Poslednja večera

Leonardo je bilo poprilično zaposlen dok je radio na Posljednjoj Večeri, njegovom najpoznatijem muralu. Jedan od njegovih prvih dnevnih poslova je bio odlučiti koje boje će koristiti tokom ostatka radnog dana. Za rad mu je trebalo mnogo boja, no na skeli na kojoj je stajao mogao je držati samo ograničen broj kantica s bojom. Njegov pomoćnik je, između ostalog, morao i nositi te kantice Leonardu gore na skelu ili skidati odgovarajuće sa skele i nositi ih dolje da bi ih ostavio na neku od polica.

U ovom zadataku, trebate napisati dva odvojena programa da pomognete pomoćniku. Prvi program će dobiti Leonardove zahtjeve (sekvencu boja koje će Leonardo koristiti u toku dana) i napraviti *kratki* niz bita za pomoćnika koji se naziva *savjet*. Za vrijeme opsluživanja Leonardovih zahtjeva u toku dana, pomoćnik neće imati pristup budućim Leonardovim zahtjevima, nego samo savjetu koji je napravio Vaš prvi program. Drugi program će dobiti savjet, a zatim će dobijati i obrađivati Leonardove savjete jedan po jedan. Ovaj program mora biti sposoban da interpretira savjet i iskoristi ga kako bi napravio optimalne izbore. Sve ovo je detaljnije opisano dalje u tekstu.

Nošenje kantica boje između police i skele

U nastavku ćemo pretpostaviti pojednostavljeni scenario. Postoji tačno N boja označenih s brojevima od 0 do N - 1 (uključivo) i Leonardo će svaki dan od pomoćnika tražiti tačno N boja. Neka C označava niz od N zahtjeva za bojom koje Leonardo postavlja. Prema tome C je niz od N brojeva između 0 i N - 1, inkluzivno. Primijetite da neke od boja ne moraju uopšte da se pojave u C, a neke se mogu pojaviti više puta.

Na skelu stane K kantica i uvijek je puna, odnosno uvijek sadži tačno K boja od N mogućih (K < N). U početku skela sadrži boje od 0 do K - 1 (uključivo).

Zaposleni pomoćnik Leonardu nosi kantice jednu po jednu, redom kojim ih on zahtjeva. Ako je boja koju zahtjeva *već na skeli*, pomoćnik ne mora nositi nikakve kantice. Inače, mora pokupiti odgovarajuću kanticu i odnijeti je na skelu. Pošto na skeli nema mjesta za još jednu kanticu, pomoćnik mora uzeti jednu od kantica sa skele i odnjeti je nazad na policu.

Leonardova optimalna strategija

Pomoćniku je u interesu da se što više odmara. Broj Leonardovih zahtjeva za koje neće morati nositi kanticu zavisi od odabira kantica koje skida sa skele tokom dana.

Preciznije, svaki put kada pomoćnik mora da skloni kanticu sa skele, različiti izbori mogu dovesti do različitih rezultata u budućnosti. Leonardo mu je objasnio kako ostvariti svoj cilj znajući C. Najbolji izbor boje koja se mora skinuti sa skele se može dobiti uvidom u skup boja trenutno na

skeli i preostalih zahtjeva za boje u C. Izbor ove boje treba napraviti prema sljedećim pravilima:

- Ako postoji boja na skeli koja neće više potrebna, asistent treba skloniti tu boju sa skele.
- U suprotnom, boja koja se sklanja sa skele treba biti *ona koja će biti potrebna što kasnije u budućnosti*. (Odnosno, za svaku od boja na skeli odredimo trenutak kada će prvi put u budućnosti biti opet potrebna. Ona boja koja je će biti potrebna posljednja se vrača na policu.)

Može se dokazati da će se asistent odmarati onoliko puta koliko je to uopšte moguće ako bude pratio Leonardovu strategiju.

