Mensaje

Aisha y Basma son dos amigas que se suelen comunicar via mensajes. Aisha tiene un mensaje M que consiste en una secuencia de S bits (o sea, de unos y ceros) que le quiere mandar a Basma. La forma en la que Aisha se comunica con Basma es enviándole **paquetes**. Un paquete es una secuencia de 31 bits, indexados del 0 al 30. Aisha quiere comunicar un mensaje M a Basma enviándole un cierto número de paquetes.

Lamentablemente, Cleopatra interceptó la comunicación entre Aisha y Basma y puede **alterar** los paquetes. Es decir, en cada paquete Cleopatra puede modificar los bits en exactamente 15 índices. Concretamente, hay un arreglo C de ceros y unos de longitud 31, que significa lo siguiente:

- Si $C[i] = 1$, entonces Cleopatra puede modificar el bit con índice i del paquete. A estos índices los denominamos como **controlados** por Cleopatra.
- Si $C[i] = 0$, entonces Cleopatra no puede modificar el bit con índice i del paquete.

El arreglo C tiene exactamente 15 unos y 16 ceros. Al mandar un mensaje M, el conjunto de índices controlados por Cleopatra es el mismo para todos los paquetes. Aisha sabe exactamente cuáles 15 índices forman el conjunto controlado por Cleopatra, pero Basma sólo sabe que son 15 y no cuáles son.

Sea A el paquete que Aisha decide mandar (que denominaremos el **paquete original**). Sea B el paquete que recibe Basma (que denominaremos el **paquete alterado**). Para cada i con $0 \leq i < 31$:

- si Cleopatra no controla el bit con índice i (o sea, $C[i] = 0$), Basma recibe el bit i tal como lo mandó Aisha ($B[i] = A[i]$).
- en caso contrario, si Cleopatra controla el bit con índice i (o sea, $C[i] = 1$), entonces Cleopatra puede elegir el valor del bit $B[i]$.

En cuanto Aisha manda un paquete, se entera de inmediato de cuál es el paquete alterado correspondiente.

Luego de que Aisha mande todos los paquetes, Basma recibe todos los paquetes alterados **en el orden en el que fueron enviados**, y debe reconstruir el mensaje original M.

Tu tarea es idear e implementar una estrategia que permita a Aisha mandar el mensaje M a Basma, de forma que Basma pueda recontruir el mensaje M a partir de los paquetes alterados. Específicamente, tienes que implementar las siguientes dos funciones:

La primera función realiza las acciones de Aisha: Toma como entrada un mensaje M y un arreglo C, y debe mandar algunos paquetes para comunicar el mensaje a Basma.

La segunda función realiza las acciones de Basma: Toma como entrada los paquetes alterados y tiene que reconstruir el mensaje original M.

Detalles de Implementación

La primera función a implementar es:

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

- M: un arreglo de largo S que contiene el mensaje que Aisha quiere mandar a Basma.
- C: un arreglo de largo 31 que indica qué índices de bits controla Cleopatra.
- Esta función puede ser llamada **a lo más 2100 veces** en cada caso de prueba.

Para mandar un paquete, la función anterior debe llamar a la siguiente:

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

- A: un paquete original (un arreglo de largo 31) que representa los bits enviados por Aisha.
- Esta función retorna el paquete alterado B, que son los bits que recibe Basma.
- Esta función puede ser llamada **a lo más 100 veces** en cada invocación de `send_message`.

La segunda función que tienes que implementar es:

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

- R: un arreglo que describe los paquetes alterados. Estos paquetes provienen de los paquetes mandados por Aisha con una llamada a `send_message`, y aparecen **en el orden en el que fueron enviados** por Aisha.
- Esta función debe retornar un arreglo de S bits que coincida con el mensaje original M.
- Esta función puede ser invocada **múltiples veces** por cada caso de prueba, **exactamente una vez** en cada respectiva llamada a `send_message`. El **orden de las llamadas a la función** `receive_message` no es necesariamente igual al orden de las respectivas llamadas a `send_message`.

Ten en cuenta que en el juez `send_message` y `receive_message` se ejecutan en **dos programas separados**.

Restricciones

- $1 \leq S \leq 1024$

- C tiene exactamente 31 elementos, de los cuales 16 son 0 y 15 son 1.

