Posljednja večera

Leonardo je bio poprilično zaposlen dok je radio na Posljednjoj večeri, svojem najpoznatijem muralu. Jedan od njegovih prvih poslova svaki dan bio je odlučiti koje boje će koristiti ostatak radnog dana. Za rad mu je trebao mnogo boja, no na skeli na kojoj je stajao mogao je držati samo ograničen broj kantica s bojom. Njegov pomoćnik je među ostalom morao i nositi te kantice Leonardu gore na skelu ili skidati odgovarajuće sa skele i nositi ih dolje da bi ih spremio na neku od polica.

U ovom zadatku morate napisati dva odvojena programa koja će pomoći pomoćniku. Prvi program će primiti čitav niz Leonardovih zahtjeva (boje koje će trebati tokom dana) i mora napraviti *kratki* niz bitova koji se zove *savjet*. Dok pomoćnik obrađuje Leonardove zahtjeve tokom dana, on neće imati pristup Leonardovim budućim zahtjevima, već samo savjetu koje je vaš prvi program generirao. U drugom programu ćete primiti savjet, kao i Leonardove zahtjeve koje morate rješavati jedan po jedan (online). Taj program mora razumijeti što savjet znači kako bi donosio optimalne odluke. U nastavku je sve detaljno objašnjeno.

Nošenje kantica boje između police i skele

U nastavku ćemo pretpostaviti pojednostavljeni scenario. Postoji točno N boja označenih s brojevima od 0 do N - 1 (uključivo) i Leonardo će svaki dan od pomoćnika tražiti točno 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. Naravno, Leonardo može neku boju tražiti više puta, a neku nijednom.

Na skelu stane K kantica i uvijek je puna, odnosno uvijek sadži točno 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 odnjeti 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

Pomoćnik bi se htio što više puta odmarati, tj. ne nositi kantice. Broj zahtjeva za koje će se moći odmarati ovisi o nizu odluka koje donese tokom dana. Točnije, svaki put kad pomoćnik skida kanticu sa skele, različiti odabiri vode do različitih ishoda u budućnosti. Leonardo mu je zato objasnio kako može postići svoj cilj znajući C. Kada spušta neku kanticu sa skele, najbolji odabir se može dokučiti promatrajući budući niz zahtjeva i koje su boje trenutno na skeli. Kantica koju će spustiti treba odabrati prema sljedećim pravilima:

- Ako ima kantica s bojom na skeli koju Leonardo više neće trebati, pomoćnik treba skinuti tu kanticu sa skele.
- Inače, mora skinuti kanticu sa bojom koju će Leonardo sljedeći put trebati *najkasnije u budućnosti*. (Odnosno za svaku boju dostupnu na skeli pogledamo kad će bit zatražena sljedeći put. Ona koju skinemo je ona koja će najkasnije biti potrebna.)

Može se dokazati da ako se koristi Leonardova strategija, pomoćnik će se odmarati najveći mogući broj puta.

Prvi primjer

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). Također, 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 s [0, 1]. Jedan mogući način kako pomoćnik može odraditi svoj posao za taj dan je opisan u nastavku.

- Prva tražena kantica (boja 2) nije na skeli, stoga je pomoćnik nosi gore i sa skele skida kanticu s bojom 1. Trenutno stanje skele je [0, 2].
- Sljedeća kantica (boja 0) već na skeli pa se pomoćnik može osloboditi.
- Za treći zahtjev (boja 3), asistent skida kanticu s bojom 0 i time mijenja stanje skele na [3, 2].
- Konačno, zadnju traženu kanticu (boja 0) treba donjeti sa police gore na skelu. Pomoćnik odluči skinuti 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 uz ograničeno pamćenje

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, nije pomoćnik kriv što je pomoćnik i njegovo pamćenje je jako loše. Ono što on može zapamtiti je najviše M bitova informacija. Ovo bi ga općenito ovo moglo spriječiti da rekonstruira cijeli niz C i zato mora smisliti neki pametan način kako će ga rekonstruirati. Ovaj niz od M bitova nazivamo *savjetnim nizom* i označit ćemo ga s A.

Drugi primjer

Ujutro, pomoćnik uzima Leonardov papir s C brojeva koje pročita nakon čega može donijeti sve odluke. Primjerice, ono što može napraviti je proučiti stanje skele nakon svakog zahtjeva. Koristeći suboptimalnu strategiju iz prvog primjera, niz stanja skele bio bi [0, 2], [0, 2], [3, 2], [3, 0]. (Prisjetite se da je u početku stanje skele [0, 1].)

