Poslednja večera

Leonardo je bio veoma aktivan kada je radio na Poslednjoj Večeri (za slučaj da ratari čitaju ovaj tekst, u pitanju je jedna od njegovih najpoznatijih slika na zidu): prvi zadatak svakog dana je bio da odredi koje će boje koristiti tokom tog dana. Leonardu je bilo potrebno puno boja ali je na njegovoj skeli moglo stati samo ograničen broj kanti sa bojama. Njegov asistent je imao zadatak da se penje uz skelu da bi mu donosio kante sa bojama, kao i da sklanja kante sa skele i spušta ih na pod.

U ovom zadatku, potrebno je napisati dva posebna programa koja će pomoći asistentu. Prvi program će kao ulaz prihvatati Leonardove naredbe (niz boja koje će Leonardu trebati tokom dana) i kreirati *kratak* string bitova, zvani *podsetnik*. Dok izvršava Leonardove naredbe tokom dana, asistent neće imati pristup Leonardovim budućim naredbama, već će imati pristup samo podsetniku kreiranom od strane vašeg prvog programa. Drugi program će kao ulaz prihvatati podsetnik, a zatim će prihvatati i izvršavati Leonardove naredbe u "online" stilu (tj. jednu po jednu). Ovaj program mora razumeti šta podsetnik predstavlja i koristiti ga da bi pravio optimalne izbore. U daljem tekstu je sve detaljnije objašnjeno.

Donošenje i odnošenje kanti s bojama

Posmatramo uprošćenu situaciju. Dato je N kanti sa bojama numerisanih od 0 do N - 1; u i-toj kanti se nalazi boja i. Tokom dana Leonardo daje naredbu asistentu za novu boju tačno N puta. Neka je C niz N Leonardovih naredbi; možemo smatrati da je C niz N celih brojeva, svaki između 0 i N - 1 (uključujući i 0 i N - 1). Primetimo da se u nizu C neke boje mogu pojaviti više puta a neke nijednom.

Skela je uvek puna i sadrži nekih K od N kanti sa bojama, pri čemu je $K < N$. Na početku, skela sadrži kante sa bojama od 0 do K - 1 (uključujući i 0 i K - 1).

Asistent izvršava Leonardove upite jedan po jedan. Kad god se tražena boja *već nalazi na skeli*, asistent ne radi ništa i odmara se. Inače, on mora sa poda da uzme kantu sa traženom bojom i da je odnese na skelu. Naravno, na skeli nema mesta za novu kantu, pa asistent mora da izabere neku kantu koja se u tom trenutku nalazi na skeli i da je spusti na pod.

Leonardova optimalna strategija

Asistent želi da se odmara što je više puta moguće. Broj naredbi na kojima on može da se odmara zavisi od izbora "koju kantu skinuti sa skele". Preciznije, svaki put kada asistent treba da skloni kantu sa skele, drugačiji izbori mogu voditi do drugačijih posledica kasnije. Leonardo mu je objasnio kako da postigne svoj cilj znajući niz C. Najbolji izbor za kantu koju treba skinuti sa skele zavisi od trenutnih kanti na skeli i od preostalih naredbi u nizu C. Koju kantu skinuti sa skele u datom trenutku treba odrediti na osnovu sledećih pravila:

- Ako na skeli postoji kanta sa bojom koja neće biti potrebna ubuduće (tj. u narednim naredbama), asistent treba ukloniti kantu sa tom bojom.
- U suprotnom, sa skele treba biti uklonjena kanta sa onom bojom *koja će biti najkasnije potrebna*. (To znači, za svaku boju koja je trenutno na skeli, odredimo njeno prvo pojavljivanje u budućim naredbama. Zatim uklonimo kantu sa bojom čije je prvo pojavljivanje najkasnije.)

Može se dokazati da će se asistent, koristeći Leonardovu strategiju, odmarati naviše moguće puta.

Primer 1

Neka je N = 4, tj. imamo 4 boje (označene od 0 do 3) i 4 naredbe. Neka je niz naredbi C = (2, 0, 3, 0). Takođe, neka je K = 2, odnosno na Leonardovoj skeli mogu biti samo dve kante s bojom u svakom trenutku. Kao što je rečeno, u početku su na skeli boje 0 i 1, što ćemo označavati sa [0, 1]. Jedan mogući način kako asistent može odraditi svoj posao za taj dan je opisan u nastavku.

