

Task C23. MONOPOLY 2

 1 sec.  256 MB

После того как вы помогли Дени сделать идеальную игровую карту для монополии в 2021 году, ей снова нужна ваша помощь! Напомним, что игровая карта для монополии состоит из N полей (пронумерованных от 1 до N) и M стрелок (направленных переходов) между ними. Здесь $M = N - 1$. Стрелки удовлетворяют следующим условиям — не бывает стрелок, направленных от поля к самому себе, и нет нескольких различных стрелок между одной и той же парой полей. Также стрелки обладают свойством, что каждое поле достижимо по ним из поля 1.

Определение: Пусть у нас есть перестановка чисел от 1 до N . Мы называем перестановку *корректным упорядочением полей*, если выполняется следующее условие: если есть стрелка от поля i к полю j , то i должно стоять **раньше** j в перестановке.

Проблема Дени в том, что она потеряла идеальную карту для игры в монополию. К счастью, её друг Боби помнит карту, но он решил немного повеселиться перед тем, как сообщить ей о том, как она устроена.

Сначала Боби говорит Дени, что перестановка $1, 2, \dots, N$ является корректным упорядочением полей. После этого Боби ответит на некоторые вопросы Дени, чтобы помочь ей выяснить, какие стрелки существуют между полями. Каждый вопрос будет касаться того, *насколько корректным упорядочением полей* является какая-то перестановка чисел от 1 до N (для более подробной информации см. раздел "Детали реализации"). Наша героиня просит вас помочь придумать и реализовать стратегию с как можно меньшим количеством вопросов.

Задача

Напишите программу **monopoly2**, которая определяет неизвестные стрелки, задавая как можно меньше вопросов Боби. Она должна содержать функцию `find_connections`, которая будет скомпилирована с программой жюри (эта программа берёт на себя роль Боби).

Детали реализации

Ваша функция `find_connections` должна иметь следующий формат:

```
std::vector<std::pair<int, int>> find_connections (int N);
```

Она будет вызвана один раз программой жюри с одним параметром — количеством полей. Функция должна вернуть список упорядоченных пар, описывающих найденные стрелки между полями. Порядок упорядоченных пар в списке не имеет значения.

Функция для задавания вопросов Боби имеет следующий формат:

```
std::vector<bool> check (std::vector<int> p);
```

Параметр p — это перестановка чисел от 1 до N в задаче. Результат — это булев вектор b размера N , где $b_i = 1$ ($0 \leq i < N$) тогда и только тогда, когда выполняется одно из следующих условий:

- существует стрелка от поля p_j к полю p_i при $i < j$;
- существует поле p_j , для которого $b_j = 1$, и можно пройти по стрелкам от p_j к p_i .

XVI INTERNATIONAL ADVANCED TOURNAMENT IN INFORMATICS SHUMEN 2025

Однозначность возвращаемых значений следует из свойств стрелок.

Если последовательность не является перестановкой чисел от 1 до N , вы получите **Wrong answer** для теста. Функция выполняется за $O(N)$. Вы можете вызвать эту функцию не более $\frac{10^8}{N}$ раз, иначе вы получите **Wrong answer**.

Ваша программа **monopoly2** должна реализовать функцию **find_connections**. Она также может содержать другой код, функции и глобальные переменные, но не должна содержать функцию **main**. Кроме того, вы не должны считывать данные из стандартного ввода или выводить данные в стандартный вывод. Ваша программа должна включать заголовочный файл **monopoly2.h** с помощью инструкции для препроцессора:

```
#include "monopoly2.h"
```

Ограничения

- ♣ $1 \leq N \leq 1\,000$;
- ♣ $M = N - 1$.

