



Uždavinių sprendimai

Svėrimas (teorinis uždavinys). Optimaliai ieškant sunkiausios iš $N + 1$ skrynių, niekada nereikės atlikti daugiau nei N svėrimų, tačiau nėra svėrimo būdo, kaip rasti sunkiausią skrynią per mažiau nei N svėrimų.

Pirma parodysime, kaip galima aptikti sunkiausią skrynią po ne daugiau nei N svėrimų. Sunumeruokime skrynias nuo 1 iki $N + 1$. Pirmu svėrimu sverkime pirmą ir antrą skrynias ir įsidėmėkime, kuri yra sunkesnė. Antru svėrimu sverkime sunkesniąją iš pirmų dviejų ir trečiąją skrynias ir įsidėmėkime, kuri yra sunkesnė. Kartokime procesą N kartų, k -tuoju svėrimu lygindami $(k + 1)$ -ą skrynią su sunkiausia iš pirmųjų k skrynių. Po N svėrimų žinosime pačią sunkiausią skrynią.

Šiek tiek sunkiau įrodyti, kad nėra geresnio sprendimo būdo. Vaizduokime skrynias kaip grafo viršūnes, o svėrimus kaip briaunas: viršūnes i ir j jungia briauna tuomet ir tik tuomet, jei svėrime i -ą ir j -ą skrynias kartu. Kad galėtume pasakyti, kuri skrynia yra sunkiausia, šis grafas turi būti jungus. Išties, jei negalime briaunomis nukeliauti nuo viršūnės i iki viršūnės j , tai negalime palyginti, kuri iš atitinkamų skrynių yra sunkesnė. Tačiau mažiausiai briaunų turintis jungus grafas yra medis, o medis, turintis $(N + 1)$ -ą viršūnę, turi N briaunų. Taigi reikia bent N svėrimų, kad galėtume nustatyti, kuri skrynia sunkiausia.

Raudonkepuraitė (uždavinys jaunesniesiems). Šis uždavinys yra šiek tiek pasunkintas atvejais uždavinio, kuriame reikia rasti mažiausią skaičių iš duoto skaičių rinkinio.

Iš pradžių pasirinkime kandidatą (neprarasdami bendrumo galime tarti, kad tai yra pirmasis laukiantysis sąrašė) ir išsaugokime jo akių, ausų ir dantų ilgius. Tuomet peržiūrėkime visus laukiančiuosius paeiliui ir jei nagrinėjamo laukiančiojo akys, ausys ir dantys bus ne ilgesni nei kandidato akys, ausys ir dantys, tai senąjį kandidatą pamirškime ir nauju kandidatu paskirkime nagrinėjamą laukiantįjį. Kai peržiūrėsime visus laukiančiuosius, likęs kandidatas ir bus tikroji močiutė. Žemiau pateikiamas programos pseudokodas:

```
1  read  $n$ 
2   $k \leftarrow 1$            ▷ kandidato numeris
3  read  $a_k, b_k, c_k$ 
4  for  $i \leftarrow 2$  to  $n$ 
5      do read  $a, b, c$ 
6          if  $a \leq a_k$  and  $b \leq b_k$  and  $c \leq c_k$ 
7              then  $k \leftarrow i$ 
8                   $a_k \leftarrow a$ 
9                   $b_k \leftarrow b$ 
10                  $c_k \leftarrow c$ 
11  print  $k$ 
```

Lyginant laukiantįjį su kandidatu (6 eilutė), galimi keli variantai. Jei $a \leq a_k$, $b \leq b_k$ ir $c \leq c_k$, tai kandidatas tikrai buvo vilkas. Gali būti, kad $a \geq a_k$, $b \geq b_k$ ir $c \geq c_k$. Tuomet žinome, kad nagrinėjamas laukiantysis tikrai yra vilkas. Tačiau nagrinėjamas laukiantysis ir kandidatas gali būti nepalyginami nei pirmuoju, nei antruoju būdu — tuomet žinome, kad nei vienas iš jų nėra tikroji močiutė. Tai reikštų, kad tikrosios močiutės dar „nesutikome“, ji turi būti



kažkur toliau sąrašė — pateiktas sprendimas remiasi faktu, kad tikroji močiutė tarp vilkų tikrai yra ir ją vienareikšmiškai galima nustatyti.

Meduolis (uždavinys jaunesniems). Meduoliui yra optimalu perkėlinėti aukščiausiai esančias puodynes į laisvas vietas apačioje, kadangi šitaip elgiantis naujos tuščios vietos bus sukuriamos viršuje, kur jų nereikės užpildyti. Uždavinį lengva išspręsti programa simuliuojant Meduolio darbą, nagrinėjant lentynas nuo apačios ir sekant, kiek puodyninių yra likę virš nagrinėjamosios lentynos. Žemiau pateikiamas programos pseudokodas su paaiškinimu.

```
1  read  $n, p$ 
2   $liko \leftarrow 0$ 
3  for  $i \leftarrow 1$  to  $n$ 
4      do read  $k[i]$ 
5           $liko \leftarrow liko + k[i]$ 
6   $perkelta \leftarrow 0$ 
7   $i \leftarrow 1$ 
8  while  $liko > 0$ 
9      do  $liko \leftarrow \max\{liko - k[i], 0\}$ 
10          $perkelta \leftarrow perkelta + \min\{liko, p - k[i]\}$ 
11          $liko \leftarrow liko - \min\{liko, p - k[i]\}$ 
12          $i \leftarrow i + 1$ 
13  print  $perkelta$ 
```