Sad pretpostavite da je M = 16, odnosno pomoćnik može zapamtiti 16 bitova informacije. Pošto je

N = 4, svaku boju možemo opisati s 2 bita. Prema tome 16 bitova su dovoljni za zapametiti gornji niz stanja skele. Prema tome, pomoćnik će zapamtiti sljedeći niz bitova: A = (0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0).

Kasnije taj dan, pomoćnik može dekodirati ovaj savjetni niz i koristiti ga da donese svoje odluke.

(Naravno, sa M = 16, pomoćnik također može zapamtiti cijeli niz C, koristeći samo 8 od 16 bitova na raspolaganju. Cilj ovog primjera je pokazati kako on ima više mogućih pristupa rješenju bez odavanja ijednog dobrog rješenja.)

Zadatak

Vaš je zadatak napisati *dva odvojena programa* u istom programskom jeziku. Ovi programi će se pokrenuti jedan nakon drugog (serijski), bez mogućnosti da komuniciraju jedan s drugim tokom izvođenja.

Prvi program je onaj koji će pomoćnik koristiti ujutro. Ovaj program prima niz C i mora izračunati savjetni niz A.

Drugi program je onaj koji će pomoćnik koristiti tokom dana. Program prima čitav savjetni niz A i mora obraditi niz Leonardovih zahtjeva, odnosno niz C. Stvar je u tome što će niz C biti otkriven programu jedan po jedan zahtjev, i svaki mora biti obrađen prije primanja sljedećeg.

Točnije, prvi program mora implementirati jednu funkciju `ComputeAdvice`(C, N, K, M) koja prima C, niz od N brojeva od 0 do N - 1, najveći broj kantica na skeli K i broj bitova M dostupnih za savjetni niz. Ovaj program mora konstruirati savjetni niz A koji koristi najviše M bitova. Program mora testnom sustavu poslati niz A tako da za svaki bit redom zove sljedeću funkciju:

- `WriteAdvice`(B) — dodaj bit B na kraj trenutnog savjetnog niza A. (Ovu funkciju možete zvati najviše M puta.)

U drugom programu morate implementirati jednu funkciju `Assist`(A, N, K, R). Funkcija prima savjetni niz A, gore definirane brojeve N i K, te duljinu R savjetnog niza A (R ≤ M). Ova funkcija mora izvršiti vašu strategiju za pomoćnika, koristeći sljedeće funkcija koje su vam dostupne:

- `GetRequest`() — vraća sljedeću boju koju Leonardo treba. (Nikakve informacije o kasnijim zahtjevima neće biti odane.)
- `PutBack`(T) — vrati kanticu s bojom T sa skele na policu. T mora biti boja koja je trenutno na polici.

Jednom pokrenuta, vaša funkcija `Assist` mora zvati `GetRequest` točno N puta. Svaki poziv će vratiti jedan Leonardov zahtjev, redom kojim ih postavlja. Ako nakon poziva funkcije `GetRequest`, vraćena boja *nije* na skeli, onda *morate* pozvati `PutBack`(T) sa odgovarajućim odabirom boje T. U suprotnom *ne smijete* zvati funkciju `PutBack`. Ne pridržavanje toga smatra se greškom i izazvat će prekid izvršavanja vašeg programa. Prisjetite se da su na početku na skeli kantice s bojama od 0 do K - 1 (uključivo).

Neki test slučaj se smatra točno rješenim ako se vaše dvije funkcije pridržavaju gore navedenim

ograničenjima, i ukupan broj poziva funkciji `PutBack` je *jednak* onom od Leonardove optimalne strategije. Naravno, može postojati više optimalnih strategija, a vaš program može implementirati bilo koju od njih. (Odnosno nije potrebno koristiti Leonardovu strategiju ako postoji neka druga također optimalna.)

Treći primjer

U nastavku s drugim primjerom, pretpostavite da je vaša funkcija `ComputeAdvice` konstruirala niz A = (`0`, `0`, `1`, `0`, `0`, `0`, `1`, `0`, `1`, `1`, `1`, `0`, `1`, `1`, `0`, `0`). Kako bi to javili sustavu, vaša funkcija mora 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)`.

