Poruka

Lamija i Vedran su prijatelji koji međusobno komuniciraju na neobičan način. Lamija ima poruku M, koja je zapravo niz od S bita (niz nula i jedinica), koju želi poslati Vedranu. Lamija komunicira s Vedranom šaljeći mu **pakete**. Paket je niz od 31 bita indeksiran od 0 do 30. Lamija želi iskomunicirati poruku M Vedranu šaljeći mu pakete.

Nažalost, Lamija u školi ima ljubomorne prijateljice koje žele upropastiti komunikaciju između Lamije i Vedrana. One metodama špijunaže, lukavstva i prepredenosti mogu promjeniti određenih 15 pozicija u paketu. Formalnije, postoji niz C dužine 31 u kojem je svaki element 0 ili 1, koji opisuje vragolije koje prijateljice izvode na sljedeći način:

- $C[i] = 1$ znači da je poziciju i u paketu **kontrolišu** prijateljice, te je mogu, ali ne moraju promjeniti po želji.
- $C[i] = 0$ znači da pozicija i nije pod utjecajem prijateljica, te ostaje onako kako je Lamija poslala.

Niz C sadrži **tačno** 15 jedinica i 16 nula. Tokom slanja jedne poruke M (kroz više paketa), pozicije koje kontrolišu prijateljice ostaju iste kroz sve pakete. Lamija zna tačno kojih 15 pozicija su riskantne, dok Vedran samo zna da je takvih pozicija 15, ali ne zna koje su.

Neka je A paket koji Lamija odluči poslati (još zvan i **originalni paket**). Neka je B paket koji prima Vedran (još zvan i **pobrkani paket**). Za svaki i, takav da $0 \leq i < 31$:

- ako prijateljice ne kontrolišu poziciju i ($C[i] = 0$), Vedran prima bit i onako kako ga je Lamija poslala ($B[i] = A[i]$),
- inače, ako prijateljice kontrolišu poziciju i ($C[i] = 1$), vrijednosti $B[i]$ odlučuju one.

Čim Lamija pošalje paket ona sazna i u koji paket su ga prijateljice promijenile.

Nakon što Lamija pošalje sve pakete, Vedran odjednom dobija sve pobrkane pakete **u istom onom poretku kako ih je Lamija i slala**. On tada pokušava rekonstruirati poruku M.

Vaš je zadatak dizajnirati i implementirati strategiju koja bi dozvolila da Lamija paketima pošalje poruku M, tako da Vedran može rekonstruisati poruku M iz pobrkanih paketa. Tačnije, trebate implementirati dvije procedure. Prva procedura izvodi radnje koje čini Lamija. Procedura dobija poruku M i niz C, te treba poslati neke pakete kako bi prenjela poruku Vedranu. Druga procedura treba imitirati Vedranov proces razmišljanja kojim će nakon šoka i nevjerice pokušati iz pobrkanih paketa koje dobije rekonstruirati poruku M.

Detalji implementacije

Prva procedura glasi:

```
void send_message(std::vector<bool> M, std::vector<bool> C)
```

- M: niz S bita koji opisuje poruku koju Lamija pokušava slati Vedranu.
- C: niz 31 bita označava pozicije koje kontroliraju prijateljice.
- Ova procedura će biti pozvana **najviše 2100 puta** u svakom testnom primjeru.

Ta procedura treba pozivati sljedeću za slanje paketa:

```
std::vector<bool> send_packet(std::vector<bool> A)
```

- A: originalni paket (niz 31 bita).
- Ova procedura vraća pobrkani paket B.
- Ova procedura smije biti pozvana najviše 100 puta tokom jednog poziva `send_message`.

Druga procedura za implementaciju je:

```
std::vector<bool> receive_message(std::vector<std::vector<bool>> R)
```

- R: niz koji sadrži pobrkane pakete. Paketi su porijeklom nastali od paketa koje je Lamija poslala tokom nekog poziva `send_message` procedure **i u tačnom poretku**. Svaki član niza R je niz 31 bita, predstavlja pobrkani paket.
- Ova procedura treba vraćati niz S bita koji bi trebao biti jednak originalnoj poruci M.
- Ova procedura će biti pozvana **više puta** u svakom testnom primjeru, **tačno jednom** za svaki `send_message` poziv. **Redoslijed** `receive_message` poziva nije nužno isti kao onaj `send_message` poziva.

Napomenimo da će `send_message` i `receive_message` procedure biti pozvane u **dva odvojena programa** tokom gradinga.