Primjer 1

Neka je N = 4, tj. imamo 4 boje (označene od 0 do 3) i 4 zahtjeva. Neka je niz zahtjeva C = (2, 0, 3, 0). Takođe, neka je K = 2, odnosno na Leonardovoj skeli mogu biti samo dvije kantice u svakom trenutku. U početku su na skeli kantice s bojama 0 i 1, što ćemo označiti sa [0, 1]. Jedan mogući način kako pomoćnik 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 sa skele kanticu sa bojom 1. Trenutno stanje skele je [0, 2].

* Sljedeća boja (boja 0) već na skeli pa se asistent može odmarati.

- Za treći zahtjev (boja 3), asistent skida kanticu s bojom 0 i time mijenja stanje skale na [3, 2].

- Konačno, posljednja tražena boja (boja 0) treba biti donesena sa police gore na skelu. Asistent odlučuje da skine boju 2, i konačno stanje skele je [3, 0].

Primjetite da u gornjem primjeru pomoćnik nije pratio Leonardovu optimalnu strategiju. Optimalna strategija bi bila skinuti kanticu s bojom 2 u trećem koraku, pa bi se pomoćnik u zadnjem koraku mogao odmarati.

Pomoćnikova strategija kada mu je memorija ograničena

Ujutro pomoćnik traži od Leonarda da napiše C brojeva na komad papira, kako bi mogao prema optimalnoj strategiji odrediti plan rada. Problem je što Leonardo želi sačuvati svoje vještine tajnima i zato odbija dati pomoćniku papir s C brojeva. Kao rješenje, on će njemu pročitati tih C brojeva i pomoćnik će ih probati zapamtiti.

Nažalost, asistentovo pamćenje je veoma loše. Ono što on može zapamtiti je najviše M bitova informacija. Generalno, ovo bi ga moglo spriječiti da rekonstruiše cijeli niz C i zato mora smisliti neki pametan način kako će ga rekonstruisati. Ovaj niz od M bitova nazivamo *nizom savjeta* i označićemo ga sa A.

Primjer 2

Ujutro, pomoćnik može uzeti Leonardov papir sa sekvencom C, pročitati sekvencu i napraviti sve potrebne izbore. Jedna stvar koju može izabrati da uradi jeste da ustanovi stanje skele nakon svakog od zahtjeva. Na primjer, kada koristi (podoptimalnu) strategiju datu u Primjeru 1, sekvenca stanja skele je [0, 2], [0, 2], [3, 2], [3, 0]. (Sjetite se da on zna da je početno stanje skele [0, 1].)

Pretpostavimo sada da imamo M = 16, tako da asistent može da zapamti do 16 bita informacija. Kako je N = 4, možemo spremati svaku boju koristeći 2 bita. Dakle, 16 bita je dovoljno da se spremi navedeni niz stanja skele. Prema tome, asistent računa sljedeći niz savjeta: A = (`0`, `0`, `1`, `0`, `0`, `0`, `1`, `0`, `1`, `1`, `1`, `0`, `1`, `1`, `0`, `0`).

Kasnije u toku dana, asistent može da dekodira ovaj niz savjeta i koristi ga za svoje izbore.

(Naravno, sa M = 16 asistent takođe može da odluči da zapamti cijelu sekvencu C, koristeći samo 8 od mogućih 16 bita. U ovom primjeru samo smo željeli ilustrovati da on može imati i druge mogućnosti, bez odavanja bilo kog dobrog rješenja.)

Postavka

Trebate napisati *dva različita programa* u istom programskom jeziku. Ovi programi će biti pokrenuti jedan poslije drugog bez mogućnosti da komuniciraju jedan sa drugim za vrijeme izvršavanja.

Prvi program će koristiti asistenti ujutru. Ovom programu će biti dat niz C, a on mora da izračuna niz savjeta A.

Drugi program će asistent koristiti tokom dana. Ovaj program će dobijati niz savjeta A, i onda će morati da obradi sekvencu C Leonardovih zahtjeva. Primijetite da će sekvenca C biti otkrivena programu samo jedan po jedan zahtjev i svaki zahtjev mora biti obrađen prije dobijanja sljedećeg.

