.

Pasirinkime tam tikrą M reikšmę ($2 \leq M \leq K$). Sunumeruokime stulpelius eilės tvarka nuo 0 iki $M - 1$.

Sakykime, kad nagrinėjamame pavyzdyje pasirinkome $M = 4$. Sunumeravę gausime:

0	1	2	3
s	v	e	i
k	i	n	u
s	u	g	i
m	t	a	d
i	e	n	i
u			

Iš pradžių parinkime $S_M = 0$. Nagrinėkime kiekvieną sveikinimo raidę paeiliui.

Visuomet saugokime $A(s, r)$ ($0 \leq s \leq M - 1$, $r = \mathbf{a}, \mathbf{b}, \ldots, \mathbf{z}$), parodantį, kiek raidžių r jau priskyrėme į stulpelį s.

Apskaičiuoti $A(s, r)$ yra paprasta. Iš pradžių parinkime $A(s, r) = 0$ su visais s ir r. Kai nagrinėjame n–tąją sveikinimo raidę ($0 \leq n \leq N-1$), raskime n dalybos iš M liekaną s_n ($0 \leq s_n \leq M-1$). Kadangi n–toji raidė bus užrašyta stulpelyje s_n, padidiname $A(s_n, r_n)$ vienetu, kur r_n žymi n–tąją sveikinimo raidę, o kitų $A(s, r)$ nekeičiame.

Tiesa, prieš padidindami $A(s_n, r_n)$ vienu, prie S_M pridedame $A(s_n, r_n)$, nes panaudodami n–tąją raidę galime sudaryti $A(s_n, r_n)$ vienodų raidžių porų stulpelyje s_n.

Tokiu būdu pasirinktai M reikšmei galime apskaičiuoti S_M, atlikdami $O(N)$ operacijų.

Pažiūrėkime, kaip tai pritaikytume nagrinėjamam pavyzdžiui. Primename, kad jame $M = 4$. Iš eilės imame sveikinimo raides:

- $r_0 = \mathbf{s}$: $0 \bmod M = 0$, taigi $s_0 = 0$, $S_M \leftarrow S_M + A(0, \mathbf{s}) = 0$ ir $A(0, \mathbf{s}) \leftarrow 1$;
- $r_1 = \mathbf{v}$: $1 \bmod M = 1$, taigi $s_1 = 1$, $S_M \leftarrow S_M + A(1, \mathbf{v}) = 0$ ir $A(1, \mathbf{v}) \leftarrow 1$;
- $r_2 = \mathbf{e}$: $2 \bmod M = 2$, taigi $s_2 = 2$, $S_M \leftarrow S_M + A(2, \mathbf{e}) = 0$ ir $A(2, \mathbf{e}) \leftarrow 1$;
- $r_3 = \mathbf{i}$: $3 \bmod M = 3$, taigi $s_3 = 3$, $S_M \leftarrow S_M + A(3, \mathbf{i}) = 0$ ir $A(3, \mathbf{i}) \leftarrow 1$;
- $r_4 = \mathbf{k}$: $4 \bmod M = 0$, taigi $s_4 = 0$, $S_M \leftarrow S_M + A(0, \mathbf{k}) = 0$ ir $A(0, \mathbf{k}) \leftarrow 1$;
- $\cdots$
- $r_8 = \mathbf{s}$: $8 \bmod M = 0$, taigi $s_8 = 0$, $S_M \leftarrow S_M + A(0, \mathbf{k}) = 1$ ir $A(0, \mathbf{s}) \leftarrow 2$;
- $\cdots$

Žinome, kaip atliekant $O(N)$ operacijų bet kuriam M apskaičiuoti S_M. Tad jei perrinktume visus galimus M nuo 2 iki N turėtume algoritmą, kurio sudėtingumas $O(N^2)$. Tačiau pamėginkime įrodyti, kad visų galimų N perrinkti nereikia.

Pradžioje žadėjome, kad ieškosime tokio K, kad užteks perrinkti M reikšmes nuo 2 iki K. Tad tolesnis žingsnis – apskaičiuoti, koks mažas gali būti K.

Tiksliau, raskime tokį K, kad su visais $M > K$ galioja $S_M < S_2$, t. y. visiems M didesniems nei K grožio įvertis bus mažesnis už grožio įvertį, kai tekstas užrašytas dviejuose stulpeliuose. Kadangi S_M neturi prasmės, kai $M > N$, galime tarti, jog $K \leq N$.

