1. Minimaalne maksmine

4 sekundit 100 punkti

Ühes riigis on kasutusel M erineva väärtusega münte, nende hulgas kindlasti ka 1-sendine. Vaatleme ülesannet maksta summa S senti minimaalse müntide arvuga, kui võib eeldada, et igat liiki münte on olemas piisav hulk.

Mõnel juhul sobib selle ülesande lahendamiseks järgmine ahne algoritm: valime suurima mündi, mille väärtus ei ületa S, lahutame summast S selle mündi väärtuse ja jätkame samamoodi, kuni jääk on null. Võiks arvata, et nii kulutatud müntide arv ongi minimaalne.

Siiski pole eeltoodud väide alati õige. Näiteks kui müntide nominaalid on 1, 2, 5, 7 ja 10 ning $S = 14$, annab ahne algoritm kolme mündiga lahenduse ($14 = 10 + 2 + 2$), aga saab maksta ka kahe mündiga ($14 = 7 + 7$).

Siit kerkib muidugi kohe küsimus, milliste väärtuste korral eeltoodud lihtne ahne algoritm valesti töötab?

Kirjutada programm, mis leiab antud müntide nominaalide jaoks summa, mille korral ahne algoritm leiab ebaoptimaalse maksmisviisi.

Sisend. Tekstifaili `min.sis` esimesel real on müntide nominaalide arv M ($1 < M < 100$). Faili teisel real on tühikutega eraldatud nominaalid A_1, A_2, ..., A_M ($1 = A_1 < A_2 < \ldots < A_M \leqslant 7\,000\,000$). Faili viimasel real on kaks täisarvu X ja Y ($0 < X < Y \leqslant 7\,000\,000$).

Väljund. Tekstifaili `min.val` esimesele reale väljastada summa S ($X \leqslant S \leqslant Y$), mille korral ahne algoritm ei leia optimaalset maksmisviisi. Faili teisele reale väljastada ahne algoritmi leitust parema maksmisviisi kirjeldus: M tühikutega eraldatud mittenegatiivset täisarvu B_1, B_2, ..., B_M, mille korral $A_1 \cdot B_1 + A_2 \cdot B_2 + \ldots + A_M \cdot B_M = S$ ja $B_1 + B_2 + \ldots + B_M$ on väiksem kui ahne algoritmi kulutatud müntide arv. Kui võimalikke lahendusi on mitu, väljastada ükskõik milline neist. Võib eeldada, et igas testis leidub vähemalt üks S väärtus, mille korral ahne algoritm ei anna optimaalset lahendust.

Näide.

`min.sis`
```
5
1 2 5 7 10
1 100
```

`min.val`
```
14
0 0 0 2 0
```

2. MP3

1 sekund 100 punkti

Mati tahab endale uue MP3-mängija osta. Ta leidis kolm võrdlustesti, milles igaühes olid kõik N müügil olevat MP3-mängijat pingeritta seatud. Mati otsustas, et kui üks mängija on kõigis kolmes testis teisest mängijast eespool, siis on teine mängija selgelt kehvem kui esimene. Muidugi tasub ostmiseks kaaluda ainult selliseid mängijaid, mis pole ühestki teisest selgelt kehvemad.

Kirjutada programm, mis leiab testide pingeridade põhjal selliste mängijate arvu, mida on üldse mõtet kaaluda.

Sisend. Tekstifaili `mp3.sis` esimesel real on müügil olevate MP3-mängijate arv N ($3 \leqslant N \leqslant 500\,000$). Faili kolmel järgmisel real on igaühel N tühikutega eraldatud täisarvu, mängijate järjestus ühes pingereas. Mängijad on tähistatud $1 \ldots N$ ja iga pingerida on nende tähiste permutatsioon.

Väljund. Tekstifaili `mp3.val` ainsale reale väljastada üks täisarv, nende MP3-mängijate arv, mis pole ühestki teisest selgelt kehvemad.

Näide.

`mp3.sis`
```
3
2 3 1
3 1 2
1 2 3
```

`mp3.val`
```
3
```

Ükski mängija pole ühestki teisest selgelt kehvem, järelikult tuleks kõiki kaaluda.

Näide.

`mp3.sis`
```
10
2 5 3 8 10 7 1 6 9 4
1 2 3 4 5 6 7 8 9 10
3 8 7 10 5 4 1 2 6 9
```

`mp3.val`
```
4
```

Kaaluda tasub mängijaid 1, 2, 3 ja 5.

3. Lühim tee

1 sekund 100 punkti

Ühes linnas on N ristmikku (nummerdatud $1 \ldots N$). Iga ristmiku r kohta on teada tema koordinaadid X_r ja Y_r. Lisaks on teada, et ristmikult r on võimalik liikuda M_r teisele ristmikule (mida edaspidi nimetame ristmiku r naaberristmikeks) mööda ühesuunalisi sirgjoonelisi tänavaid (kahesuunalised tänavad on kirjeldatud kahe ühesuunalise tänavana).