- Prva tražena boja (boja 2) nije na skeli, pa je asistent nosi gore i skida boju 1 (tj. kantu sa bojom 1) sa skele. Trenutno stanje skele je [0, 2].
- Sledeća boja (boja 0) već je na skeli, pa se asistent može odmarati.
- Za treću naredbu (boja 3), asistent skida kantu sa bojom 0 i time menja sranje skele na [3, 2].
- Konačno, poslednja zahtevana boja (broj 0) se mora staviti sa poda na skelu. Asistent odlučuje da skine kantu sa bojom 2, i trenutno stanje na skeli postaje [3, 0].

Primetimo da u gornjem primeru asistent nije radio prema Leonardovoj optimalnoj strategiji. Optimalna strategija bi bila da izbaci kantu sa bojom 2 u trećem koraku, što bi mu omogućilo da se odmara u poslednjem koraku.

Asistentova strategija kada je njegova memorija ograničena

Na početku dana, asistent zamoli Leonarda da zapiše niz C na komadu papira, kako bi on mogao da odredi i koristi optimalnu strategiju. Međutim, Leonardo želi da sačuva svoju tehniku tajnom i odbija da ostavi papir asistentu. On jedino dozvoljava asistentu da pročita C i pokuša da ga zapamti.

Nažalost, asistent ima loše pamćenje. On je u stanju da zapamti najviše M bitova. U opštem slučaju, zbog toga se može desiti da on ne može rekonstruisati ceo niz C. Zbog toga asistent mora smisliti pametan način izračunavanja niza bitova koje će zapamtiti. Taj niz ćemo zvati "podsetnik" i označavati ga sa A.

Primer 2

Ujutru, asistent može uzeti Leonardov papir sa nizom C, pročitati niz, i napraviti sve neophodne odluke. Jedna mogućnost koju asistent može odabrati je da ispita stanja skele posle svake naredbe. Na primer, kada koristi (neoptimalnu) strategiju iz Primera 1, niz sa stanjima skele bi bio [0, 2], [0, 2], [3, 2], [3, 0]. (Podsetimo se da on zna da je početno stanje skele [0, 1].)

Sada pretpostavimo da je M = 16, pa asistent može da zapamti najviše 16 bitova informacije. Kako

je N = 4, možemo zapamtiti svaku boju koristeći 2 bita. Dakle, 16 bitova je dovoljno da se zapamti gornji niz stanja skele i asistent računa sledeći podsetnik: A = (0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0).

Kasnije tokom dana asistent može dekodirati podsetnik i koristiti ga kako bi odlučio koje boje da spušta sa skele.

(Naravno, sa M = 16 asistent može odlučiti da sačuva ceo niz C koristeći samo 8 umesto raspoloživih 16 bitova. U ovom primeru smo samo želeli da pokažemo da ima i drugih varijanti, koje ne moraju biti dobro rešenje).

Postavka problema

Potrebno je da napišete *dva odvojena programa* u istom programskom jeziku. Ovi programi će biti pokrenuti jedan posle drugog, bez mogućnosti da komuniciraju jedan sa drugim za vreme izvršavanja.

Prvi program će asistent koristiti ujutru. Ovom programu će biti dat niz C, a on mora da izračuna podsetnik - niz A.

Drugi program će asistent koristiti tokom dana. Ovaj program će dobijati podsetnik - niz A, i onda će morati da obradi niz C Leonardovih naredbi. Primetimo da će niz C biti otkriven programu samo jedna po jedna naredba i svaka naredba mora biti obrađena pre dobijanja sledeće.

Preciznije, u prvom programu treba da implementirate jednu funkciju `ComputeAdvice`(C, N, K, M) pri čemu za ulaz imate niz C od N celobrojnih vrednosti (svaka iz skupa 0, …, N - 1), broj K kanti na skeli, i broj bitova M dostupnih za podsetnik. Ovaj program mora da izračuna niz podsetnik A koji se sastoji od najviše M bitova. Program treba da pošalje niz A sistemu pozivanjem sledeće funkcije (koja vam je obezbeđena), redom za svaki bit iz A:

- `WriteAdvice`(B) — dodaje bit B trenutnom podsetniku A. (Ovu metodu možete pozvati najviše M puta).