Tegul $\mathcal{R} = \{\mathbf{a}, \mathbf{b}, \ldots, \mathbf{z}\}$ žymi raidyną — lotyniškas mažąsias raides. Jį sudaro 26 raidės, tai yra $|\mathcal{R}| = 26$. Tegul raidė r ($r \in \mathcal{R}$) kartojasi sveikinime k_r kartų. Tuomet

$$\sum_{r \in \mathcal{R}} k_r = N. \tag{2}$$

Iš pradžių norime aprėžti S_2 iš apačios, t. y. rasti už kokią reikšmę S_2 yra nemažesnė. Užrašykime sveikinimą dviem stulpeliais (kitaip sakant, parinkime $M = 2$). Sakykime, kad raidė r pirmame stulpelyje kartojasi a_r, o antrame — b_r kartų. Aišku, kad

$$a_r + b_r = k_r. \tag{3}$$

Iš raidžių r pirmame stulpelyje galime sudaryti $a_r(a_r-1)/2$ porų, o antrame — $b_r(b_r-1)/2$ porų. Taigi

$$S_2 = \sum_{r\in\mathcal{R}} \frac{1}{2}\left(a_r(a_r-1)+b_r(b_r-1)\right) \tag{4}$$

$$= \frac{1}{2}\sum_{r\in\mathcal{R}}\left(a_r^2+b_r^2\right) - \frac{1}{2}\sum_{r\in\mathcal{R}} k_r \tag{5}$$

$$= \frac{1}{2}\sum_{r\in\mathcal{R}}\left(a_r^2+b_r^2\right) - \frac{N}{2}. \tag{6}$$

Iš (3) žinome, kad $a_r+b_r=k_r$, todėl panaudoję (2) gauname $\sum_{r\in\mathcal{R}}(a_r+b_r)=\sum_{r\in\mathcal{R}}k_r=N$.

Kadangi realiųjų skaičių, kurių suma žinoma, kvadratų suma mažiausia tokiu atveju, jei tie skaičiai vienodi (šis teiginys vadinamas *kvadratinio–aritmetinio vidurkio savybe*), suma $\sum_{r\in\mathcal{R}}\left(a_r^2+b_r^2\right)$ būtų mažiausia, jei visi a_r ir b_r (visoms raidėms r) būtų lygūs, tai yra jei $a_r=b_r=\frac{N}{2|\mathcal{R}|}=\frac{N}{52}$. Tai reiškia, kad

$$S_2 \geq \frac{1}{2}\cdot|\mathcal{R}|\cdot 2\cdot\left(\frac{N}{52}\right)^2 - \frac{N}{2} \tag{7}$$

$$= \frac{N^2}{104} - \frac{N}{2}. \tag{8}$$

Nelygybė (8) aprėžia S_2 iš apačios. Dabar norime aprėžti S_M iš viršaus, kai M didelis. Parinkime $M>K$. Tuomet sveikinimą sudarys M stulpelių, iš kurių kiekviename bus po mažiau nei $N/M+1$ raidė. S_M reikšmė būtų didžiausia, jei kiekviename stulpelyje visos raidės būtų vienodos. Taigi peržiūrėjęs bet kurį vieną stulpelį Vilkas padidins grožio įvertį mažiau nei $\frac{1}{2}\cdot\frac{N}{M}\cdot\left(\frac{N}{M}+1\right)$. Tai reiškia, kad

$$S_M < M\cdot\frac{1}{2}\cdot\frac{N}{M}\cdot\left(\frac{N}{M}+1\right) \tag{9}$$

$$= \frac{N}{2}\cdot\left(\frac{N}{M}+1\right) \tag{10}$$

$$< \frac{N^2}{2K} + \frac{N}{2}. \tag{11}$$

Sugretinę nelygybes (8) ir (11) matome, kad galime parinkti bet kokią K reikšmę, kuri tenkina

$$\frac{N^2}{2K} + \frac{N}{2} \leq \frac{N^2}{104} - \frac{N}{2}, \tag{12}$$

tai yra

$$K \geq \frac{N^2}{2\left(\frac{N^2}{104}-N\right)} \tag{13}$$

$$= \frac{52}{1-\frac{104}{N}}. \tag{14}$$

Panagrinėję dešinę nelygybės (14) pusę matome, kad jei $N \geq 156$, tai $\frac{52}{1-104/N} \leq 156$, todėl galime parinkti $K = \min\{N, 156\}$.