Preciznije, u prvom programu treba da implementirate jednu rutinu `ComputeAdvice`(C, N, K, M) pri čemu za ulaz imate niz C od N cjelobrojnih vrijednosti (svaka iz skupa 0, …, N - 1), broj K boja na skeli, i broj bita M dostupnih za savjet. Ovaj program mora da izračuna niz savjeta A koji se sastoji od najviše M bita. Program treba da komunicira niz A sistemu pozivanjem, za svaki bit iz A redom, sljedeću rutinu:

- `WriteAdvice`(B) — dodaje bit B trenutnom nizu savjeta A. (I dalje možete pozvati ovu rutinu najviše M puta).

U drugom programu treba da implementirate jednu rutinu `Assist`(A, N, K, R). Ulaz za ovu rutinu je niz savjeta A, cjelobrojne vrijednosti N i K kako je definisano gore, i stvarnu dužinu R niza A u bitima (R ≤ M). Ova rutina treba da izvrši predloženu strategiju za asistenta, koristeći sljedeće rutine koje su Vam obezbijeđene:

- `GetRequest`() — vraća sledeću boju koju je tražio Leonardo. (Nikakve informacije o budućim zahtjevima se ne otkrivaju.)

- `PutBack`(T) — stavlja boju T sa skele nazad na policu. Možete pozvati ovu rutinu samo ako je T jedna od boja koje su trenutno na skeli.

Kada se izvrši, Vaša rutina `Assist` mora pozvati `GetRequest` tačno N puta, svaki put dobijajući jedan od Leonardovih zahtijeva, redom. Nakon svakog poziva `GetRequest`, ako vraćena boja *nije* na skeli, *morate* takođe pozvati i `PutBack`(T) sa svojim izborom T. U suprotnom, *ne smijete* pozvati `PutBack`. Ako ne uradite tako, to će se smatrati greškom i izazvaće

terminaciju Vašeg programa. Prisjetimo se da na početku skela sadrži boje od 0 do K - 1, uključivo.

Testni slučaj će se smatrati riješenim ako Vaše dvije rutine prate sva zadata ograničenja, a ukupan broj poziva `PutBack` je *tačno* jednako Leonardovoj optimalnoj strategiji. Primijetite da ako postoji više različitih strategija da se postigne isti broj poziva `PutBack`, Vašem programu je dozvoljeno da izvrši bilo koju od njih. (Na primjer, nije potrebno da se prati Leonardova strategija, ako postoji neka druga, jednako dobra, strategija.)

Primjer 3

Nadovezujući se na primjer 2, pretpostavimo da ste u `ComputeAdvice` izračunali da je A = (`0`, `0`, `1`, `0`, `0`, `0`, `1`, `0`, `1`, `1`, `1`, `0`, `1`, `1`, `0`, `0`). Kako biste to komunicirali sistemu, morate napraviti sljedeći 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)`.

Vaša druga rutina `Assist` bi onda bila izvršena, dobijajući gore navedeni niz A, i vrijednosti N = 4, K = 2, i R = 16. Rutina `Assist` onda mora da izvrši tačno N = 4 poziva `GetRequest`. Takođe, nakon nekih od ovih zahtijeva, `Assist` će morati da pozove `PutBack`(T) sa odgovarajućim izborom T.

Tabela ispod prikazuje niz poziva koji odgovaraju (suboptimalnim) izborima iz primjera 1. Crtica označava da nema poziva `PutBack`.

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

Podzadatak 1 [8 bodova]

- $N \leq 5\,000$.
- Možete korisiti najviše M = 65 000 bita.

Podzadatak 2 [9 bodova]

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

Podzadatak 3 [9 bodova]

- $N \leq 100\,000$.

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

Podzadatak 4 [35 bodova]

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

Podzadatak 5 [do 39 bodova]

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

Bodovanje za ovaj zadatak zavisi od dužine R savjeta koji program komunicira. Preciznije, ako je R_{max} maksimum (od svih testnih slučajeva) dužine niza savjeta generisane od strane rutine `ComputeAdvice`, Vaš rezultat će biti:

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

Detalji implementacije

Trebate predati tačno dvije datoteke *u istim programskim jezicima*.

