Viimane õhtusöök

Leonardo oli väga hõivatud, kui ta töötas Viimase Õhtusöögi, oma kõige kuulsama seinamaali, kallal. Üks tema päeva esimesi töid oli otsustada, mis temperavärve ta kasutab ülejäänud tööpäeva jooksul. Ta vajas palju värve, kuid sai tellingul hoida ainult piiratud hulka nendest. Tema abiline pidi muude kohustuste kõrvalt tellingule ronima, et värve talle üles kätte viia ja siis alla tuua ning panna tagasi oma kohale põrandal.

Selles ülesandes pead sa kirjutama kaks eraldi programmi abilise aitamiseks. Esimene programm saab sisendiks Leonardo instruktsioonid (nende värvide jada, mida Leonardo päeva jooksul vajab) ja koostab *lühikese* bittide jada, mida nimetatakse *abijadaks*. Kui abiline täidab töö käigus Leonardo nõudmisi, siis ei ole tal käepärast infot Leonardo järgmiste nõudmiste kohta, ainult see abijada, mille pani kokku sinu esimene programm. Teine programm saab ette abijada ja seejärel võtab vastu ja töötleb Leonardo nõudmised elavas järjekorras (s.t. üks nõudmine korraga). See programm peab aru saama, mida abijada tähendab ja kasutama seda optimaalsete valikute tegemiseks. Kõik see on allpool detailsemalt selgitatud.

Värvide liigutamine põranda ja tellingute vahel

Vaatleme üht lihtsustatud stsenaariumi. Oletame, et on olemas N värvi, tähistatud arvudega 0 kuni N-1, ja et iga päev küsib Leonardo assistendi käest uut värvi täpselt N korda. Olgu C Leonardo poolt küsitud N värvi küsimise jada. Seega me võime mõelda jadast C kui N arvu järjestusest, kus iga arv on 0 kuni N-1 (kaasa arvatud). Pange tähele, et mõned värvid ei pruugi jadas C üldse esineda ja teised võivad esineda mitmeid kordi.

Värvide ruum tellingul on alati täis ja sisaldab mingeid K värvi N värvi hulgast ($K < N$). Tööpäeva alguses on tellingul värvid numbritega 0 kuni K-1 (kaasa arvatud).

Abiline täidab Leonardo nõudmisi ükshaaval. Kui nõutud värv *on juba tellingul*, saab abiline puhata. Vastasel juhul peab ta võtma põrandalt nõutud värvi ja viima selle tellingule. Kuna tellingul pole uue värvi jaoks ruumi, peab abiline valima ühe tellingul oleva värvi ja viima selle alla põrandale.

Leonardo optimaalne strateegia

Abiline tahab puhata nii palju kui võimalik. Nende värviküsimiste ehk nõudmiste arv, mille ajal abiline saab puhata, sõltub tema valikutest töö tegemise käigus. Täpsemalt öeldes, iga kord, kui abiline peab eemaldama tellingutelt ühe värvi, võivad erinevad valikud viia erineva tulemuseni tulevikus. Leonardo seletas talle, kuidas ta võib saavutada oma eesmärgi, teades C sisu. Parim valik tellingult eemaldatava värvi osas on saavutatav, arvestades, millised värvid on parajasti tellingul ja millised on tulevased värvinõudmised jadas C. Tellingult tuleb valida värv vastavalt järgmistele

reeglitele:

- kui tellingul on värv, mida ei lähe tulevikus enam vaja, siis peaks abiline tellingult eemaldama selle värvi;
- vastasel juhul tuleb eemaldada tellingult värv, *mida läheb vaja hiljem kui ükskõik millist teist värvi parajasti tellingul olevate värvide hulgast*. (Ehk, iga tellingul oleva värvi jaoks tuleb leida selle esimene kasutamine tulevikus ja eemaldada tuleb värv, mida läheb vaja kõige kaugemas tulevikus.)

On võimalik tõestada, et Leonardo strateegiat kasutades saab abiline puhata maksimaalsel võimalikul arvul juhtudel.

Näide 1

Olgu N = 4, seega on meil 4 värvi (numbritega 0 kuni 3) ja 4 nõudmist. Olgu nõudmiste jada C = (2, 0, 3, 0). Samuti oletame, et K = 2. See tähendab, et igal ajahetkel on Leonardol tellingul ruumi 2 värvi hoidmiseks. Vastavalt eeltoodule on algselt tellingul värvid 0 ja 1. Me paneme tellingu sisu kirja järgmiselt: [0, 1]. Üks võimalikest viisidest, kuidas abiline võib nõudmistega hakkama saada, on järgmine.

- Esimene nõutud värv (number 2) ei ole tellingul. Abiline viib selle sinna ja otsustab tellingult alla tuua värvi 1. Tellingu seisuks saab [0, 2].

