$\alpha + \omega$, 2001, Nr. 1, 81–83

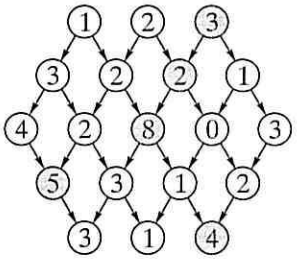
Korys

Jūratė Skūpienė
jurate@ktl.mii.lt

Praėjusiame žurnalo numeryje išspausdinome apžvalginį straipsnį apie šeštąją Baltijos šalių informatikos olimpiadą, vykusią 2000 *m. vasarą Švedijoje, Haningės mieste, Karališkajame technologijos institute. Nuo šio numerio pradedame straipsnių ciklą, kuriame bus spausdinami šios olimpiados uždaviniai su sprendimais. Kiekvieną olimpiados uždavinį rengė vis kitos Baltijos valstybės atstovai. Pradėsime nuo artėjančios pasaulinės informatikos olimpiados šeimininkų — suomių pasiūlyto uždavinio „Korys".*

Skaičių korį, kurio kraštinių ilgiai lygūs 3, matote paveiksle. Kelias prasideda kurioje nors viršutinės eilutės korio akutėje ir baigiasi kurioje nors apatinės eilutės korio akutėje. Iš akutės kelias gali eiti tik įstrižai žemyn į kairę arba įstrižai žemyn į dešinę. Sudarant kelią, *vieną kartą* leidžiama *vienos* horizontalios korio eilutės du skaičius sukeisti vietomis. (Šis sukeitimas iš esmės reiškia, kad galite paimti didžiausią kurios nors eilutės skaičių ir padėti jį į bet kurią tos pačios eilutės akutę.)

Užduotis. *Parašykite programą, kuri paskaičiuotų, kokia gali būti didžiausia kelią sudarančių akučių skaičių suma, kai leidžiama vienos pasirinktos eilutės du skaičius sukeisti vietomis.*



Ribojimai:

- *akutėse įrašyti sveikieji skaičiai iš intervalo* [0; 99];
- *korio kraštinės ilgis yra sveikasis skaičius iš intervalo* [1; 99].

Sprendimas. Jei nebūtų leidžiama sukeisti, tai būtų klasikinis dinaminio programavimo uždavinys. Korys perkeliamas į dvimatę lentelę. Po to korys peržiūrimas iš viršaus į apačią (bei iš kairės į dešinę) ir kiekvienoje jo akutėje įrašytas skaičius pakeičiamas didžiausia akučių, išsidėsčiusių iki tos korio akutės, suma. Ieškomoji suma būtų lygi didžiausiam apatinės eilutės skaičiui.

Prieš modifikuodami šį sprendimą, atkreipkime dėmesį, kad nėra akivaizdu, kaip visas eilutes sutalpinti į atmintį. Todėl rašydami programą, darome paprasčiau: atmintyje laikome tik naujai perskaitytą ir buvusią korio eilutes. Mat ir anksčiau aprašytu, ir modifikuoto sprendimo atveju norint užpildyti kurią nors eilutę (t. y. kiekvienoje nagrinėjamos eilutės akutėje įrašytą skaičių pakeisti didžiausia akučių, išsidėsčiusių iki tos akutės, suma), reikia žinoti tik nagrinėjamą ir prieš tai buvusią eilutes.

Modifikuoto sprendimo algoritmui naudosime keturias lenteles:

eilutė — naujai perskaityta korio eilutė;

nekeista — korio eilutė; jos kiekviename langelyje įrašyta didžiausia kelią iki to langelio sudarančių akučių suma, jei *keitimo iki šiol nebuvo;*

sukeista_seniau — korio eilutė; jos kiekviename langelyje įrašyta didžiausia kelią iki to langelio sudarančių akučių suma, jei *keitimas būtinai buvo atliktas ankstesnėje (ne šioje) eilutėje;*

sukeista_dabar — korio eilutė; jos kiekviename langelyje įrašyta didžiausia kelią iki to langelio sudarančių akučių suma, jei *keitimas atliktas šioje eilutėje;*

Perskaitome pirmąją korio eilutę; ją įsimename lentelėje *nekeista*:

nekeista[*i*] := *eilutė*[*i*];

Šioje eilutėje atliekant keitimą, visus jos elementus būtų galima pakeisti maksimalia reikšme. Tą ir padarome. Rezultatą įsimename lentelėje *sukeista_seniau*:

sukeista_seniau[*i*] := max(*eilutė*)

Perskaitome tolesnę eilutę. Galimi trys variantai:

- *Keitimas iki šiol nebuvo darytas ir nebus šioje eilutėje daromas.*
  Tuomet, remdamiesi lentelėmis *eilutė* bei *nekeista*, atnaujiname eilutę *nekeista*, t. y. randame kelius, kur suma nuo viršutinės eilutės iki perskaitytosios eilutės kiekvieno elemento didžiausia.
- *Keitimas jau padarytas ankstesnėje eilutėje.*
  Tuomet, remdamiesi lentelėmis *eilutė* bei *sukeista_seniau*, atnaujiname eilutę *sukeista_seniau*, t. y. randame ilgiausius kelius nuo viršutinės eilutės iki perskaitytosios kiekvieno elemento.
- *Keitimas daromas šioje eilutėje.*
  Randame didžiausią lentelės *eilutė* elementą. Jį pridedame prie visų lentelės *nekeista* elementų ir rezultatą priskiriame lentelei *sukeista_dabar*. Eilutės *nekeista* reikšmė nepasikeičia.

