Zpráva

Aisha si dopisuje se svojí kamarádkou Basmou. Aisha má zprávu M, což je posloupnost S bitů (tj. jedniček nebo nul), kterou by Basmě ráda poslala. Aisha komunikuje s Basmou tak, že jí posílá **pakety**. Paket je posloupnost 31 bitů očíslovaných od 0 do 30. Aisha by ráda Basmě svoji zprávu M poslala tak, že ji pošle nějaký počet paketů.

Bohužel byla komunikace mezi Aishou a Basmou kompromitována Cleopatrou, která může pakety **poškodit**. To znamená, že v každém paketu může Cleopatra upravit bity na přesně 15 pozicích. Konkrétně existuje pole C délky 31, ve kterém jsou všechny prvky 0 nebo 1, s následujícím významem:

- $C[i] = 1$ znamená, že bit na pozici i může Cleopatra změnit. Tyto pozice nazveme **ovládané** Cleopatrou.
- $C[i] = 0$ znamená, že Cleopatra bit na pozici i změnit nemůže.

Pole C obsahuje přesně 15 jedniček a 16 nul. Během přenášení zprávy M bude množina pozic ovládaných Cleopatrou stejná pro všechny pakety. Aisha ví přesně kterých 15 pozic Cleopatra ovládá. Basma ví jen to, že Cleopatra ovládá 15 pozic, ale neví které pozice.

Nechť A je paket, který se Aisha rozhodne odeslat (nazveme jej **původní paket**). Nechť B je paket, který Basma přijme (nazveme jej **poškozený paket**). Pro každé i takové, že $0 \leq i < 31$:

- pokud Cleopatra neovládá bit na pozici i ($C[i] = 0$), Basma přijme bit i tak, jak ho Aisha odeslala ($B[i] = A[i]$),
- jinak, pokud Cleopatra ovládá bit na pozici i ($C[i] = 1$), tak hodnotu $B[i]$ určí Cleopatra.

Ihned poté, co Aisha odešle paket, tak se dozví obsah příslušného poškozeného paketu.

Poté, co Aisha odešle všechny pakety, tak Basma přijme všechny poškozené pakety **v tom pořadí, ve kterém byly odeslány** a musí zrekonstruovat původní zprávu M.

Vaším úkolem je vymyslet a implementovat strategii, která umožní Aishe poslat Basmě zprávu M tak, aby Basma mohla M zrekonstruovat z poškozených paketů. Konkrétně byste měli implementovat dvě funkce. První funkce provede akce Aishy. Dostane zprávu M a pole C a měla by odeslat pakety tak, aby Basmě zprávu přenesla. Druhá funkce provede akce Basmy. Dostane poškozené pakety a měla by zrekonstruovat původní zprávu M.

Implementační detaily

První funkce, kterou máte implementovat, je:

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

- M: pole délky S udávající zprávu, kterou chce Aisha poslat Basmě.
- C: pole délky 31 udávající pozice bitů ovládaných Cleopatrou.
- Tato funkce může být zavolána **nejvýše 2100krát** v každém z testů.

Tato funkce by měla poslat paket zavoláním následující funkce:

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

- A: původní paket (pole délky 31) obsahující bity odeslané Aishou.
- Tato funkce vrátí poškozený paket B obsahující bity, které přijme Basma.
- Tato funkce může být zavolána nejvýše 100krát během každého zavolání `send_message`.

Druhá funkce, kterou máte implementovat, je:

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

- R: pole popisující poškozené pakety. Tyto pakety vychází z paketů odeslaných Aishou během jednoho volání `send_message` a jsou uvedeny **v pořadí, ve kterém byly odeslány** Aishou. Každý prvek R je pole délky 31 reprezentující poškozený paket.
- Tato funkce by měla vrátit pole S bitů, které se shoduje s původní zprávou M.
- Tato funkce může být zavolána **vícekrát** v každém z testů, **právě jednou** za každé odpovídající volání `send_message`. **Pořadí volání** `receive_message` nemusí být nutně stejné jako pořadí odpovídajících volání `send_message`.

Upozorňujeme, že ve vyhodnocovacím prostředí jsou funkce `send_message` a `receive_message` volané ve **dvou různých programech**.