Ograničenja

- $1 \leq S \leq 1024$
- C ima tačno 31 elemenata, od kojih su 16 jednaki 0, a 15 su jednaki 1.

Podzadaci i ocjenjivanje

Ako u nekom testnom primjeru pozivi procedure `send_packet` nisu u skladu s pravilima napomenutima gore, ili je izlaz procedure `receive_message` pogrešan dobit ćete 0 bodova za taj testni primjer.

Inače, neka je Q maksimalni broj poziva procedure `send_packet` među svim pozivima `send_message` među svim test podacima. Neka je nadalje X jednak:

- 1, if $Q \leq 66$
- 0.95^{Q-66}, if $66 < Q \leq 100$

Tada je broj bodova ostvaren po sljedećem kriteriju:

Podzadatak	Bodovi	Dodatna ograničenja
1	$10 \cdot X$	$S \leq 64$
2	$90 \cdot X$	Nema dodatnih ograničenja.

Napomenimo da se grader u nekim slučajevima ponaša **adaptivno**. To znači da pobrkani paketi koje šalje `send_packet` ovise o ulaznim argumentima, ali potencijalno i mnogim drugim stvarima, uključujući i prethodnim pozivima ove procedure i pseudo-nasumičnim brojevima koje generiše grader. Grader je **determinističan** u smislu da pokretanjem dva puta sa istim paketima grader će napraviti iste promjene nad njima.

Primjer

Razmotrimo sljedeći primjer.

```
send_message([0, 1, 1, 0],
             [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
              1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1])
```

Poruka koju Lamija pokušava da pošalje Vedranu je $[0, 1, 1, 0]$. Bitovi na pozicija od 0 do 15 ne mogu da budu promjenjeni od strane prijateljica, dok oni na pozicijama od 16 do 30 mogu.

Radi jednostavnosti, pretpostavimo da će prijateljice popunjavati pozicije koje kontrolišu alternirajući sa 0 i 1, tj. stavit će 0 na prvu poziciju koju kontroliraju (pozicija 16 u ovom slučaju), 1 na drugu (pozicija 17), 0 na treću koju kontroliraju (pozicija 18), i tako dalje.

Lamija će svakim paketom slati 2 bita iz originalne poruke na sljedeći način: prvi od njih zapisat će na svih prvih 8 pozicija koje kontrolira te drugi na preostalih 8.

Dakle ovako:

```
send_packet([0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1,
             0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0])
```

Primjetimo da pošto prijateljice nasumice mogu promjeniti zadnjih 15 pozicija, nije bitno što Lamija tamo pošalje kako one mogu biti prebrisane. Sa pretpostavljenom strategijom pobrkani paket izgleda ovako: $[0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1]$.

Lamije sada odlučuju poslati posljednja dva bita poruke M u drugom paketu na sličan način:

```
send_packet([1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0,
             0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0])
```

Pobrkani paket sada glasi: $[1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0]$.

Lamija može da šalje još, ali ne želi.

Grader sada izvrši sljedeći poziv:

```
receive_message([[0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1,
                  0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0],
                 [1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0,
                  0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0]])
```

Vedran rekonstruiše poruku M sa sljedećom strategijom. Od svakog paketa uzme prvi bit koji se pojavljuje dva puta za redom i zadnji takav. Odnosno, od prvog pobrkanog paketa dobija $[0,1]$, a iz drugog $[1,0]$. Spajajući dobija poruku $[0,1,1,0]$, Što je i tačan izlaz za proceduru `receive_message`.

Može se pokazati da sa ovom strategijom za poruke dužine 4 Lamija i Vedran će uvijek se sporazumjeti, neovisno od toga kako prijateljice pobrkaju pakete. Ova strategija nije uspješna u generalnom slučaju.

Sample Grader

Sample grader nije adaptivan kao onaj pravi. Umjesto toga, prijateljice se ponašaju deterministički, uvijek će popunjavati svoje pozicije alternirajući s 0 i 1, kako je opisano gore (čak i ako nisu uzastopne).

Ulazni format: **Prva linija sadrži broj T, broj scenarija.** T scenarija sljedi. Svaki od scenarija izgleda ovako:

```
S
M[0]  M[1]  ...  M[S-1]
C[0]  C[1]  ...  C[30]
```

Izlazni format: Sample grader zapisuje svaki od T scenarija redoslijedom kako su navedeni u ulazu ovako:

```
K L
D[0]  D[1]  ...  D[L-1]
```

Ovdje je, K broj poziva `send_packet`, D je poruka koju je vratio poziv od `receive_message` i L je njena dužina.