- Järgmine nõutud värv (number 0) on juba tellingul ja abiline saab puhata.

- Kolmanda nõudmise peale (number 3) eemaldab abiline värvi 0, saavutades tellingu seisuks [3, 2].

- Lõpuks tuleb viia põrandalt tellingule viimane nõutud värv (number 0). Abiline otsustab tellingult eemaldada värvi 2 ja tellingu seisuks saab [3, 0].

Pange tähele, et esitatud näites ei järginud abiline Leonardo optimaalset strateegiat. Optimaalse strateegia korral oleks kolmanda sammu ajal pidanud abiline eemaldama värvi 2 ja siis oleks ta saanud viimase sammu ajal jälle puhata.

Abilise strateegia, kui tema mälu on piiratud

Ühel hommikul palub abiline Leonardot, et ta kirjutaks jada C paberile, et ta saaks plaanida ja tegutseda optimaalse strateegia järgi. Kuna Leonardo tahab hoida oma töö tehnikaid saladuses, ei luba ta abilisele sellist paberit. Ainus, mida ta abilisele lubab, on jada C läbi lugeda ja püüda see meelde jätta.

Kahjuks on abilise mälu väga halb. Ta on võimeline meelde jätma ainult kuni M bitti infot. Üldiselt võib see tal kogu jada C meelde jätmist takistada. Seega peab abiline leiutama kavala viisi, kuidas koostada selline bittide jada, mida ta suudab meelde jätta. Nimetame selle jada *abijadaks* ja tähistame tähega A.

Näide 2

Hommikul võib abiline võtta Leonardo paberi jadaga C, lugeda seda ja teha kõik vajalikud valikud. Üks asi, mida ta võib tahta teha, on uurida tellingu seisu iga nõudmise järel. Näiteks, kui kasutada (ebaoptimaalset) strateegiat näitest 1, oleks tellingu olekute jada [0, 2], [0, 2], [3, 2], [3, 0]. (Meenutame, et ta teab tellingu algolekut [0, 1].)

Oletame nüüd, et M = 16, seega abiline on võimeline meelde jätma kuni 16 bitti informatsiooni. Kuna N = 4, siis saame salvestada iga värvi 2 biti abil. Seega 16 bitti on piisav, et salvestada eeltoodud tellingu olekute jada. Abiline koostab siis järgmise abijada: A = (`0`, `0`, `1`, `0`, `0`, `0`, `1`, `0`, `1`, `1`, `1`, `0`, `1`, `1`, `0`, `0`).

Hiljem saab abiline seda abijada dekodeerida ja oma valikute tegemiseks kasutada.

(Muidugi võib abiline M = 16 korral otsustada meelde jätta terve jada C, kasutades ära ainult 8 bitti võimalikust 16-st. Selle näitega me tahtsime illustreerida, et tal võib olla ka teistsuguseid valikuid ning see näide ei reeda ühtegi head lahendust.)

Ülesanne

Sa pead kirjutama *kaks eraldiseisvat programmi* samas programmeerimiskeeles. Need programmid käivitatakse järjestikku nii, et neil ei ole võimalik üksteisega töö ajal suhelda.

Esimene programm on see, mida abiline kasutab hommikul. Sellele programmile antakse ette jada C ja see peab koostama abijada A.

Teist programmi kasutab abiline töö ajal. See programm saab sisendisse abijada A ja peab töötlema Leonardo nõudmisi jadas C. Pane tähele, et jada C avalikustatakse sellele programmile üks nõudmine korraga ja iga nõudmine peab olema rahuldatud enne, kui antakse järgmine.

Täpsemalt öeldes, esimeses programmis tuleb realiseerida protseduur `ComputeAdvice`(C, N, K, M), kus jada C koosneb N täisarvust (igaüks neist 0…N-1), K on värvide arv tellingul ja M on bittide arv, mida saab kasutada abijada jaoks. See protseduur peab koostama abijada A, mis koosneb kuni M bitist. Seejärel peab programm edastama abijada A väljakutsuvale süsteemile, kutsudes abijada A iga biti jaoks järgmist protseduuri:

- `WriteAdvice`(B) — lisa bitt B abijada A lõppu. (Seda protseduuri saad välja kutsuda ülimalt M korda).

Teises programmis pead sa realiseerima protseduuri `Assist`(A, N, K, R). Selle protseduuri sisendiks on abijada A, täisarvud N ja K nii, nagu need defineeriti eespool ja abijada tegelik pikkus R bittides (R ≤ M). See protseduur peab läbi viima sinu poolt pakutava strateegia abilise jaoks, kasutades järgmisi etteantud alamprogramme:

- `GetRequest`() — tagastab järgmise värvi, mida Leonardo nõuab. (Infot tulevaste nõuete kohta ei anta.)