Omezení

- $1 \leq S \leq 1024$
- C má právě 31 prvků, z nichž 16 je rovných 0 a 15 je rovných 1.

Podúlohy a bodování

Pokud v libovolném z testů volání funkce `send_packet` nebudou odpovídat pravidlům uvedeným výše, nebo pokud návratová hodnota libovolného z volání funkce `receive_message` nebude správná, poté bude skóre vašeho řešení na tomto testu 0.

Jinak, nechť Q je maximální počet volání procedury `send_packet` napříč všemi volání `send_message` napříč všemi testy. Nechť je dále X rovné:

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

Poté se skóre počítá následujícím způsobem:

Podúloha	Počet bodů	Další omezení
1	$10 \cdot X$	$S \leq 64$
2	$90 \cdot X$	Žádná další omezení.

Upozorňujeme, že je v některých případech grader může být **adaptivní**. To znamená, že návratové hodnoty funkce `send_packet` mohou záviset nejen na jejích argumentech, ale i na mnoho dalších věcech, včetně vstupů a návratových hodnot předchozích volání této funkce a pseudo-náhodných čísel vygenerovaných graderem. Grader je **deterministický** v tom významu, že pokud jej pustíte dvakrát a v obou voláních odešlete stejné pakety, tak v nich grader udělá stejné změny.

Příklad

Uvažujme následující volání.

```
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])
```

Zpráva, kterou se Aisha snaží poslat Basmě, je $[0, 1, 1, 0]$. Bity na pozicích 0 až 15 Cleopatra nemůže změnit, zatímco bity na pozicích 16 až 30 Cleopatra změnit může.

Pro potřeby této ukázky předpokládejme, že Cleopatra ovládané pozice vyplňuje střídavě bity 0 a 1. Tedy nastaví první ovládanou pozici (v našem případě pozici 16) na 0, druhou ovládanou pozici (pozici 17) na 1, třetí ovládanou pozici (pozici 18) na 0, a tak dále.

Aisha se může rozhodnout poslat dva bity původní zprávy v jednom paketu následujícím způsobem: pošle první bit v prvních 8 pozicích, které Aisha ovládá, a druhý bit v následujících 8 pozicích, které ovládá.

Aisha se poté rozhodne odeslat následující paket:

```
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])
```

Všimněte si, že Cleopatra může změnit bity na posledních 15 pozicích, takže je Aisha může nastavit libovolně, protože mohou být přepsány. S naší předpokládanou strategií Cleopatry funkce vrátí: $[0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1]$.

Aisha se rozhodne odeslat poslední dva bity M v druhém paketu podobným způsobem, jako minule:

```
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])
```

S naší předpokládanou strategií Cleopatry funkce tentokrát vrátí pole: $[1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0]$.

Aisha může poslat více paketů, ale rozhodne se tak neučinit.

Grader poté provede následující volání funkce:

```
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 zprávu M zrekonstruuje následujícím způsobem. Z každého paketu vezme první bit, který se opakuje dvakrát po sobě, a poslední bit, který se opakuje dvakrát po sobě. Z prvního paketu tedy vezme bity $[0,1]$ a z druhého paketu vezme bity $[1,0]$. Tím, že je dá dohromady, zrekonstruuje zprávu $[0,1,1,0]$, což je pro toto volání `receive_message` správná návratová hodnota.

Dá se dokázat, že s touto předpokládanou strategií Cleopatry a pro zprávy délky 4 Basma s touto strategií vždy správně zrekonstruuje M, bez ohledu na hodnotu C. Toto nicméně v obecném případě neplatí.

Ukázkový grader

Ukázkový grader není adaptivní. Místo toho Cleopatra vyplňuje ovládané pozice střídavě bity 0 a 1, jak je popsáno výše v příkladu.

Formát vstupu: **První řádek vstupu obsahuje celé číslo T, udávající počet scénářů.** Následuje T scénářů. Každý z nich má následující formát:

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

Formát výstupu: Ukázkový grader vypíše výsledek každého z T scénářů ve stejném pořadí, jako jsou uvedeny na vstupu, v následujícím formátu:

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

Zde je K počet volání `send_packet`, D je zpráva vrácená funkcí `receive_message` a L délka této zprávy.