U drugom programu treba implementirati samo funkciju `Assist`(A, N, K, R). Ulaz za ovu funkciju su podsetnik A, celi brojevi N i K kao što je definisano gore, i dužina R podsetnika A u bitovima (R ≤ M). Ova funkcija izvršava predloženu strategiju za asistenta, koristeći sledeće funkcije koje su Vam obezbeđene:

- `GetRequest`() — vraća sledeću boju koju Leonardo traži. (Informacije o budućim zahtevima u tom trenutku nisu poznate)

- `PutBack`(T) — skida kantu sa bojom T sa skele. Možete pozvati ovu funkciju samo sa vrednošću T za koju važi da se kanta sa tom bojom nalazi na skeli.

Pri izvršavanju vaša funkcija `Assist` mora pozvati `GetRequest` tačno N puta, dobijajući svaki put jednu Leonardovu naredbu. Posle svakog poziva funkcije `GetRequest`, ako tražena boja *nije* na skeli, *morate* pozvati funkciju `PutBack`(T) sa bojom T koju ste izabrali. U suprotnom *ne smete* da pozivate funkciju `PutBack`. U slučaju da ne uradite tako, smatraće se da ste napravili

grešku i vaš program će biti prekinut. Podsećamo da se na početku na skeli nalaze boje od 0 do K - 1, uključujući i njih.

Pojedinačni test primer se smatra rešenim ako vaše funkcije zadovoljavaju sva postavljena ograničenja i ukupan broj poziva `PutBack` je "baš jednak" broju koji se dobija primenom Leonardove optimalne strategije. Primetimo da ako postoji više strategija koje daju isti broj poziva `PutBack`, vaš program može koristiti bilo koju od njih. (tj. nije nužno primenjivati Leonardovu strategiju, ako postoji neka druga podjednako dobra strategija.)

Primer 3

Nastavmo sa Primerom 2, smatrajući da ste u funkciji `ComputeAdvice` izračunali A = (`0`, `0`, `1`, `0`, `0`, `0`, `1`, `0`, `1`, `1`, `1`, `0`, `1`, `1`, `0`, `0`). Da biste to prosledili sistemu, potrebno je izvesti sledeći niz poziva : `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)`.

Tada se može izvršiti vaša druga funkcija `Assist` pri čemi su argumenti za nju gore navedeni niz A, i vrednosti N = 4, K = 2, i R = 16. Funkcija `Assist` tada treba da izvrši tačno N = 4 poziva funkcije `GetRequest`. Takođe, posle nekih of tih poziva, `Assist` treba da pozove funkciju `PutBack`(T) sa adekvatnom vrednošću parametra (argumenta) T.

Tabela ispod prikazuje sekvencu poziva koja odgovara (neoptimalnom) izboru iz Primera 1. Crtica označava da funkcija `PutBack` nije pozvana.

`GetRequest()`	Akcija (radnja)
2	`PutBack`(1)
0	-
3	`PutBack`(0)
0	`PutBack`(2)

Podzadatak 1 [8 bodova]

- $N \leq 5\,000$.
- Možete koristiti najviše M = 65 000 bitova.

Podzadatak 2 [9 bodova]

- $N \leq 100\,000$.
- Možete koristiti najviše M = 2 000 000 bitova.

Podzadatak 3 [9 bodova]

- $N \leq 100\,000$.

- $K \leq 25\,000$.
- Možete koristiti najviše M = 1 500 000 bitova.

Podzadatak 4 [35 bodova]

- $N \leq 5\,000$.
- Možete koristiti najviše M = 10 000 bitova.

Podzadatak 5 [najviše 39 bodova]

- $N \leq 100\,000$.
- $K \leq 25\,000$.
- Možete koristiti najviše M = 1 800 000 bitova.

Broj osvojenih bodova u ovom podzadatku zavisi od dužine R podsetnika koji vaša prva funkcija prosledi. Preciznije, ako je R_{max} maksimalna (od svih test slučajeva) dužina podsetnika proizvedenog pozivanjem funkcije `ComputeAdvice`, broj osvojenih bodova će biti:

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

Detalji implementacije

Potrebno je predati tačno dva fajla *u istom programskom jeziku*.