Nakon toga, vaša druga funkcija `Assist` će biti pozvana i primit će gornji niz A, kao i vrijednosti N = 4, K = 2 i R = 16. `Assist` tada mora obaviti točno N = 4 poziva funkciji `GetRequest` kako bi dobili zahtjev od Leonarda. Također, nakon nekih od tih zahtjeva, `Assist` mora pozvati `PutBack`(T) sa odgovarajućim odabirom boje T.

Tablica koja slijedi prikazuje niz funkcijskih poziva koji odgovaraju suboptimalnom nizu odabira iz prvog primjera. Znak crtice označava da nema poziva funkciji `PutBack`.

`GetRequest()`	Action
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 [do 39 boda]

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

Broj bodova za ovaj podzadatak ovisi o duljini R savjetnog niza kojeg vaš program konstruira. Točnije, ako je R_{max} najveća duljina savjetnog niza nad skupom test slučajeva, dobit ćete:

- 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 $R_{max} \geq 1\,800\,000$.

Implementacija

Morate predati točno dvije datoteke sa odgovarajućim funkcijama *u istom programskom jeziku*.

Prva datoteka se mora zvati `advisor.c`, `advisor.cpp` ili `advisor.pas`. Ta datoteka mora implementirati funkciju `ComputeAdvice` kao što je gore opisano, i može zvati funkciju `WriteAdvice`. Druga datoteka se mora zvati `assistant.c`, `assistant.cpp` ili `assistant.pas`. U njoj morate implementirati funkciju `Assist` kao što je gore opisano, i može zvati funkcije `GetRequest` i `PutBack`.

Slijede deklaracije funkcija.

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;
```

Vaše funkcije se moraju ponašati kao što je gore opisano. Naravno, možete slobodno implementirati druge interne pomoćne funkcije. Za C/C++ programe, vaše pomoćne funkcije bi trebale biti deklarirane kao static, pošto će ih sustav za testiranje povezati (linkati) zajedno. Alternativa je da su sve pomoćne funkcije nazivete različitim imenima (za oba programa skupa!). Ne smijete koristiti standardni ulaz ili izlaz, kao i pisanje ili čitanje po bilo kakvim datotekama.

Dok programirate, također pripazite na sljedeće (predlošci koji su vam na raspolaganju ovo već zadovoljavaju).

C/C++ programi

Na početku vašeg rješenja, morate uključiti datoteke `GetRequest` u prvom programu (savjetnik) i `PutBack` u drugome (pomoćnik). To možete postići kako slijedi:

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

ili

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

Datoteke `advisor.h` i `assistant.h` su vam na raspolaganju u natjecateljskoj okolini i možete ih skinuti sa weba. Također su vam na raspolaganju (na isti način) programi za testiranje i prevođenje vaših rješenja. Točnije, nakon što kopirate svoje rješenje u direktorij sa ovim skriptama, možete pokrenuti `compile_c.sh` or `compile_cpp.sh` (ovisno o jeziku).

Pascal programi

You have to use the units `advisorlib` and `assistantlib`, respectively, in the advisor and in the assistant. This is done by including in your source the line:

```
uses advisorlib;
```

ili

```
uses assistantlib;
```

The two files `advisorlib.pas` and `assistantlib.pas` will be provided to you in a directory inside your contest environment and will also be offered by the contest Web interface.

You'll also be provided (through the same channels) with code and scripts to compile and test your solution. Specifically, after copying your solution into the directory with these scripts, you will have to run `compile_pas.sh`.

Primjer sustava za testiranje

Sustav za testiranje prima ulaz kako slijedi:

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

Sustav će prvo izvršiti funkciju `ComputeAdvice`. Ovo će generirati datoteku `advice.txt`, koja sadrži bitove savjetnog niza odvojene razmacima i terminirane s brojem 2.

Zatim će pokrenuti vašu funkciju `Assist` i generirati će izlaz gdje će svaka linija biti u formi "`R [broj]`" ili "`P [broj]`". Linije prvog tipa označavaju poziv funkciji `GetRequest()` zajedno s odgovorom koje dobije. Linije drugog tipa označavaju poziv funkciji `PutBack`() zajedno s brojem boje skinute sa skele. Izlaz je terminiran linijom "`E`".

Imajte na umu da će na službenom sustavu vrijeme izvođenja vjerojatno biti drugačije od onog na vašem lokalnom računalu. Ova razlika ne bi trebala biti velika. Svejedno, slobodno možete koristiti test sučelje na webu da provjerite da vaše rješenje radi dovoljno brzo.

Vremenska i memorijska ograničenja

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