1–5 eilutėse perskaitomi pradiniai duomenys bei susumuojamas puodyninių skaičius kintamajame $liko$. Toliau skaičiuojamas perkėlimų skaičius. Eilutės 9 ir 11 užtikrina, kad nagrinėjant i -ąją lentyną, kintamojo $liko$ reikšmė yra lygi puodyninių, esančių (likusių) aukščiau i -osios lentynos, skaičiui. Eilutėje 10 nurodoma, kad į nagrinėjamą lentyną perkeliama $p - k[i]$ trūkstamos puodynės iš viršaus. O jei viršuje yra likę mažiau puodyninių, tai perkeliama tiek, kiek liko. Šios puodynės atimamos iš viršuje likusių (11 eilutė).

Pastebėsime, kad jei pradiniai duomenys pateikiami faile, tai sprendime galima nenaudoti masyvų: pirmą kartą skaitant pradinius duomenis galima suskaičiuoti lentynų sumą, o antrą kartą — rasti perkėlimų skaičių, kadangi lentynos pradiniuose duomenyse pateikiamos nuo žemiausiai esančios iki aukščiausiai esančios.

Šokiai (uždavinys vyresniesiems). Pirma paaiškinsime vieną optimalią porų sudarymo strategiją.

Nėra prasmės neporuoti paties aukščiausio vaikinio. Kad ir koks būtų poravimas, jei nesuropuotas aukščiausias vaikinys, tai jis gali pakeisti bet kurią kitą. Vadinasi, visada bus optimalus poravimas, kuriame aukščiausias vaikinys yra suporuotas.

Šiam aukščiausiam vaikinui geriausia į porą skirti kuo aukštesnę merginą, kuri tik nėra per aukšta. Šitaip parinkdami šokių partnerę tikrai nepralošime. Viena vertus, kažkokią merginą turime skirti šiam vaikinui. Kita vertus, parinkdami kuo aukštesnę merginą, nerizikuosime ir paliksime žemesnes merginas žemesniems vaikinams.

Kartodami šiuos veiksmus, gausime maksimalų skaičių porų. Uždavinys prašo suskaičiuoti, koks bus tas skaičius. Galima simuliuoti poravimo algoritmą, vaikus ir merginas išrikiavus



Lietuvos mokinių informatikos olimpiada

Miesto (rajono) etapas • 2009 m. gruodžio 11 d.

pagal jų ūgį. Tačiau žemiau pateiktoje programoje elgiamasi kitaip: pirma suskaičiuojamas vaikinų ir merginų skaičius pagal jų ūgį. Tai galima padaryti, nes ūgiai pateikiami kaip sveiki skaičiai ir kinta mažame intervale: nuo 160 cm iki 200 cm. Tuomet labai lengva vaikus ir merginas nagrinėti pagal jų ūgį ir gaunamas algoritmas, kurio efektyvumas net nepriklauso nuo šokėjų skaičiaus n .

```
read  $n$ 
for  $i \leftarrow 1$  to  $n$ 
  do read  $h$ ;  $V[h] \leftarrow V[h] + 1$            ▷ vaikinų skaičius pagal jų ūgį
for  $i \leftarrow 1$  to  $n$ 
  do read  $h$ ;  $M[h] \leftarrow M[h] + 1$            ▷ merginų skaičius pagal jų ūgį
atsakymas  $\leftarrow 0$ 
for  $v \leftarrow h_{max}$  downto  $h_{min}$              ▷  $v$  — nagrinėjamas vaikinų ūgis
  do for  $m \leftarrow v - 1$  downto  $h_{min}$        ▷  $m$  — nagrinėjamas merginų ūgis
    do  $k \leftarrow \min\{V[v], M[m]\}$          ▷ kiek tokių porų pavyks sudaryti?
       $V[v] \leftarrow V[v] - k$ 
       $M[m] \leftarrow M[m] - k$ 
      atsakymas  $\leftarrow atsakymas + k$ 
print atsakymas
```

Pseudokode naudojamos konstantos: $h_{min} = 160$ ir $h_{max} = 200$.

Šios programos bendras sudėtingumas yra $O(n)$. Kai n didelis, pagrindinis laikas sugaištamas pradiniais duomenims perskaityti.

Šokiai (uždavinys vyresniesiems) — antras sprendimas. Pirma masyje išsaugokime visų vaikinų ūgius. Tuomet paeiliui skaitykime merginų ūgius ir tikrinkime, ar yra bent vienas nesuporuotas vaikas, aukštesnis už šią merginą. Jei yra, suporuokime šią merginą su *pačiu žemiausiu* iš aukštesnių už ją nesuporuotų vaikinų. Jei nėra, palikime šią merginą nesuporuotą.

