Viesti

Aisha ja Basma ovat ystäviä, jotka viestittelevät keskenään. Aishalla on viesti M, joka on S bittiä (eli nollia tai ykkösiä) sisältävä jono, jonka hän haluaisi lähettää Basmalle. Aisha kommunikoi Basman kanssa lähettämällä hänelle **paketteja**. Paketti on 31 bitin jono, joka on indeksoitu 0:sta 30 asti. Aisha haluaisi lähettää viestin M Basmalle lähettämällä hänelle jonkin määrän paketteja.

Valitettavasti Cleopatra sai pääsyn Aishan ja Basman välisen viestintään ja pystyy **pilaamaan** paketit. Toisin sanoen jokaisessa paketissa Cleopatra voi muokata bittejä tarkalleen 15 indeksissä. Tarkemmin sanottuna on taulukko C jonka pituus on 31, jossa jokainen alkio on joko 0 tai 1, jolla on seuraava merkitys:

- $C[i] = 1$ tarkoittaa, että Cleopatra voi muuttaa bittiä indeksillä i. Kutsumme näitä indeksejä **Cleopatran hallitsemiksi**.
- $C[i] = 0$ tarkoittaa, että Cleopatra ei voi muuttaa bittiä indeksillä i.

Taulukko C sisältää täsmälleen 15 ykköstä ja 16 nollaa. Viestiä lähetettäessä Cleopatran hallitsema indeksien joukko pysyy samana kaikille paketeille. Aisha tietää tarkasti, mitä 15 indeksejä Cleopatra hallitsee. Basma tietää vain, että 15 indeksiä on Cleopatran hallinnassa, mutta hän ei tiedä mitkä indeksit.

Olkoon A paketti, jonka Aisha lähettää (jota kutsumme **alkuperäiseksi paketiksi**). Olkoon B Basman vastaanottama paketti (jota kutsumme **pilatuksi paketiksi**). Jokaiselle i:lle siten, että $0 \leq i < 31$:

- jos Cleopatra ei hallitse bittiä indeksillä i ($C[i] = 0$), Basma vastaanottaa bitin i Aishan lähettämänä ($B[i] = A[i]$),
- muuten, jos Cleopatra hallitsee bittiä indeksillä i ($C[i] = 1$), $B[i]$:n arvon päättää Cleopatra.

Välittömästi jokaisen paketin lähettämisen jälkeen Aisha saa tietää, mikä on alkuperäistä pakettia vastaava pilattu paketti.

Kun Aisha on lähettänyt kaikki paketit, Basma vastaanottaa kaikki pilatut paketit **lähetysjärjestyksessä** ja hänen täytyy muodostaa alkuperäinen viesti M.

Sinun tehtäväsi on suunnitella ja toteuttaa strategia, jonka avulla Aisha voisi lähettää viestin M Basmalle, jotta Basma voi palauttaa viestin M pilatuista paketeista. Tarkemmin ottaen sinun tulee toteuttaa kaksi funktiota. Ensimmäinen funktio suorittaa Aishan strategian. Sille annetaan viesti M ja taulukko C, ja sen tulee lähettää jokin määrä paketteja viestin välittämiseksi Basmalle. Toinen

funktio suorittaa Basman strategian. Sille annetaan pilatut paketit ja sen pitäisi palauttaa alkuperäinen viesti M.

Toteutuksen yksityiskohdat

Ensimmäinen funktio, joka sinun tulee toteuttaa on:

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

- M: taulukko, jonka pituus on S, joka sisältää viestin, jonka Aisha haluaa lähettää Basmalle.
- C: taulukko, jonka pituus on 31, joka sisältää Cleopatran hallitsemien bittien indeksit.
- Tätä funktiota voidaan kutsua **korkeintaan 2100 kertaa** kussakin testitapauksessa.

Tämän funktion pitäisi kutsua seuraavaa funktiota paketin lähettämiseksi:

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

- A: alkuperäinen paketti (taulukko, jonka pituus on 31), joka sisältää Aishan lähettämät bitit.
- Tämä funktio palauttaa pilatun paketin B, joka sisältää Basman vastaanottamat bitit.
- Tätä funktiota voidaan kutsua enintään 100 kertaa jokaisessa `send_message` -kutsussa.

Toinen funktio, joka sinun tulee toteuttaa, on:

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

- R: taulukko, joka sisältää pilatut paketit. Paketit ovat Aishan yhdessä `send_message` kutsussa lähettämistä paketeista ja Aisha antaa ne **lähetysjärjestyksessä**. Jokainen R:n alkio on taulukko, jonka pituus on 31 ja joka kuvaa pilattua pakettia.
- Tämän funktion tulee palauttaa S bittiä sisältävä taulukko, joka on yhtä suuri kuin alkuperäinen viesti M.
- Tätä funktiota voidaan kutsua **useita kertoja** kussakin testitapauksessa, **täsmälleen kerran** jokaista vastaavaa `send_message` -kutsua kohden. **funktion** `receive_message` **kutsujen** järjestys ei välttämättä ole sama kuin vastaavien `send_message` -kutsujen järjestys.

Huomaa, että testijärjestelmässä `send_message` ja `receive_message` -funktioita kutsutaan **erillisissä ohjelmissa**.

