Задача 1. Футбольная команда

Имя входного файла: team.in
Имя выходного файла: team.out
Ограничение по времени: 2 секунда
Ограничение по памяти: 256 Мбайт

Говорят, что на исход матча влияет место, где играют футболисты. Но по последним данным британских учёных, не только место. Так же на результат влияют номера футболистов. Причем, чем ближе сумма номеров одной команды к сумме номеров другой, тем выше вероятность ничейного результата. Недоверчивые болельщики решили проверить теорию британских учёных для команды с произвольным количеством участников.

Вам необходимо разбить n футболистов на 2 команды так, чтобы разность между суммами номеров этих команд была наименьшей.

Формат входного файла

Первая строка содержит n ($2 \le n \le 20$) — количество футболистов. В следующей строке заданы n целых чисел — номера футболистов в диапазоне от 1 до 10^8.

Формат выходного файла

Выходной файл должен содержать минимальную разность между суммами номеров противоположных команд.

Пример входных и выходных данных

team.in	team.out
4 1 3 2 2	0
5 7 1 3 4 2	1

Задача 2. Мозаика

Имя входного файла: mosaic.in
Имя выходного файла: mosaic.out
Ограничение по времени: 1 секунда
Ограничение по памяти: 256 Мбайт

Все элементы магнитной мозаики фирмы «ABBYY» имеют прямоугольную форму. Два элемента можно соединить только в том случае, если у них совпадает хотя бы один из размеров: длина, ширина, или и то, и другое. Магнитные элементы поворачивать и переворачивать нельзя. Пару элементов мозаики, которые нельзя соединить, назовем *негармоничной*. Например, пара 1×2 и 2×3 является негармоничной, а пары 2×3 и 1×3 или 2×3 и 2×3 являются гармоничными.

Дизайнеры «ABBYY» выложили все элементы мозаики в ряд, не соединяя их между собой. Назовем набором несколько подряд лежащих элементов мозаики в этом ряду. Они выбрали несколько наборов элементов, которые хотят оставить для создания инсталляции. Для каждого такого набора им нужно выяснить, есть ли в нем негармоничная пара элементов.

Требуется написать программу, которая для различных наборов подряд лежащих элементов мозаики определит номера элементов, образующих негармоничную пару, или сообщит, что такой пары нет.

Формат входного файла

В первой строке входного файла записано одно число n – количество элементов, из которых состоит мозаика ($2 \le n \le 100\,000$). В следующих n строках записаны по два целых числа A_i и B_i, задающих длину и ширину i-го элемента мозаики соответственно ($1 \le A_i, B_i \le 10^9$, $1 \le i \le n$).

В $(n + 2)$-й строке записано одно целое число k – количество наборов, в каждом из которых нужно определить номера двух негармоничных элементов ($1 \le k \le 100\,000$). В следующих k строках записаны пары целых чисел n_1 и n_2 – номера первого и последнего элементов набора соответственно, в котором необходимо найти два негармоничных элемента мозаики ($1 \le n_1 < n_2 \le n$).

Формат выходного файла

Выходной файл должен содержать k строк, каждая из которых содержит два разделённых пробелом числа — номера элементов мозаики, образующих негармоничную пару в соответствующем наборе. Если решений несколько, можно вывести любое из них. Если в наборе негармоничная пара отсутствует, требуется вывести в соответствующей строке 0 0.

Пример входных и выходных данных

mosaic.in	mosaic.out
4 2 2 1 2 1 3 2 3 2 2 3 2 4	0 0 4 2

Подзадачи и системы оценки

Данная задача содержит четыре подзадачи. Для оценки каждой подзадачи используется своя группа тестов. Баллы за подзадачу начисляются только в том случае, если все тесты из этой группы успешно пройдены.

Подзадача 1 (*оценивается в* 20 *баллов*)

Количество элементов мозаики $n \le 100$, число наборов $k \le 100$.

Подзадача 2 (*оценивается в* 30 *баллов*)

Количество элементов мозаики $n \le 1\,000$, число наборов $k \le 1\,000$.