Taigi perskaitę naują eilutę, perskaičiuojame lentelių *nekeista, sukeista_seniau, sukeista_dabar* reikšmes. Tolesniems skaičiavimams turės įtakos tik tai, ar buvo atliktas keitimas, ar ne. Nebebus svarbu, kurioje iš ankstesnių eilučių tai buvo padaryta. Todėl dar kartą atnaujiname eilutę *sukeista_seniau*:

sukeista_seniau[*i*] := max(*sukeista_seniau*[*i*], *sukeista_dabar*[*i*]),

čia *i* kinta nuo 1 iki eilutės ilgio.

Aprašyti veiksmai kartojami, kol peržiūrimos visos eilutės. Tada belieka išrinkti didžiausią lentelių *nekeista* bei *sukeista_seniau* elementą. Tai ir bus ieškomoji suma.

Reikia nepamiršti, kad čia yra korys, o ne, pavyzdžiui, kvadratinė lentelė, t. y. keičiasi eilučių ilgiai ir į kai kurias akutes iš aukščiau esančios eilutės yra tik vienas kelias, o ne du ar atvirkščiai. Kuriant programą, labai nesunku atsižvelgti į šiuos faktorius. Todėl aprašydami algoritmą į tai nesigilinome.

Panagrinėkime 1 paveiksle pateikto korio pirmąsias keturias eilutes ir pabandykime rasti kelią su didžiausia suma.

Perskaitome pirmąją korio eilutę:

eilutė	1	2	3			

Ją įsimename lentelėje *nekeista*:

nekeista	1	2	3			

Randame didžiausią pirmosios eilutės elementą ir juo užpildome lentelę *sukeista_seniau*:

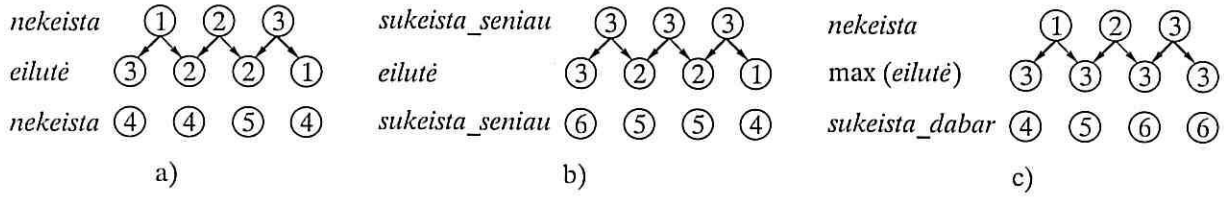
sukeista_seniau	3	3	3			

Perskaitome tolesnę eilutę:

eilutė	3	2	2	1		

Tolesnį algoritmo veikimą iliustruoja pateikti paveikslai:

1 *žingsnis*



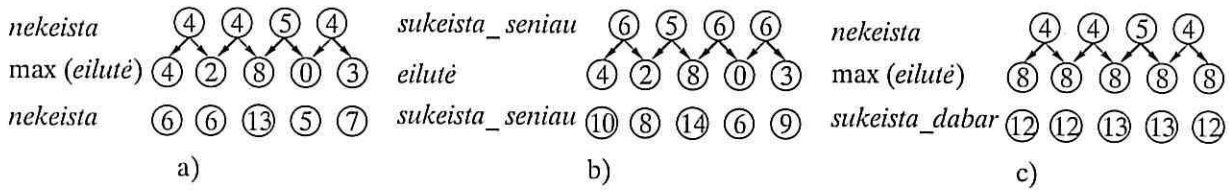
Atnaujiname eilutę *sukeista_seniau*:

sukeista_seniau ⑥ ⑤ ⑤ ④
sukeista_dabar ④ ⑤ ⑥ ⑥
sukeista_seniau ⑥ ⑤ ⑥ ⑥

2 *žingsnis*

Perskaitome tolesnę eilutę:

eilutė	4	2	8	0	3	



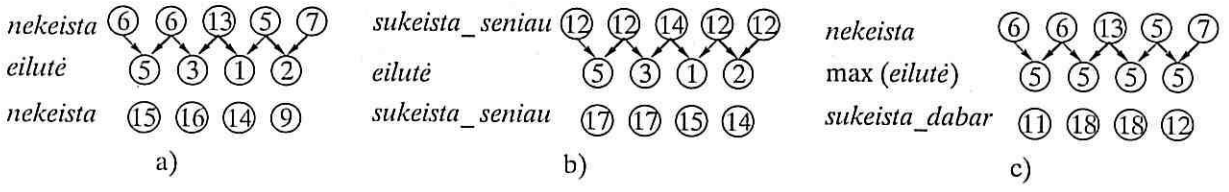
Atnaujiname eilutę *sukeista_seniau*:

sukeista_seniau ⑩ ⑧ ⑭ ⑥ ⑨
sukeista_dabar ⑫ ⑫ ⑬ ⑬ ⑫
sukeista_seniau ⑫ ⑫ ⑭ ⑬ ⑫

3 *žingsnis*

Perskaitome tolesnę eilutę:

eilutė	5	3	1	2	



Atnaujiname eilutę *sukeista_seniau*:

sukeista_seniau ⑰ ⑰ ⑮ ⑭
sukeista_dabar ⑪ ⑱ ⑱ ⑫
sukeista_seniau ⑰ ⑱ ⑱ ⑭

Išrenkame didžiausią eilučių *nekeista* bei *sukeista_seniau* elementą (18). Tai ir yra ieškomoji kelio su didžiausia suma reikšmė. Atlikę dar porą žingsnių, rastume ir didžiausią kelio, einančio per visą korį, akučių sumą.

$\alpha + \omega$