Paskutinė vakarienė

Leonardo intensyviai tapė savo garsiausią freską, *Paskutinė vakarienė*. Kiekvienos dienos pradžioje jis nuspręsdavo kokių temperos (tam tikra dažų rūšis) spalvų jam reiks tą dieną. Leonardo reikėdavo daug spalvų, tačiau ant pastolių galėdavo laikyti tik nedidelį spalvų skaičių. Jo padėjėjas turėdavo užlipti ant pastolių ir atnešti reikiamos spalvos dažų, tada nulipti su dažais, kurių nebereikia, bei padėti juos atgal į lentyną.

Jums reikia parašyti dvi atskiras programas, kurios padėtų Leonardo pagalbininkui. Pirmajai programai pateikiami Leonardo pageidavimai (spalvų, kurių tą dieną reikės Leonardui, seka). Programa pagalbininkui turi pateikti patarimų seką (bitų eilutę). Dieną, vykdydamas Leonardo pageidavimus, pagalbininkas negalės iš Leonardo užrašų perskaityti būsimų tos dienos pageidavimų.

Antrajai programai bus pateikiama patarimų seka. Tuomet ši programa paeiliui gaus Leonardo pageidavimus ir juos po vieną apdoros. Ji turi suprasti, ką reiškia gautoji patarimų seka ir kaip ją panaudoti optimaliam pasirinkimui. Žemiau paaiškinsime išsamiau.

Dažų nešiojimas tarp pastolių ir lentynos

Leonardo turi N spalvų dažus, kur spalvos sunumeruotos nuo 0 iki N - 1. Kiekvieną dieną Leonardo paprašo kitos spalvos lygiai N kartų. Tegul C yra N spalvų, kurių paprašė Leonardo, seka. Kitaip sakant, C yra N skaičių nuo 0 iki N - 1 (imtinai) seka. Atkreipkite dėmesį, kad vienų spalvų šioje sekoje gali visai nebūti, o kitos gali būti sutinkamos keletą kartų.

Ant pastolių visuomet prikrauta tiek dažų, kiek telpa, t.y. K spalvų iš N turimų, kur $K < N$. Pradiniu momentu, ant pastolių sudėtos spalvos, kurių numeriai nuo 0 iki K - 1 imtinai.

Pagalbininkas po vieną vykdo Leonardo paliepimus. Jei dažai su Leonardo pageidaujama spalva jau yra ant pastolių, pagalbininkas ilsisi. Priešingu atveju jis paima iš lentynos reikiamą spalvą ir užkelia ant pastolių. Kadangi ant pastolių vietos nėra, pagalbininkas pasirenka, kurią spalvą nukelti nuo pastolių ir padėti atgal į lentyną.

Optimali Leonardo strategija

Pagalbininkas nori kuo daugiau ilsėtis. Poilsio momentų (t.y. Leonardo pageidavimų, kurių metu jam nieko nereikia daryti) skaičius priklauso nuo jo paties pasirinkimų. Leonardo paaiškina jam, kaip pasiekti savo tikslą žinant C. Išanalizavus ant pastolių esančias spalvas bei likusius pageidavimus (sekoje C), galima optimaliai nuspręsti, kurią spalvą nukelti nuo pastolių:

- Jei ant pastolių yra spalva, kurios tą dieną Leonardo nebereikės, ją galima nukelti nuo pastolių.

- Priešingu atveju nuo lentynos nuimama spalva, kurios *tą dieną prireiks vėliausiai*.

Galima įrodyti, kad taikydamas šią Leonardo pasiūlytą strategiją pagalbininkas ilsėsis daugiausiai.

1-as pavyzdys

Tarkime, N = 4. Taigi, turime 4 spalvas (sunumeruotas nuo 0 iki 3) ir 4 pageidavimus: C = (2, 0, 3, 0). Ant Leonardo pastolių vienu metu telpa dviejų spalvų dažai, t.y. K = 2, tad pradiniu momentu ant pastolių yra dažai su spalvomis 0 ir 1. Šitai žymėsime tokiu būdu: [0, 1]. Pagalbininkas gali dirbti taip:

- Ant pastolių nėra pirmos reikiamos spalvos (jos numeris 2). Pagalbininkas ją užkelia ant pastolių ir nukelia spalvą 1. Ant pastolių lieka spalvos [0, 2].
- Antra pageidaujama spalva (jos numeris 0) jau yra ant pastolių, tad pagalbininkas gali ilsėtis.
- Trečios pageidaujamos spalvos (jos numeris 3) ant pastolių nėra ir pagalbininkas nukelia spalvą 0; ant pastolių lieka spalvos [3, 2].
- Galiausiai ant pastolių užkeliama paskutinioji spalva (0), kurios reikia tą dieną. Pagalbininkas nukelia 2-ą spalvą ir ant pastolių lieka spalvos [3, 0].