Samadel koordinaatidel võib erinevatel tasapindadel olla mitu ristmikku, aga kui nende vahel pole tänavat kirjeldatud, siis ei saa otse ühelt ristmikult teisele sõita. Samuti võivad tänavad väljaspool ristmikke üksteise alt läbi minna, aga kui selline lõikumine pole ristmikuna kirjeldatud, ei saa seal ühelt tänavalt teisele keerata.

Üks firma tahab tulla turule kaardirakendusega nutitelefonidele, millel ei tarvitse olla piisavalt mälu, et linna kaart kohe koos rakendusega sisse laadida. Seetõttu tuleb vajalikke andmeid arvutuste käigus jooksvalt serverist pärida. Muidugi on väiksem päringute arv parem.

Kirjutada programm, mis leiab lühima tee antud ristmikult S antud ristmikule F, kasutades selleks võimalikult vähe päringuid serverisse (päringute loendamine on kirjeldatud allpool).

Selles ülesandes ei tohi lahenduseks olev programm ise sisendit lugeda ega väljundit kirjutada. Selle asemel tuleb kasutada teeki, milles on järgmised funktsioonid:

Funktsioon	Kirjeldus

`void init(`
`  int *n, int *s, int *f);`
`procedure init(`
`  var n, s, f : longint);`
	Tagastab parameetritena antud muutujates N, S, F väärtused. Seda funktsiooni tuleb välja kutsuda täpselt üks kord enne kõiki teisi funktsioone.

`void getXYM(int r,`
`  int *x, int *y, int *m);`
`procedure getXYM(r : longint;`
`  var x, y, m : longint);`
	Saab esimeses parameetris ristmiku numbri r ja tagastab järgmiste parameetritena antud muutujates X_r, Y_r, M_r väärtused.

`void getQ(int r,`
`  int *q);`
`procedure getQ(r : longint;`
`  var q : longint);`
	Saab esimeses parameetris ristmiku numbri r ja tagastab teise parameetrina antud muutujas ühe naaberristmiku numbri. Iga järgmine `getQ()` väljakutse sama r väärtusega tagastab järgmise naabri numbri. Seda funktsiooni võib iga r väärtuse jaoks välja kutsuda ülimalt M_r korda.

`void put(int r);` `procedure put(r : longint);`	Teatab järgmisena läbitava ristmiku numbri r. Kui lühim tee ristmikult S ristmikule F läbib k ristmikku (S ja F kaasa arvatud), tuleb seda funktsiooni välja kutsuda täpselt k korda, kusjuures esimesel väljakutsel peab r väärtus olema S ja viimasel väljakutsel F.
`void done();` `procedure done();`	Salvestab funktsiooni `put()` kaudu teatatud tulemused hindamiseks. Seda funktsiooni tuleb välja kutsuda täpselt üks kord pärast kõiki teisi funktsioone.

Teegi kasutamiseks C/C++ programmis tuleb programmi lisada rida `#include "teelib.h"`, Pascalis rida `{$include teelib.inc}`.

Oma lahenduse oma arvutis testimiseks saate koos sisendi ja väljundi näidetega teegid, mis loevad sisendi failist `tee.sis` ja väljastavad faili `tee.val` tegevuste logi ja hindamise tulemuse. Sisendfaili esimesel real on N ($1 \leqslant N \leqslant 7000$), S ($1 \leqslant S \leqslant N$), F ($1 \leqslant F \leqslant N$); järgmisel N real on igaühel ühe ristmiku kirjeldus: X_r ($0 \leqslant X_r \leqslant 10\,000$), Y_r ($0 \leqslant Y_r \leqslant 10\,000$), M_r ($0 \leqslant M_r \leqslant 5$) ja nende järel M_r naaberristmiku tähised; sisendis kirjeldatud tänavate koguarv ei ületa 20 000. Väljundfailis on programmi ja teegi vahelise suhtluse logi; faili viimane rida näitab, kas väljastatud tee oli korrektne, kuid ei anna infot selle kohta, kas see oli lühim võimalik või kui palju selle leidmise eest punkte saaks. Teekidest võib kasutada ainult eelpool kirjeldatud funktsioone (ja serveris testimisel kasutatakse niikuinii teistsuguseid teeke).

Hindamine. Iga testi eest saadud punktide arv sõltub funktsioonide `getXYM()` ja `getQ()` väljakutsete koguarvust Q. Kui lahendus väljastab lühima tee, võrreldakse arvu Q näidislahenduse tehtud väljakutsete arvuga Q'. Kui $Q \leqslant Q'$, saab lahendus 100% selle testi väärtusest, vastasel korral $(40 \cdot Q'/Q + 20)\%$ testi väärtusest. Kui programm väljastab tee, mis ei vii ristmikult S ristmikule F või mis pole lühim võimalik või rikub teegi kasutamisel eelpool toodud reegleid, saab ta selle testi eest 0 punkti.