- `PutBack`(T) — võtab tellingult värvi T ja paneb selle põrandale. Seda protseduuri tohib välja kutsuda ainult sellise värvi T väärtusega, mis on parajasti tellingul.

Töö käigus peab sinu protseduur `Assist` pöörduma funktsiooni `GetRequest` poole täpselt N korda, saades iga kord teada järjekordse Leonardo nõudmise. Pärast iga pöördumist funktsiooni `GetRequest` poole: kui tagastatav värv *ei ole* tellingul, *pead* sa välja kutsuma ka protseduuri `PutBack`(T) sinu poolt valitud T väärtusega; vastasel juhul *ei tohi* protseduuri `PutBack` poole pöörduda. Kui programm neid nõudeid ei järgi, siis loetakse see veaks ja programmi töö lõpetatakse. Meenutame, et tööpäeva alguses on tellingul värvid numbritega 0 kuni K-1, äärmised väärtused kaasa arvatud.

Iga test loetakse lahendatuks, kui sinu kaks programmi järgivad kõiki püstitatud piiranguid ja protseduuri `PutBack` poole pöördumiste koguarv on *täpselt võrdne* Leonardo optimaalse strateegia poolt pakutavaga. Pane tähele, et kui leidub mitu strateegiat, mis saavutavad sama pöördumiste arvu protseduuri `PutBack` poole, võib sinu programm realiseerida suvalise neist. (S.t. ei ole kohustuslik järgida Leonardo strateegiat, kui on olemas teine sama hea strateegia.)

Näide 3

Jätkates näidet 2, oletame, et protseduuris `ComputeAdvice` leidsid sa abijada A = (`0`, `0`, `1`, `0`, `0`, `0`, `1`, `0`, `1`, `1`, `1`, `0`, `1`, `1`, `0`, `0`). Selleks, et edastada see süsteemile, pead sa tegema järgmise väljakutsete jada: `WriteAdvice(0)`, `WriteAdvice(0)`, `WriteAdvice(1)`, `WriteAdvice(0)`, `WriteAdvice(0)`, `WriteAdvice(0)`, `WriteAdvice(1)`, `WriteAdvice(0)`, `WriteAdvice(1)`, `WriteAdvice(1)`, `WriteAdvice(1)`, `WriteAdvice(0)`, `WriteAdvice(1)`, `WriteAdvice(1)`, `WriteAdvice(0)`, `WriteAdvice(0)`.

Seejärel käivitatakse sinu teine protseduur `Assist` ja see saab sisendiks eespool koostatud abijada A ning väärtused N = 4, K = 2 ja R = 16. Protseduur `Assist` peab siis tegema täpselt N = 4 pöördumist funktsiooni `GetRequest` poole. Lisaks peab protseduur `Assist` mõne sellise nõudmise järel pöörduma protseduuri `PutBack`(T) poole sobiva T väärtusega.

Järgnev tabel näitab väljakutsete jada, mis vastab (ebaoptimaalsetele) valikutele näitest 1. Miinusmärk tähistab, et protseduuri `PutBack` poole ei pöörduta.

`GetRequest()`	Tegevus
2	`PutBack`(1)
0	-
3	`PutBack`(0)
0	`PutBack`(2)

Alamülesanne 1 [8 punkti]

- $N \leq 5\,000$.
- Sa võid kasutada ülimalt M = 65 000 bitti.

Alamülesanne 2 [9 punkti]

- $N \leq 100\,000$.

- Sa võid kasutada ülimalt M = 2 000 000 bitti.

Alamülesanne 3 [9 punkti]

- N ≤ 100 000.
- K ≤ 25 000.
- Sa võid kasutada ülimalt M = 1 500 000 bitti.

Alamülesanne 4 [35 punkti]

- N ≤ 5 000.
- Sa võid kasutada ülimalt M = 10 000 bitti.

Alamülesanne 5 [kuni 39 punkti]

- N ≤ 100 000.
- K ≤ 25 000.
- Sa võid kasutada ülimalt M = 1 800 000 bitti.

Selle alamülesande skoor sõltub sinu programmi poolt kasutatava abijada pikkusest R. Täpsemalt öeldes, kui R_{max} on suurim (üle kõigi testide) sinu protseduuri `ComputeAdvice` poolt koostatud abijadade pikkus, siis on sinu skoor:

- 39 punkti, kui $R_{max} \leq 200\,000$;
- 39 $(1\,800\,000 - R_{max}) / 1\,600\,000$ punkti, kui $200\,000 < R_{max} < 1\,800\,000$;
- 0 punkti, kui $R_{max} \geq 1\,800\,000$.

Realisatsioon

Sa pead esitama täpselt kaks faili, mis on *kirjutatud samas programmeerimiskeeles*.