Atkreipkite dėmesį, kad šiame pavyzdyje Leonardo pagalbininkas netaikė optimalios strategijos: trečiu žingsniu nukelti 2-ą spalvą, kad galėtų ilsėtis, kai Leonardo prireiks paskutiniosios spalvos.

Pagalbininko strategija, kai jo atmintis ribota

Ryte pagalbininkas paprašo, kad Leonardo jam duotų ant popieriaus užrašytą C. Tačiau Leonardo taip saugo savo tapybos paslaptis, kad jis nesutinka pagalbininkui visai dienai patikėti savo užrašų. Jis tik leidžia pagalbininkui perskaityti C, o pastarasis bando įsiminti.

Deja, pagalbininko atmintis nekokia. Jis gali įsiminti tik M bitų. Todėl norėdamas turėti galimybę atkurti C, jis turi sugalvoti būdą kaip tai padaryti: ką įsiminti ir kaip iš įsimintos informacijos atkurti C. Pagalbininko įsimintą seką pažymėsime A ir ją pavadinsime *patarimų seka*.

2-as pavyzdys

Ryte pagalbininkas paima Leonardo užrašus, kuriuose yra C, juos perskaito ir nusprendžia, ką įsiminti. Pavyzdžiui, jis gali nuspręsti po kiekvieno pageidavimo įsiminti ant pastolių esančias spalvas. 1-ame pavyzdyje ant pastolių esančios spalvos kito taip: [0, 2], [0, 2], [3, 2], [3, 0]. Jam nereikia įsiminti, kokios spalvos yra ant pastolių pradiniu momentu ([0, 1]), nes atėjęs dirbti jas mato.

Tarkime, kad pagalbininkas gali įsiminti M = 16 bitų. Jei N = 4, tai kiekvieną spalvą galima įsiminti naudojant 2 bitus. Todėl 16-os bitų užtenka norint įsiminti aukščiau pateiktą pastolių būsenų seką.

Taigi, pagalbininkas suformuoja tokią patarimų seką: A = (`0`, `0`, `1`, `0`, `0`, `0`, `1`, `0`, `1`, `1`, `1`, `0`, `1`, `1`, `0`, `0`).

Dieną pagalbininkas gali iškoduoti patarimų seką ir pasinaudoti ja priimdamas sprendimus.

Šiuo atveju, kai M = 16, pagalbininkas galėtų įsiminti ir seką C. Tam prireiktų 8 bitų iš 16 galimų.

Užduotis

Ta pačia programavimo kalba parašykite *dvi atskiras programas*. Šios programos bus vykdomos paeiliui ir vykdymo metu negalės bendrauti tarpusavyje.

Pirmąją programą pagalbininkas panaudos ryte. Ši programa gaus seką C ir turės sudaryti patarimų seką A.

Antrąją programą pagalbininkas panaudos dieną. Ši programa gaus patarimų seką A ir turės paeiliui apdoroti Leonardo pageidavimų seką C. Bus pateikiama po vieną pageidavimą iš sekos C ir prieš gaunant tolesnį pageidavimą, turės būti įvykdytas ankstesnis pageidavimas.

Konkrečiau, pirmoje programoje turite realizuoti paprogramę `ComputeAdvice`(C, N, K, M), kurios pradiniai duomenys yra masyvas C, sudarytas iš N sveikųjų skaičių, ant pastolių telpančių spalvų skaičius K ir galimų įsiminti bitų skaičius M. Ši programa turi sudaryti patarimų seką A, kurią sudaro ne daugiau nei M bitų. Tada programa turi perduoti sistemai šią seką kiekvienam bitui iškviesdama tokią paprogramę:

- `WriteAdvice`(B) — prideda bitą B prie patarimų sekos A. Šią paprogramę leidžiama iškviesti daugiausiai M kartų.