Подзадача 3 (*оценивается в* 20 *баллов*)

Количество элементов мозаики $n \le 5\,000$, число наборов $k \le 5\,000$.

Подзадача 4 (*оценивается в* 30 *баллов*)

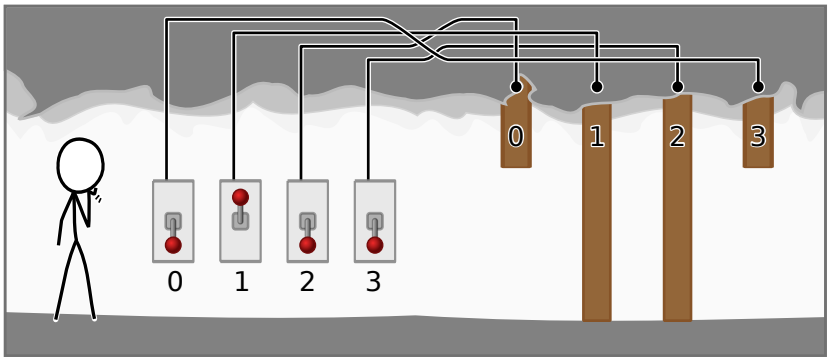
Количество элементов мозаики $n \le 100\,000$, число наборов $k \le 100\,000$.

International Olympiad in Informatics 2013
6-13 July 2013
Brisbane, Australia
Day 2 tasks

cave

Russian — 1.0

При поиске места проведения соревнования — UQ Центра, вы заблудились и оказались в секретной группе пещер под университетом. Вход в группу пещер преграждает система безопасности, включающая в себя `N` дверей, расположенных друг за другом, и `N` переключателей, каждый из которых соединён только с одной из дверей. Различные переключатели соединены с различными дверями.



Двери пронумерованы от входа в группу пещер от `0` до `(N - 1)` в порядке от ближних к дальним. Переключатели также пронумерованы от `0` до `(N - 1)`, однако, вы не знаете, с какой дверью соединен каждый переключатель.

Все переключатели расположены у входа в группу пещер. Каждый переключатель может быть в одном из двух положений: "вверх" или "вниз". Одно из этих положений является правильным. Если он находится в правильном положении, то соединенная с переключателем дверь будет открыта, иначе эта дверь будет закрыта. Правильное положение может различаться для разных переключателей, и вы не знаете, какое из положений для какого переключателя является правильным.

Вы хотите понять, как устроена система безопасности. Для этого вы можете задать комбинацию положений переключателей, то есть установить каждый из переключателей в любое из положений; после чего войти в пещеру и узнать номер первой закрытой двери. Двери, находящиеся за ней, не видны.

У вас есть время на то, чтобы попробовать не более `70 000` комбинаций положений переключателей. Ваша задача — определить правильное положение для каждого переключателя, а также для каждого переключателя найти дверь, с которой он соединен.

Детали реализации

Ваше решение должно реализовывать функцию `exploreCave()`. Она может вызывать функцию проверяющего модуля `tryCombination()` не более 70 000 раз и должна завершаться вызовом функции `answer()`. Эти функции описаны ниже.

Функция проверяющего модуля: `tryCombination()`

C/C++
```
int tryCombination(int S[]);
```

Pascal
```
function tryCombination(var S: array of LongInt) : LongInt;
```

Описание

Эту функцию реализует проверяющий модуль. Она позволяет вам задать комбинацию положений переключателей для того, чтобы войти в группу пещер и узнать номер первой закрытой двери. Если все двери открыты, то функция возвращает значение `-1`. Эта функция работает за время `O(N)`, то есть время работы в худшем случае пропорционально `N`.

Эта функция может быть вызвана не более `70 000` раз.

Параметры

- `S`: массив длины `N`, задающий комбинацию положений переключателей. Элемент массива `S[i]` соответствует переключателю с номером `i`. Значение `0` означает, что переключатель находится в положении "вверх". Значение `1` означает, что переключатель находится в положении "вниз".
- *Возвращаемое значение*: номер первой закрытой двери или `-1`, если все двери открыты.

