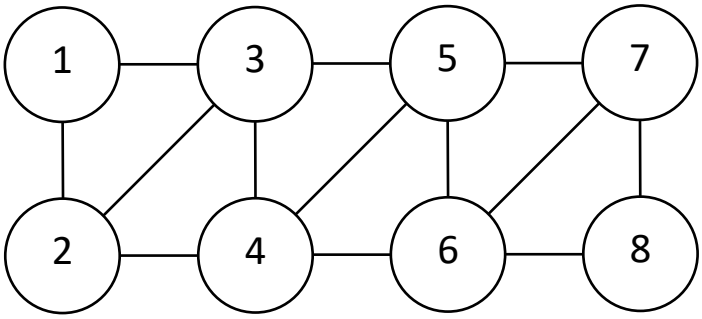
Васи наконец решила посетить Рузе и погулять по парку. Она прочитала путеводитель и выяснила, что парк имеет прямоугольную форму, и его длинные стороны имеют длину равную N. Вдоль каждой из этих сторон находятся по $N + 1$ интересному месту (кафе, теннисные корты, фонтаны, и т. п.), таким образом всего в парке $2N + 2$ интересных места. Они пронумерованы и соединены аллеями как показано на рисунке ниже (приведен рисунок для $N = 3$). У парка один вход (в интересном месте номер 1) и один выход (в интересном месте номер $2N + 2$).



Все аллеи в парке односторонние, и когда посетитель прогуливается по аллее, он должен соблюдать направление движения. Аллеи ориентированы таким образом, что в парке **нет циклов.** Также известно, что вход – единственное интересное место, **из которого все соответствующие аллеи выходят**, а выход – единственное интересное место, **в которое все соответствующие аллеи входят.** Для всех других интересных мест выполнено, что **для них есть как входящие, так и исходящие аллеи**.

Будем говорить, что определенная аллея имеет положительное направление, если она ведет из интересного места с меньшим номером в интересное место с большим номером. В противном случае будем говорить, что она имеет отрицательное направление. Васи знает, что две аллеи, ведущие из входа, а также две аллеи, ведущие в выход, имеют положительное направление (поскольку вход имеет номер 1, а выход – номер $2N + 2$), но она не знает ничего про направление остальных аллей, и она хочет его выяснить, чтобы составить план прогулки по парку.

Для любителей задач по математике и информатике около входа в парк размещается компьютер, на котором исполняется специальная программа. Она позволяет выполнять следующие странные запросы: по заданному списку аллей она возвращает результат выполнения операции XOR над их направлениями (положительное направление аллеи задается цифрой 1, а отрицательное – цифрой 0). *Напомним, что XOR двух двоичных цифр равен 1, если они различны, либо 0, если они одинаковые. В случае более чем двух операндов мы сначала выполняем операцию XOR для первых двух операндов, затем для результата и третьего операнда, и так далее.* **Программа работает таким образом, что, если ей на вход подать только одну аллею, она просто возвращает её направление.**

Васи хочет выяснить направление каждой аллеи, сделав не слишком много запросов к программе.

Жюри скопировало программу с компьютера у входа в парк и запустило на проверяющем сервере. Помогите Васи, напишите функцию **run**, которая будет скомпилирована вместе с программой жюри и будет взаимодействовать с ней, задавая вопросы описанного выше типа, и определит в итоге направление каждой аллеи.

Детали реализации

Ваша функция `run` должна иметь следующую сигнатуру:

```
void run(int n);
```

Она будет вызвана один раз и получит в качестве аргумента целое положительное число N – длину длинной стороны парка (общее число интересных мест равно $2N + 2$).

Для взаимодействия с программой жюри используются следующие две функции:

```
bool get_xor(const std::vector<std::pair<int, int>>& edges);
void state_direction(const std::pair<int, int>& edge, bool direction);
```

При каждом вызове `get_xor` возвращается результат выполнения операции XOR для направлений всех аллей, переданных в векторе `edges`. Каждая аллея задается парой чисел – номерами интересных мест, которые она соединяет (порядок не важен). Обратите внимание, функция `get_xor` выполняется за линейное время, и может потребовать линейной памяти для своего выполнения.

Ваша функция должна вызвать `state_direction` для каждой аллеи, передав номера интересных мест, которые она соединяет, в параметре `edge`, и выясненное направление аллеи в параметре `direction`. Если ваша программа в какой-то момент вызовет одну из двух функций с некорректным описанием аллеи или вызовет `state_direction` с неправильным направлением аллеи, тест будет считаться не пройденным. Тест также будет считаться не пройденным, если для какой-либо аллеи не будет вызвана функция `state_direction`.