Prva datoteka treba da se zove `advisor.c`, `advisor.cpp` ili `advisor.pas`. Ova datoteka mora implementirati rutinu `ComputeAdvice` kako je opisano iznad i može pozvati rutinu `WriteAdvice`. Naziv druge datoteke je `assistant.c`, `assistant.cpp` ili `assistant.pas`. Ova datoteka mora implementirati rutinu `Assist` kako je opisano iznad i može pozvati rutine `GetRequest` i `PutBack`.

Potpisi za sve rutine slijede.

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 trebaju da se ponašaju u skladu sa ranije datim opisima. Naravno, možete implementirati i druge pomoćne funkcije. Za C/C++ programe, Vaše pomoćne funkcije bi trebale da budu deklarisane kao statičke(`static`) inače će ih probni tester povezati. Alternativno, možete jednostavno izbjegavati da nazivate dvije funkcije (u dva različita programa) istim imenom. Vaš program ne smije ni na koji način koristiti standardni ulaz i izlaz niti bilo koju drugu datoteku.

Dok programirate svoje rješenje, morate voditi računa o sljedećim instrukcijama (uzorci koje možete naći u svom takmičarskom okruženju već zadovoljavaju zahtjeve izlistane ispod).

C/C++ programi

Na početku Vašeg rješenja, morate uključiti datoteke `advisor.h` i `assistant.h` respektivno, u savjetodavcu i u asistentu. Ovo se izvodi uključivanjem sljedeće linije u Vaš kod:

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

ili

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

Dvije datoteke `advisor.h` i `assistant.h` bit će Vam dostupne u folderu unutar takmičarskog okruženja i bit će dostupne na takmičarskom Web interfejsu. Osim toga, bit će Vam dostupan (na isti način) kod i skripte za kompajliranje i testiranje Vašeg rješenja. Konkretno, nakon kopiranja rješenja u folder sa skriptama, pokrenite `compile_c.sh` ili `compile_cpp.sh` (ovisno od programskog jezika koji koristite).

Pascal programi

Trebate koristiti jedinice `advisorlib` i `assistantlib` za savjetnika i pomoćnika, respektivno. Ovo se postiže dodavanjem sljedećih linija u vaš kod:

```
uses advisorlib;
```

ili

```
uses assistantlib;
```

Dvije datoteke `advisorlib.pas` i `assistantlib.pas` bit će Vam dostupne u folderu

unutar takmičarskog okruženja i bit će dostupne na takmičarskom Web interfejsu. Osim toga, bit će Vam dostupan (na isti način) kod i skripte za kompajliranje i testiranje Vašeg rješenja. Konkretno, nakon kopiranja rješenja u folder sa skriptama, pokrenite `compile_pas.sh`.

Probni tester

Probni tester će prihvatati ulazne podatke kako slijedi:

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

Tester će najprije izvršiti funkciju `ComputeAdvice`. Ovo će napraviti datoteku `advice.txt` koja sadrži pojedinačne bite savjetovane sekvence međusobno odvojene razmacima i terminirane brojem 2.

Nakon toga, izvršite će funckiju `Assist` i napraviti izlaz kod kojeg je svaki red u jednom od oblika "`R [number]`" ili "`P [number]`". Redovi prvog oblika ukazuju na pozive funkcije `GetRequest()` i dobijene odgovore. Redovi drugog oblika predstavljaju pozive `PutBack`() i boje koje su izabrane da se vrate. Izlaz je terminiran redom oblika "`E`".

Molimo Vas da vodite računa da na službenom testeru vrijeme izvršavanja može biti donekle različito nego na Vašem lokalnom računaru. Ova razlika ne bi trebala biti značajna. Ipak, preporučujemo da koristite testni interfejs da provjerite da li će vaš program raditi unutar vremenskog ograničenja.

Vremenaska i memorijska ograničenja

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