Prvi fajl treba da ima naziv `advisor.c`, `advisor.cpp` ili `advisor.pas`. Ovaj fajl sadrži implementaciju funkcije `ComputeAdvice` opisane gore i ta funkcija može pozivati funkciju `WriteAdvice`. Drugi fajl treba da ima naziv `assistant.c`, `assistant.cpp` ili `assistant.pas`. Ovaj fajl sadrži implementaciju funkcije `Assist` prema opisu navedenom gore i ona može pozivati funkcije `GetRequest` i `PutBack`.

Opisi tih funkcija imaju sledeći izgled

C/C++ programi

```
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();
```

Pascal programi

```
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;
```

Ove funkcije se moraju ponašati kao što je prethodno opisano. Naravno, vi možete implementirati i druge funkcije za vaše interne potrebe. U C/C++ programima te pomoćne funkcije treba da budu delarisane kao static, pošto program za testiranje (grader) treba da ih poveže zajedno. Ili, izbegavajte da imate funkcije sa istim imenom u različitim programima. Vaše funkcije ne smeju koristiti standardni ulaz/izlaz, a ni druge fajlove.

Pri programiranju vaših rešenja treba voditi računa i o sledećim zahtevima (zaglavlja koja su ostavljeni u vašem takmičarskom okruženju takođe zadovoljavaju zahteve navedene u nastavku).

C/C++ programi

Na početku rešenja treba uključiti fajl `advisor.h` i `assistant.h`, redom u advisor i u assistant. To se postiže dodavanjem u izvorni fajl sledećeg reda:

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

ili

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

Fajlovi `advisor.h` i `assistant.h` će biti smešteni u direktorijum u vašem takmičarskom okruženju, ali će biti ponuđeni i na takmičarskom Web interfejsu. Takođe će (na isti način) biti obezbeđeni kod i skripta za kompajliranje i pokretanje (testiranje) vašeg rešenja. Nakon kopiranja vašeg rešenja i tih skripti u vaš direktorijum, treba da izvršite `compile_c.sh` ili `compile_cpp.sh` (zavisno od jezika u kome je napisano rešenje - kod).

Pascal programi

Treba koristiti module `advisorlib` i `assistantlib`, redom u advisor i u assistant. To se postiže dodavanjem u izvorni program sledećeg reda:

```
uses advisorlib;
```

ili

```
uses assistantlib;
```

Fajlovi `advisorlib.pas` i `assistantlib.pas` će biti smešteni u direktorijum u okviru

vašeg takmičarskog okruženja i takodje će biti ponudjeni u okviru takmičarskog Web interfejsa. Takodje će biti (na ista dva načina) obezbedjen izvorni kod i skripta za prevodjenje i izvršavanje vašeg rešenja. Nakon kopiranja vašeg rešenja u direktorijum na kome se nalazi skripta, trena pokrenuti (izvršiti) `compile_pas.sh`.

Primer programa za testiranje (grejdera)

Program za testiranje će prihvatiti ulazne falove koji imaju sledeći format:

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

Program za testiranje prvo izvršava funkciju `ComputeAdvice`. Pri tome se generiše fajl `advice.txt`, koji sadrži individualne bitove podsetnika, razdvojene prazninama. Na kraju se upisuje cifra 2.

Nakon toga program za testiranje izvršava vašu funkciju `Assist` i generiše izlaz u kojoj je svaki red oblika `"R [broj]"`, ili `"P [broj]"`. Redovi prvog tipa označavaju pozive funkcije `GetRequest()`, broj koji sledi u produžetku je dobijeni rezultat pri pozivu. Redovi drugog tipa označavaju pozive funkcije `PutBack`(), a broj koji sledi je oznaka boje koja se vraća. Izlaz se završava redom koji ima oblik `"E"`.

Imajte na umu da se vreme izvršavanju na zvaničnom grejderu može razlikovati od vremena na vašem lokalnom računaru. Ta razlika ne mora biti značajna. Zato možete koristiti testno okruženje kako biste proverili da li se Vaše rešenje izvršava u zadatom vremenskom ograničenju.

Vremenska i Memorijska ograničenja

- Vremensko ograničenje: 7 sekundi.
- Memorijsko ograničenje: 256 MiB.