

Дано бінарну послідовність a_0, \dots, a_{n-1} . Ви не знаєте її елементів, але можете запитати суму будь-якої пари різних елементів. Ваше завдання — відновити всю послідовність, використавши якнайменше таких запитів.

Формат взаємодії

Це інтерактивна задача. Ви повинні написати програму, яка знайде правильну відповідь, взаємодіючи з інтерактором через стандартні вхідні та вихідні потоки даних.

На початку ваша програма повинна зчитати ціле число n ($3 \leq n \leq 200\,000$) — довжину послідовності.

Далі програма може ставити запити у такому форматі:

- Вивести рядок у вигляді:

$i \ j$

на стандартний потік виходу, де $i \ i \ j$ — різні цілі числа ($0 \leq i \neq j \leq n-1$). Цей рядок означає запит про суму $a_i + a_j$.

- У відповідь ви отримаєте одне число:

s

де s — це сума $a_i + a_j$ ($0 \leq s \leq 2$).

Коли ви визначили всі елементи послідовності, виведіть два рядки:

- Перший рядок: число n ;
- Другий рядок: n чисел a_0, a_1, \dots, a_{n-1} (елементи послідовності).

Після цього ваша програма повинна завершити роботу. Подальші запити можуть призвести до Wrong answer.

Не забудьте очистити буфер виводу після кожного запиту чи відповіді:

- У C++ використовуйте `std::endl` або `fflush(stdout)`;
- У Python використовуйте `print(..., flush=True)`.

Інакше ваша програма може отримати Time Limit Exceeded.

Ваша програма не може відкривати ніякі файли чи використовувати будь-які інші ресурси. Можна використовувати стандартний потік помилок (standart error stream) для дебагу, але зважайте, що запис на цей потік вимагає часу.

C++, вхідний/вихідний потік використовуючи streams

Не забудьте включити потрібну бібліотеку (`#include<iostream>`). До кожного виведеного рядка варто додати `std::endl`. Перегляньте приклад початкової взаємодії наведений нижче.

```
std::cin >> n;
std::cout << i << ' ' << j << std::endl;
std::cin >> sum;
```

C++, вхідний/вихідний потік використовуючи stdio

Не забудьте включити потрібну бібліотеку (`#include<stdio.h>`). Після кожного виведеного рядка варто запустити `fflush(stdout)`. Перегляньте приклад початкової взаємодії наведений нижче.

```
scanf("%d", &n);
printf("%d %d\n", i, j);
fflush(stdout);
scanf("%d", &sum);
```

Python

Кожен виклик `print` має мати `flush = True` як аргумент. Перегляньте приклад початкової взаємодії наведений нижче.

```
n = int(input())
print(f"{i} {j}", flush = True)
sum = int(input())
```

Приклад взаємодії

Дія	Параметри	Результат	Пояснення
зчитування n	–	5	$n = 5$
запит суми	0 1	1	$a_0 + a_1 = 1$
запит суми	1 2	1	$a_1 + a_2 = 1$
запит суми	3 4	2	$a_3 + a_4 = 2$, тому $a_3 = a_4 = 1$
запит суми	0 3	2	$a_0 + a_3 = 2$, тому $a_0 = 1$, і далі $a_1 = 0$, $a_2 = 1$
відповідь	5 1 0 1 1 1		Правильна відповідь. $m = 4 \leq n = 5$ — 100% балів.

Оцінювання

Підзадача	Обмеження	Бали
1	$3 \leq n \leq 1000$	50
2	$3 \leq n \leq 200\,000$	50

Інтерактор **не зобов'язаний** одразу фіксувати всю послідовність a_0, \dots, a_{n-1} . Він може адаптувати її під час взаємодії, за умови, що всі попередні відповіді на запити залишаються правильними. Іншими словами, інтерактор є адаптивним.

Позначимо m — кількість запитів, зроблених вашою програмою. Ви отримаєте такий відсоток балів:

Кількість запитів	Відсоток балів
$m \leq n$	100%
$m = n + 1$	80%
$n + 1 < m \leq n^2 - n$	50%
$m > n^2 - n$	0% (результат Wrong answer)