Esimese faili nimi on `advisor.c`, `advisor.cpp` või `advisor.pas`. See fail peab realiseerima protseduuri `ComputeAdvice` nii, nagu eespool sai kirjeldatud ja võib välja kutsuda protseduuri `WriteAdvice`. Teise faili nimi on `assistant.c`, `assistant.cpp` või `assistant.pas`. See fail peab realiseerima protseduuri `Assist` nii, nagu eespool sai kirjeldatud ja võib välja kutsuda funktsiooni `GetRequest` ja protseduuri `PutBack`.

Alamprogrammide signatuurid on järgmised.

C/C++ programmid

```
void ComputeAdvice(int *C, int N, int K, int M);
void WriteAdvice(unsigned char a);
```

```
void Assist(unsigned char *A, int N, int K, int R);
void PutBack(int T);
int GetRequest();
```

Pascali programmid

```
procedure ComputeAdvice(var C : array of LongInt; N, K, M : LongInt);
procedure WriteAdvice(a : Byte);
```

```
procedure Assist(var A : array of Byte; N, K, R : LongInt);
procedure PutBack(T : LongInt);
function GetRequest : LongInt;
```

Need protseduurid peavad käituma nii, nagu eespool sai kirjeldatud. Muidugi on sul lubatud kirjutada teisi alamprogramme sisemiseks kasutuseks. C/C++ programmides peavad sinu sisemised funktsioonid olema deklareeritud kui `static`, kuna lokaalne hindaja lingib need kokku. Alternatiiviks on hoiduda samanimeliste funktsioonide kasutamisest mõlemas programmis. Sinu poolt esitatud programmid ei tohi mingil viisil suhelda standardsisendi ja -väljundiga ega ka ühegi failiga.

Programmeerimise ajal pead sa arvestama järgmiste instruktsioonidega (töökeskkonnas olevad mallid juba rahuldavad esitatavaid nõudeid).

C/C++ programmid

Oma lahenduse algusesse pead sa lisama read failide `advisor.h` ja `assistant.h` kaasamiseks. Need on vastavalt nõuandja ja abilise jaoks. Selleks lisa oma lähtekoodile rida:

```
#include "advisor.h"
```

või

```
#include "assistant.h"
```

Failid `advisor.h` ja `assistant.h` on sulle kättesaadavad sinu töökeskkonnas ja samuti ka võistluse veebilehel. Samuti on samades kohtades olemas kood ja skriptid sinu lahenduse kompileerimiseks ja testimiseks. Täpsemalt, pärast oma lahenduse kopeerimist nende skriptidega samasse kataloogi pead sa käivitama `compile_c.sh` või `compile_cpp.sh` sõltuvalt sinu poolt valitud keelest.

Pascali programmid

Sa pead kasutama mooduleid `advisorlib` ja `assistantlib` vastavalt nõuandja ja abilise jaoks. Selleks pead sa lisama oma programmi koodi rea:

```
uses advisorlib;
```

või

```
uses assistantlib;
```

Failid `advisorlib.pas` ja `assistantlib.pas` on sulle kättesaadavad sinu töökeskkonnas ja samuti ka võistluse veebilehel. Samuti on samades kohtades olemas kood ja skriptid sinu lahenduse kompileerimiseks ja testimiseks. Täpsemalt, pärast oma lahenduse kopeerimist nende skriptidega samasse kataloogi pead sa käivitama `compile_pas.sh`.

Lokaalne hindaja

Lokaalne hindaja on valmis töötama järgmisel viisil vormistatud sisendandmetega:

- rida 1: N, K, M;
- read 2, …, N + 1: C[i].

Hindaja käivitab kõigepealt protseduuri `ComputeAdvice`. See tekitab faili `advice.txt`, mis sisaldab tühikutega eraldatult abijada üksikuid bitte ja lõputunnusena arvu 2.

Seejärel käivitab see protseduuri `Assist` ja genereerib väljundi, kus iga rida on kas kujul "`R [arv]`" või "`P [arv]`". Esimest tüüpi read tähistavad protseduuri `GetRequest()` väljakutseid ja saadud vastuseid. Teist tüüpi read tähistavad protseduuri `PutBack`() väljakutseid ja värve, mis on valitud tellingult põrandale panemiseks. Väljund lõppeb reaga kujul "`E`".

Pane tähele, et sinu programmi töötamise aeg ametliku hindajaga võib erineda sellest, mis paistab lokaalsest hindajast. Vahe ei tohiks olla kuigi suur, aga sellegipoolest võid kasutada testliidest ja kontrollida oma lahenduse mahtumist ajapiirangu raamidesse.

Aja- ja mälupiirangud

- Ajapiirang: 7 sekundit.
- Mälupiirang: 256 MiB.