Įsitinkime, kad šis algoritmas suporuos didžiausią įmanomą skaičių porų. Tegul P_k būna didžiausias galimas skaičius porų, kurias galime sudaryti iš *visų* vaikinų ir *pirmųjų* k merginų ($k = 0, 1, 2, \dots, n$). Įrodysime, jog peržiūrėję k merginų minėtu algoritmu, rasime būtent P_k porų.

Naudokime matematinės indukcijos principą. Pastebėkime, kad $P_0 = 0$ ir $P_{k+1} \leq P_k + 1$ su visais $k = 0, 1, \dots, n - 1$. Aišku, kad peržiūrėję 0 merginų būsime radę 0 porų. Tarkime, kad peržiūrėję j merginų radome P_j porų su kažkoku j ($0 \leq j \leq n - 1$).

Jei yra dar bent vienas nesuporuotas vaikas, aukštesnis už $(j+1)$ -ąją merginą, tai algoritmas ras partnerį šiai merginai, todėl išnagrinėję $j + 1$ merginą turėsime $P_j + 1$ porą. Tačiau optimalus skaičius P_{j+1} yra ne didesnis už $P_j + 1$, taigi iš tikro būsime radę optimalų skaičių porų.

Dabar tarkime, kad visi vaikinai, aukštesni už $(j + 1)$ -ąją merginą, yra jau suporuoti. Šiuo atveju panaudoję $j + 1$ merginą rasime P_j porų, taigi turime parodyti, kad $P_{j+1} = P_j$, t.y. kad negalime sudaryti $P_j + 1$ poros. Iš tiesų, jei sudarytume $P_j + 1$ porą, turėtume panaudoti $(j + 1)$ -ąją merginą (mat panaudoję tik pirmas j merginų galime sudaryti ne daugiau kaip



P_j porų). Tačiau tuomet šiai merginai turėtume priskirti partnerį, kuris jau yra priskirtas. Kadangi mes visuomet priskirdavome žemiausius galimus partnerius, tai viena iš anksčiau suporuotų merginų liktų be partnerio. Taigi negalime sudaryti $P_j + 1$ poros, todėl ir šiuo atveju būsime radę optimalų skaičių porų.

Kadangi peržiūrėję 0 merginų turime optimalų skaičių porų, tai peržiūrėję 1 merginą turėsime optimalų skaičių porų, todėl ir peržiūrėję 2 merginas turėsime optimalų skaičių porų, ir taip toliau iki peržiūrėsime visas n merginų ir tuomet turėsime optimalų skaičių porų (šis argumentas ir yra vadinamas matematinės indukcijos principu).

```
1  atsakymas ← 0
2  read n
3  for i ← 1 to n
4      do read v[i]                ▷ i-tojo vaikinų ūgis
5  for i ← 1 to n
6      do read m                    ▷ nagrinėjamos merginos ūgis
7          u ← ∞                    ▷ kol kas rasto žemiausio tinkamo vaikinų ūgis
8          for j ← 1 to n
9              do if v[j] > m and v[j] < u
10                 then nr ← j      ▷ kol kas rasto žemiausio tinkamo vaikinų numeris
11                     u ← v[j]
12             if u < ∞
13                 then atsakymas ← atsakymas + 1
14                     v[nr] ← 0    ▷ šį vaikiną suporavome, todėl jo nebeuagrinėsime
15  print atsakymas
```

Obuoliai (uždavinys vyresniesiems). Uždavinys prašo pagal duotas koordinates suskaičiuoti, kiek yra obuolių, nukritusių arčiau negu 3 metrus nuo artimiausios obels. Bendruoju atveju, atstumą tarp dviejų taškų galima tikrinti pagal Pitagoro teoremą:

$$(x_1 - x_2)^2 + (y_1 - y_2)^2 < d^2.$$

Svarbu pastebėti, kad tikrinant nelygybę reikėtų netraukti šaknies, nes tai būtų nereikalingi skaičiavimai, galintys įvelti tikslumo klaidų.

Tačiau šiame uždavinyje atstumą galima palyginti dar paprasčiau, nes visos koordinatės pateikiamos sveikais skaičiais ir tikrinama, tik ar atstumas neviršija 3 metrų. Toliau pateikiamas programos pseudokodas:



Lietuvos mokinių informatikos olimpiada

Miesto (rajono) etapas • 2009 m. gruodžio 11 d.

```
read  $n$ 
for  $i \leftarrow 1$  to  $n$ 
    do read  $X[i], Y[i]$ 
 $atsakymas \leftarrow 0$ 
read  $m$ 
for  $i \leftarrow 1$  to  $m$ 
    do read  $x, y$ 
         $arti \leftarrow \text{FALSE}$ 
        for  $j \leftarrow 1$  to  $n$ 
            do if  $|X[j] - x| < 3$  and  $|Y[j] - y| < 3$ 
                then  $arti \leftarrow \text{TRUE}$ 
        if  $arti = \text{TRUE}$ 
            then  $atsakymas \leftarrow atsakymas + 1$ 
print  $atsakymas$ 
```