Функция проверяющего модуля: `answer()`

C/C++
```
void answer(int S[], int D[]);
```

Pascal
```
procedure answer(var S, D: array of LongInt);
```

Описание

Вызовите эту функцию, когда определена комбинация положений переключателей, при которой все двери открыты, а также для каждого переключателя определена дверь, с которой он соединён.

Параметры

- `S`: массив длины `N`, содержащий правильное положение каждого переключателя. Формат данного массива такой же, как и у параметра вышеописанной функции `tryCombination()`.
- `D`: массив длины `N`, элементы которого для каждого переключателя задают, с какой дверью он соединён. Элемент `D[i]` должен содержать номер двери, с которой соединён переключатель с номером `i`.
- *Возвращаемое значение*: эта функция не возвращает управление вашей программе, то есть приводит к завершению вашей программы.

Ваша функция: `exploreCave()`

C/C++
```
void exploreCave(int N);
```

Pascal
```
procedure exploreCave(N: longint);
```

Описание

Ваше решение должно реализовывать эту функцию.

Эта функция должна использовать функцию проверяющего модуля `tryCombination()`, чтобы определить правильное положение для каждого переключателя, а также для каждого переключателя определить дверь, с которой он соединен. Когда ваша функция получит эту информацию, она должна вызвать функцию `answer()`.

Параметры

- `N`: количество переключателей и дверей в группе пещер.

Пример

Предположим, что двери и переключатели расположены, как показано на рисунке выше. Возможный порядок вызова функций представлен ниже.

Вызов функции	Возвращаемое значение	Объяснение
`tryCombination([1, 0, 1, 1])`	`1`	Ситуация соответствует показанной на рисунке. Переключатели 0, 2 и 3 находятся в положении "вниз", а переключатель 1 — в положении "вверх". Функция возвращает значение `1`, что значит, что дверь 1 — первая дверь слева, которая закрыта.
`tryCombination([0, 1, 1, 0])`	`3`	Двери 0, 1 и 2 открыты, а дверь 3 — закрыта.
`tryCombination([1, 1, 1, 0])`	`-1`	Перевод переключателя 0 в положение "вниз" приводит к тому, что все двери открылись, что обозначается возвращаемым значением `-1`.
`answer([1, 1, 1, 0], [3, 1, 0, 2])`	*(Программа завершается)*	Ваша программа определила, что комбинация `[1, 1, 1, 0]` задаёт правильное положение каждого из переключателей, и переключатели 0, 1, 2 и 3 соединены с дверями 3, 1, 0 и 2, соответственно.

Ограничения

- Ограничение по времени: 2 секунды.
- Ограничение по памяти: 32 МиБ.
- $1 \le N \le 5000$

Подзадачи

Подзадача	Баллы	Дополнительные ограничения на входные данные
1	12	Для каждого i, переключатель с номером i соединён с дверью с номером i. Ваша задача — определить правильное положение для каждого переключателя.
2	13	Правильное положение всех переключателей всегда `[0, 0, 0, …, 0]`. Ваша задача — определить, какой переключатель соединён с какой дверью.
3	21	$N \le 100$
4	30	$N \le 2000$
5	24	*(нет)*

Взаимодействие с проверяющим модулем

Проверяющий модуль на вашем компьютере будет считывать входные данные из файла `cave.in`, который должен иметь следующий формат:

- строка 1: `N`
- строка 2: `S[0] S[1] … S[N - 1]`
- строка 3: `D[0] D[1] … D[N - 1]`

Здесь `N` — количество дверей и переключателей, `S[i]` — правильное положение переключателя с номером `i`, а `D[i]` — дверь, с которой соединён переключатель с номером `i`.

В частности, вышеописанный пример должен быть задан в таком формате:

```
4
1 1 1 0
3 1 0 2
```

Особенности конкретных языков программирования

C/C++	Вы должны подключить заголовочный файл с помощью `#include "cave.h"`.
Pascal	Вы должны написать модуль с заголовком `unit Cave`, а также импортировать процедуры проверяющего модуля с помощью `uses GraderHelpLib`. Все массивы нумеруются, начиная с `0` (а не с `1`).

Для примера посмотрите шаблоны решений на вашем компьютере.