Antroje programoje realizuokite paprogramę `Assist`(A, N, K, R). Šiai paprogramei bus pateikta patarimų seka A, aukščiau aprašyti sveikieji skaičiai N ir K bei tikrasis patarimų sekos ilgis R ($R \leq M$). Paprogramė turi įvykdyti jūsų siūlomą pagalbininko strategiją kviesdama šias, jums pateiktas paprogrames:

- `GetRequest`() — grąžina tolesnę spalvą, kurios pageidauja Leonardo. (Apie ateities pageidavimus nepranešama.)
- `PutBack`(T) — spalvą T nukelti nuo pastolių į lentyną. Iškvietimo metu spalva T turi būti ant pastolių.

Paprogramė `Assist` turi iškviesti paprogramę `GetRequest` lygiai N kartų, paeiliui gaudama Leonardo pageidavimus. Po kiekvieno paprogramės `GetRequest` iškvietimo, jei grąžinta spalva *nėra* ant pastolių, *reikia* iškviesti paprogramę `PutBack`(T) su pasirinkta spalva T. Priešingu atveju, paprogramės `PutBack` iškviesti *negalima*. Kitoks poelgis yra laikomas klaida ir dėl to programos vykdymas bus nutrauktas. Primename, kad pradžioje ant pastolių yra visos spalvos nuo 0 iki K - 1, imtinai.

Konkretus testas išspręstas, jei abi programos laikosi joms iškeltų reikalavimų, o bendras kreipinių į `PutBack` skaičius *lygus* kreipinių skaičiui taikant Leonardo optimalią strategiją. Atkreipiame dėmesį, kad, jei kelios skirtingos strategijos leidžia pasiekti tą patį kreipinių į `PutBack` skaičių, tai jūsų programa gali rinktis bet kurią. T.y. nereikalaujama taikyti Leonardo strategiją, jei egzistuoja kita tokia pat gera strategija.

3-ias pavyzdys

Tęsdami antrą pavyzdį, tarkime, kad `ComputeAdvice` apskaičiavo A = (`0`, `0`, `1`, `0`, `0`, `0`, `1`, `0`, `1`, `1`, `1`, `0`, `1`, `1`, `0`, `0`).

Norint perduoti šią seką sistemai, reikia atlikti tokius kreipinius: `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)`.

Jūsų antroji programa `Assist` bus vykdoma ir jai bus pateikti tokie pradiniai duomenys: aukščiau pateikta seka A, N = 4, K = 2, ir R = 16. Programa `Assist` lygiai N = 4 kartus kreipsis į `GetRequest`. Po kai kurių kreipinių `Assist` turės kreiptis į `PutBack`(T) su tinkamai parinktu T.

Lentelėje žemiau pateikta kreipinių seka, atitinkantį neoptimalų 1 pavyzdžio sprendinį. Brūkšnelis reiškia, kad į `PutBack` nesikreipta.

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

1 užduotis [8 taškai]

- N ≤ 5 000.
- Galite panaudoti daugiausiai M = 65 000 bitų.

2 užduotis [9 taškai]

- N ≤ 100 000.
- Galite panaudoti daugiausiai M = 2 000 000 bitų.

3 užduotis [9 taškai]

- N ≤ 100 000.
- K ≤ 25 000.
- Galite panaudoti daugiausiai M = 1 500 000 bitų.

4 užduotis [35 taškai]

- $N \leq 5\,000$.
- Galite panaudoti daugiausiai M = 10 000 bitų.

5 užduotis [iki 39 taškų]

- $N \leq 100\,000$.
- $K \leq 25\,000$.
- Galite panaudoti daugiausiai M = 1 800 000 bitų.

Šios užduoties įvertinimas priklauso nuo jūsų programos perduoto patarimo ilgio R. Konkrečiau, jei R_{max} yra didžiausias (iš visų testų) jūsų paprogramės `ComputeAdvice` sudarytos patarimų sekos ilgis, tai jūsų įvertinimas bus lygus:

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

Realizacija

Pateikite du failus, parašytus *ta pačia programavimo kalba*.