Вам требуется отправить на проверку файл **park.cpp**. Он, помимо функции `run`, может содержать любые вспомогательные функции, классы, переменные, и так далее. Но он не должен содержать функцию `main`, а также должен подключать заголовочный файл `park.h`, используя инструкцию препроцессора `#include "park.h"` в начале файла.

Ограничения

$1 \leq N \leq 100\,000$

Система оценки

Ваше решение получит баллы, отличные от 0, если оно корректно решает задачу для каждого теста жюри, и количество вызовов функции `get_xor` для каждого теста не превышает $3N$. Количество баллов, которые вы получите, определяется следующей таблицей (здесь Q – количество вызовов `get_xor` для заданного теста; $\lceil x \rceil$ – минимальное целое число $\geq x$):

15 баллов : $Q \leq 3N$ для всех тестов, но $Q > 2N$ хотя бы для одного теста.

45 баллов: $Q \leq 2N$ для всех тестов, но $Q > \left\lceil \frac{3}{2}N \right\rceil$ хотя бы для одного теста.

70 баллов: $Q \leq \left\lceil \frac{3}{2}N \right\rceil$ для всех тестов, но $Q > \left\lceil \frac{16}{11}N \right\rceil$ хотя бы для одного теста.

100 баллов: $Q \leq \left\lceil \frac{16}{11}N \right\rceil$ для всех тестов.

Локальное тестирование

Вам доступен файл `Lgrader.cpp`, который вы можете скомпилировать вместе со своим решением, чтобы запускать тесты локально. После запуска программы вам необходимо ввести число N и затем ввести информацию об ориентации каждой аллеи (число 0 или 1). Он выводит запросы на экран, поясняя, что происходит. Вы можете модифицировать этот файл, чтобы сделать взаимодействие более удобным.

Адаптивное тестирование на сервере

На некоторых тестах (неизвестных вам) программа жюри при проверке вашего решения может адаптироваться к запросам, которые делает ваша программа. Таким образом, не гарантируется, что направление каждой аллеи зафиксировано заранее, оно может определяться по ходу, в зависимости от того, какие запросы задает ваша программа.

При запуске функции тестирования в проверяющей системе у вас не будет доступа к этому, адаптирующемуся, проверяющему модулю. При тестировании на сервере вы должны использовать следующий формат входных данных: сначала натуральное число *N*, затем направления аллей, $4N + 1$

нулей и единиц (направления аллей следует вводить в порядке 1-2, 1-3, 2-3, 2-4, …, k-(k+1), k-(k+2), …).

Пример взаимодействия программы

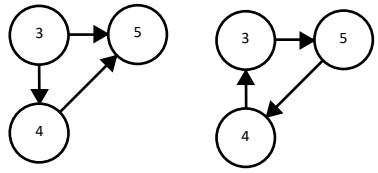
№	Действия функции `run`	Действия программы жюри
1.		`run(2)`
2.	`state_direction({1,2}, 1)`	
3.	`state_direction({1,3}, 1)`	
4.	`state_direction({4,6}, 1)`	
5.	`state_direction({5,6}, 1)`	
6.	`get_xor({{3,4},{3,5},{4,5}})`	`1`
7.	`get_xor({{3,5}})`	`1`
8.	`state_direction({3,4}, 1)`	
9.	`state_direction({3,5}, 1)`	
10.	`state_direction({4,5}, 1)`	
11.	`get_xor({{2,3},{2,4}})`	`0`
12.	`state_direction({2,3}, 1)`	
13.	`state_direction({2,4}, 1)`	
14.	Функция завершается	

Пояснения

1. Программа жюри вызывает функцию `run` с параметром $N = 2$.

2-5. `run` сообщает программе жюри направления аллей из входа и в выход.

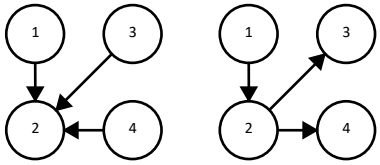
6-7. `run` задает два запроса о XOR-е и получает 1 в качестве результата. Рассмотрим два возможных



направления этих аллей:

8-10. Во втором случае получается цикл, но по условию циклов не бывает, поэтому `run` сообщает программе жюри направления аллей для первого варианта.

11. `run` задает вопрос о XOR-е направлений двух аллей и получает 0. Следовательно их направления одинаковые:



12-13. В первом случае все аллеи для интересного места номер 2 ведут в него, но известно, что в парке есть только одно такое интересное место (выход), значит этот вариант невозможен, таким образом `run` сообщает программе жюри информацию о направлениях для второго варианта.

14. `run` завершается. Использовано 3 запроса.