Subtareas y Puntuación

Si en cualquiera de los casos de prueba, las llamadas a `send_packet` no cumplen con las reglas mencionadas anteriormente, o el valor que retorna cualquiera de las llamadas a `receive_message` es incorrecto, el puntaje de tu solución para ese caso de prueba será 0.

En el caso contrario, sea Q el máximo número de llamadas a la función `send_packet` entre todas las invocaciones a `send_message` sobre todos los casos de prueba, y sea X definido como

- $X = 1$, si $Q \leq 66$
- $X = 0.95^{Q-66}$, si $66 < Q \leq 100$

Entonces, el puntaje se calcula de la siguiente manera:

Subtarea	Puntaje	Restricciones Adicionales
1	$10 \cdot X$	$S \leq 64$
2	$90 \cdot X$	Sin restricciones adicionales.

Nota que en algunos casos el comportamiento del evaluador puede ser **adaptativo**. Esto significa que los valores que retorna `send_packet` pueden depender no sólo de los valores de entrada sino también de muchas otras cosas, incluyendo las entradas y valores de retorno de llamadas anteriores a esta función, y números pseudo-aleatorios generados por el evaluador. El evaluador es **determinístico**: si se corre dos veces y en ambas envias los mismos paquetes, el evaluador los alterará de la misma manera.

Ejemplo

Considera la siguiente llamada:

```
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])
```

El mensaje que Aisha le intenta mandar a Basma es $[0, 1, 1, 0]$. Los bits con índices 0 a 15 no pueden ser modificados por Cleopatra, y los bits con índices 16 a 30 sí pueden ser modificados por Cleopatra.

Para el ejemplo, supongamos que Cleopatra llena bits consecutivos bajo su control de manera alternada con ceros y unos. Es decir, le asigna 0 al primer índice que controla (índice 16 en este

caso), 1 al segundo índice que controla (índice 17 en nuestro caso), 0 al tercer índice que controla (índice 18 en este caso), y así sucesivamente.

Aisha puede decidir mandar dos bits del mensaje original en un paquete de la siguiente manera: manda el primer bit de M en los primeros 8 índices que ella controla y el segundo bit de M en los siguientes 8 índices bajo su control.

Entonces, Aisha decide enviar el siguiente paquete:

```
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])
```

Ten en cuenta que Cleopatra puede modificar los bits de los últimos 15 índices, entonces Aisha puede asignarlos de forma arbitraria, dado que Cleopatra podría sobreescribirlos. Con la estrategia de Cleopatra que asumimos en el ejemplo, la función retornará:

$[0,0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0]$.

Aisha decide mandar los últimos dos bits de M en el segundo paquete de una manera similar:

```
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])
```

Con la estrategia que asumimos de Cleopatra, la función retornará $[1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0,1,0]$.

Aisha podría mandar más paquetes, pero decide no hacerlo. Luego, el evaluador realiza la siguiente llamada:

```
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 reconstruye el mensaje M de la siguiente manera: de cada paquete, toma el primer bit que aparece dos veces consecutivas, y el último bit que aparece dos veces consecutivas. Es decir, del primer paquete toma los bits $[0,1]$ y del segundo paquete toma los bits $[1,0]$. Concatenando estos bits, recupera el mensaje $[0,1,1,0]$, que es el valor correcto que debe retornar `receive_message` para esta llamada.

Se puede demostrar que con la estrategia que asumimos para Cleopatra y para mensajes de largo 4, este método de Basma recupera correctamente el mensaje M, sin importar el valor de C. Sin embargo, esta estrategia no es correcta en general.

Evaluador de Ejemplo

El evaluador de ejemplo no es adaptativo. En cambio, Cleopatra llena bits consecutivos bajo su control con bits 0 y 1 alternadamente, como se describe en el ejemplo anterior.

Formato de entrada: **La primera línea contiene un entero T, que denota la cantidad de escenarios**. Luego aparecen T escenarios, cada uno de ellos en el siguiente formato:

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

Formato de salida: El evaluador local muestra el resultado de cada uno de los T escenarios en el mismo orden en el que aparecen en la entrada, usando el siguiente formato:

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

Donde K es el número de llamadas a `send_packet`, D es el mensaje retornado por `receive_message` y L es su largo.