Rajat

- $1 \leq S \leq 1024$
- C sisältää täsmälleen 31 alkiota, joista 16 ovat 0 ja 15 ovat 1.

Osatehtävät ja pisteytys

Jos jossakin testitapauksessa, kutsut funktiolle `send_packet` eivät ole edellä mainittujen sääntöjen mukaisia, tai minkä tahansa `receive_message` kutsun palautusarvo on virheellinen, ratkaisusi pistemäärä kyseisessä testitapauksessa on 0.

Muussa tapauksessa olkoon Q funktion `send_packet` kutsujen enimmäismäärä kaikkien `send_message` -kutsujen joukossa kaikissa testitapauksissa. Olkoon myös X yhtä suuri kuin:

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

Tällöin pisteet lasketaan seuraavasti:

Osatehtävä	Pisteet	Lisäehdot
1	$10 \cdot X$	$S \leq 64$
2	$90 \cdot X$	Ei lisäehtoja.

Huomaa, että joissakin tapauksissa testijärjestelmän käyttäytyminen voi olla **mukautuvaa**. Tämä tarkoittaa, että `send_packet` palauttamat arvot voi riippua paitsi sen syötearvoista, myös monista muista asioista, mukaan lukien tämän funktion aikaisempien kutsujen syöte- ja palautusarvoista ja testijärjestelmän luomista näennäissatunnaisista luvuista. Testijärjestelmä on **deterministinen** siinä mielessä, että jos suoritat sen kahdesti ja molemmissa suorituksissa lähetät samat paketit, se tekee niihin samat muutokset.

Esimerkki

Tarkastellaan seuraavaa kutsua:

```
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, 1])
```

Viesti, jonka Aisha yrittää lähettää Basmalle, on $[0, 1, 1, 0]$. Cleopatra ei voi muuttaa bittejä, joiden indeksit ovat 0 - 15, kun taas Cleopatra voi muuttaa bittejä, joiden indeksit ovat 16 - 30.

Tämän esimerkin vuoksi oletetaan, että Cleopatra täyttää peräkkäiset bitit, joita hän hallitsee vuorotellen 0 ja 1, eli hän asettaa 0 ensimmäiseen hänen hallitsemaansa indeksiin (indeksi 16 meidän tapauksessamme), 1 toiseen hallitsemaansa indeksiin (indeksi 17), 0 hänen hallitsemaansa kolmanteen indeksii (indeksi 18), ja niin edelleen.

Aisha voi päättää lähettää kaksi bittiä alkuperäisestä viestistä yhdessä paketissa seuraavasti: hän lähettää ensimmäisen bitin ensimmäisistä 8 indekseistä, joita hän hallitsee ja toisen bitin seuraavista 8 indekseistä, joita hän hallitsee.

Aisha päättää sitten lähettää seuraavan paketin:

```
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])
```

Huomaa, että Cleopatra voi vaihtaa bittejä viimeisissä 15 indekseissä, joten Aisha voi asettaa ne mielivaltaisesti, koska ne voidaan pilata. Cleopatran oletetun strategian avulla funktio palauttaa: $[0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1]$.

Aisha päättää lähettää M:n kaksi viimeistä bittiä toisessa paketissa samalla tavalla kuin ennenkin:

```
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])
```

Cleopatran oletetun strategian avulla funktio palauttaa: $[1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0]$.

Aisha voi lähettää lisää paketteja, mutta hän päättää olla lähettämättä.

Tämän jälkeen testijärjestelmä tekee seuraavan funktiokutsun:

```
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]])
```

Basma palauttaa viestin M seuraavasti. Jokaisesta paketista hän ottaa ensimmäisen bitin, joka esiintyy kahdesti peräkkäin, ja viimeisen bitin, joka esiintyy kahdesti peräkkäin. Eli ensimmäisestä paketista hän ottaa bitit $[0,1]$ ja toisesta paketin hän ottaa bitit $[1,0]$. Laittamalla ne yhteen, hän saa selvitettyä viestin $[0,1,1,0]$, joka on oikea palautusarvo tälle kutsulle `receive_message`.

Voidaan osoittaa, että Cleopatran oletetulla strategialla ja viesteillä, joiden pituus on 4, tämä Basman lähestymistapa palauttaa viestin M oikein C:n arvoista riippumatta. Yleisessä tapauksessa se ei kuitenkaan pidä paikkaansa.

Esimerkki testijärjestelmästä

Esimerkkinä oleva testijärjestelmä ei ole mukautuva. Sen sijaan Cleopatra täyttää peräkkäiset bitit, joita hän hallitsee, vuorotellen 0 ja 1 bitteillä, kuten yllä olevassa esimerkissä on kuvattu.

Syötteen muoto: **Syötteen ensimmäinen rivi sisältää kokonaisluvun T, joka määrittää testitapausten lukumäärän.** T testitapausta on kuvattu seuraavilla riveillä. Jokainen niistä kuvataan seuraavassa muodossa:

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

Tulosteen muoto: Esimerkkinä oleva testijärjestelmä kirjoittaa jokaisen T testitapauksen tuloksen samassa järjestyksessä kuin ne on annettu syötteessä seuraavassa muodossa:

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

Tässä K on `send_packet` -kutsujen määrä, D on `receive_message` :n palauttama viesti ja L on sen pituus.