Kadangi viso algoritmo sudėtingumas yra $O(KN)$, o $N \leq 20\,000$, tokia programa neviršija vykdymo laiko ribojimų.

```
read N                                              ▷ Sveikinimo ilgis
for i ← 0 to N − 1
    do
        read raide[i]                               ▷ Sveikinimas
didziausias ← −∞                                    ▷ Didžiausias galimas grožio įvertis
for m ← 2 to min{N, 156}                            ▷ Perrenkame galimas reikšmes M = m
    do
        ivertis ← 0                                 ▷ Įvertis parinkus konkrečią m reikšmę
        for s ← 0 to m − 1,   r ← 'a' to 'z'
            do
                A[s, r] ← 0                         ▷ Raidžių r skaičius stulpelyje s
        for i ← 0 to N − 1                          ▷ Nagrinėjame raides paeiliui
            do
                ivertis ← ivertis + A[i mod m, raide[i]]
                A[i mod m, raide[i]] ← A[i mod m, raide[i]] + 1
        if ivertis ≥ didziausias
          then
                didziausias ← ivertis
                M ← m                               ▷ M reikšmė, su kuria įvertis didžiausias
print M, didziausias
```

Testų aprašymas.

<table>
<tr><th>Nr.</th><th>Vertė</th><th>N</th><th>M</th><th>Pastabos</th></tr>
<tr><td>0</td><td>1</td><td>21</td><td>2</td><td rowspan="2">Sąlygoje pateikti testai</td></tr>
<tr><td>0</td><td>1</td><td>12</td><td>3</td></tr>
<tr><td>1</td><td>4</td><td>20</td><td>5</td><td>Lengvas testas</td></tr>
<tr><td>2</td><td>4</td><td>92</td><td>23</td><td><code>abc...w</code> kartojasi keturis kartus</td></tr>
<tr><td>3</td><td>4</td><td>100</td><td>47</td><td rowspan="2">Atsitiktinai generuoti testai</td></tr>
<tr><td>4</td><td>4</td><td>200</td><td>32</td></tr>
<tr><td>5</td><td>4</td><td>295</td><td>2</td><td>Techniniai nurodymai</td></tr>
<tr><td>6</td><td>5</td><td>1000</td><td>11</td><td>Atsitiktinai generuotas testas</td></tr>
<tr><td>7</td><td>4</td><td>1318</td><td>2</td><td>Dainos „Hotel California" žodžiai</td></tr>
<tr><td>8</td><td>5</td><td>1500</td><td>15</td><td rowspan="2">Atsitiktinai generuoti testai</td></tr>
<tr><td>9</td><td>4</td><td>3000</td><td>19</td></tr>
<tr><td>10</td><td>4</td><td>4994</td><td>2</td><td>Hano–Banacho teoremos pristatymas ir įrodymas</td></tr>
<tr><td>11</td><td>4</td><td>10000</td><td>3</td><td rowspan="6">Atsitiktinai generuoti testai</td></tr>
<tr><td>12</td><td>4</td><td>12001</td><td>27</td></tr>
<tr><td>13</td><td>4</td><td>15000</td><td>12</td></tr>
<tr><td>14</td><td>4</td><td>15001</td><td>7</td></tr>
<tr><td>15</td><td>4</td><td>15001</td><td>13</td></tr>
<tr><td>16</td><td>4</td><td>17500</td><td>31</td></tr>
<tr><td>17</td><td>5</td><td>19000</td><td>3</td><td rowspan="3">Apsaugoti atsitiktinai generuoti testai</td></tr>
<tr><td>18</td><td>5</td><td>19999</td><td>13</td></tr>
<tr><td>19</td><td>5</td><td>20000</td><td>29</td></tr>
<tr><td>20</td><td>7</td><td>20000</td><td>2</td><td>Vien <code>t</code> raidės</td></tr>
</table>

Dauguma testų buvo sugeneruoti atsitiktinai (tiesa, jei kiekviena raidė kiekvienoje pozicijoje būtų vienodai tikėta, visų atsitiktinių testų atsakymas būtų $M = 2$).

Generuojant kiekvieną testą, pirma buvo parenkama M reikšmė, ir tuomet raidės atsitiktinai dėliojamos taip, kad testo atsakymas būtų toks, koks parinktas. Toks generavimo algoritmas leistų programoms, kurios sutrumpina sveikinimą (tai yra nagrinėja tik, pavyzdžiui, pirmą tūkstantį raidžių), dažnai atspėti teisingą M. $17, 18$ ir 19 testo tokia programa neįveiktų.

Daliniai ribojimai ($N \leq 1\,500$) galioja pirmuose aštuoniuose testuose ir abiejuose pavyzdiniuose testuose. Juos įveiktų programa, kurios sudėtingumas $O(N^2)$. Šių testų bendra vertė yra 36, tai yra lygiai 40% visų testų vertės (90). Testus, kuriuose $N > 1\,500$, įveikia efektyvesnės programos.