Pirmasis failas turi vadintis `advisor.c`, `advisor.cpp` arba `advisor.pas`. Šiame faile turi būti realizuota aukščiau aprašyta paprogramė `ComputeAdvice` ir galima iškviesti paprogramę `WriteAdvice`. Antrasis failas turi vadintis `assistant.c`, `assistant.cpp` arba `assistant.pas`. Šiame faile turi būti realizuota aukščiau aprašyta paprogramė `Assist` ir galima iškviesti paprogrames `GetRequest` ir `PutBack`.

Visų paprogramių antraštės yra tokios.

Programuojantiems C/C++

```
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();
```

Programuojantiems Paskaliu

```
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;
```

Šios paprogramės turi veikti taip, kaip aprašyta aukščiau. Be abejo, galite sukurti daugiau paprogramių vidiniam vartojimui. C/C++ programose vidinio naudojimo paprogramės turėtų būti `static`, kadangi pavyzdinis vertintojas kompiliuos programas kartu. Arba, galite tiesiog nevadinti paprogramių vienodais vardais. Jūsų pateiktas sprendimas neturi dirbti (skaityti ar rašyti) nei su standartiniu įvedimo/išvedimo įrenginiu, nei su jokiu kitu failu.

Programuodami savo sprendimą, taip pat turite atlikti žemiau pateiktus veiksmus (programų šablonai, kuriuos galite rasti savo kompiuteriuose, jau atitinka žemiau pateiktus reikalavimus).

Programuojantiems C/C++

Failo pradžioje turite įtraukti failą `advisor.h` arba `assistant.h`, atitinkamai pirmoje ir antroje programose. Tai padarysite įterpę tokią eilutę:

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

arba

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

Šie du failai, `advisor.h` ir `assistant.h`, bus pateikti kataloge jūsų kompiuteryje bei varžybų sistemoje. Jums taip pat bus pateiktos (tais pačiais būdais) pusprogramės jūsų sprendimo kompiliavimui ir testavimui. Konkrečiau, nukopijavę savo sprendimą į katalogą, turintį šias paprogrames, turėsite įvykdyti `compile_c.sh` arba `compile_cpp.sh` (priklausomai nuo pasirinktos kalbos).

Programuojantiems Paskaliu

Jums reikia naudoti bibliotekas `advisorlib` arba `assistantlib`, atitinkamai pirmoje ir antroje programose. Tai padarysite įterpę tokią eilutę:

```
uses advisorlib;
```

arba

```
uses assistantlib;
```

Du failai, `advisorlib.pas` ir `assistantlib.pas`, bus pateikti kataloge jūsų kompiuteryje bei varžybų sistemoje. Jums taip pat bus pateiktos (tais pačiais būdais) pusprogramės jūsų sprendimo kompiliavimui ir testavimui. Konkrečiau, nukopijavę savo sprendimą į katalogą, turintį šias paprogrames, turėsite įvykdyti `compile_pas.sh`.

Pavyzdinis vertintojas

Pavyzdinis vertintojas duomenis perskaito tokiu formtu:

- 1-oji eilutė: N, K, M;
- 2-oji, …, (N + 1)-oji eilutės: C[i].

Vertintojas pirmiausia įvykdys paprogramę `ComputeAdvice`. Bus sugeneruotas failas `advice.txt` su atskirais patarimų sekos bitais, atskirtais tarpais ir užbaigtais skaičiumi 2.

Tada įvykdys paprogramę `Assist` ir išves rezultatus, kurių kiekviena eilutė yra arba "`R [number]`", arba "`P [number]`". Pirmojo tipo eilutės reiškia paprogramės `GetRequest()` iškvietimus ir gautus atsakymus. Antrojo tipo eilutės reiškia paprogramės `PutBack()` iškvietimus ir pasirinktas nukeliamas spalvas. Po visų šių eilučių bus eilutė "`E`".

Atkreipiame dėmesį, kad oficialiai vertinant jūsų programos darbo trukmė gali truputį skirtis nuo darbo trukmės jūsų kompiuteryje. Šis skirtumas neturėtų būti žymus. Tačiau, vis tiek siūlome naudotis varžybų sistemos testavimo sąsaja ir pasitikrinti, ar jūsų sprendimas atitinka laiko ribojimą.

Laiko ir atminties ribojimai

- Laiko ribojimas: 7 sekundės.
- Atminties ribojimas: 256 MiB.