Подзадачи

Подзадача	Баллы	Необходимые подзадачи	N	Другие ограничения
0	0	—	—	Карта из примера взаимодействия.
1	5	—	$\leq 1\,000$	$1, 2, \dots, N$ — единственное <i>корректное</i> упорядочивание полей.
2	6	0	≤ 6	—
3	17	1	$\leq 1\,000$	Перестановка $1, 2, \dots, N$ также может быть получена следующим образом: выписать поле 1, затем выписать поля к которым есть прямые стрелки от 1, после этого выписать поля к которым ведут прямые стрелки от второго выписанного поля, и так далее.
4	13	0, 2	≤ 300	—
5	59	0 — 4	$\leq 1\,000$	Если мы начинаем с поля 1 и следуем по стрелкам, мы не сможем использовать более 25 стрелок.

Баллы за каждую подзадачу начисляются только в том случае, если все тесты для нее и необходимые подзадачи были **успешно** пройдены, и баллы равны минимальному баллу теста в ней и необходимых подзадачах, умноженному на баллы подзадачи.

Система оценивания

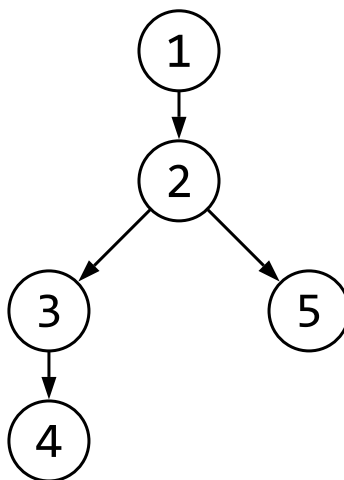
Каждый тест получает балл, который является дробным числом между 0 и 1 включительно. Если тест имеет положительный балл, он считается **успешным** для вашего решения. Тест имеет положительный балл, если вы успешно находите стрелки между полями.

Если cnt — это количество вызовов функции `check` для данного теста, то балл теста рассчитывается следующим образом:

- Для всех подзадач: если $cnt > \frac{10^8}{N}$, то балл равен 0.
- Подзадачи 0, 1 и 2.
 - Если $cnt \leq \frac{10^8}{N}$, то балл равен 1.
- Подзадача 3.
 - Если $cnt \leq 10\,000$, то балл равен 1.
 - Если $10\,000 < cnt \leq 15\,000$, то балл равен $\min(\frac{1000}{cnt-9000}, 1)$.
 - Если $15\,000 < cnt \leq \frac{10^8}{N}$, то балл равен 0.1.
- Подзадача 4.
 - Если $cnt \leq 50\,000$, то балл равен 1.
 - Если $50\,000 < cnt \leq 200\,000$, то балл равен $\min(\frac{25000}{cnt-25000}, 1)$.
 - Если $200\,000 < cnt \leq \frac{10^8}{N}$, то балл равен 0.1.
- Подзадача 5.
 - Если $cnt \leq 170$, то балл равен 1.
 - Если $170 < cnt \leq 1000$, то балл равен $\min((\frac{170+3000}{cnt+3000})^{\frac{3000}{cnt}}, 1)$.
 - Если $1000 < cnt \leq 20\,000$, то балл равен $\min((\frac{170}{cnt})^{0.4}, 0.5)$.
 - Если $20\,000 < cnt \leq \frac{10^8}{N}$, то балл равен 0.1.

Пример взаимодействия

Пусть у нас есть следующая иллюстрация карты с 5 полями и 4 стрелками:



Действия вашей программы	Действия и ответы жюри
	<code>find_connections(5)</code>
<code>check({2, 3, 4, 5, 1})</code>	<code>return {1, 1, 1, 1, 0}</code>
<code>check({1, 5, 2, 3, 4})</code>	<code>return {0, 1, 0, 0, 0}</code>
<code>check({1, 4, 3, 2, 5})</code>	<code>return {0, 1, 1, 0, 0}</code>
<code>return {{1, 2}, {2, 3}, {3, 4}, {2, 5}}</code>	

Локальное тестирование

Для локального тестирования предоставлены следующие файлы: `monopoly2.h`, `Lgrader.cpp`, файл-пример `monopoly2.cpp` для вашей программы и файл с картой из примера взаимодействия. Если предоставленные файлы находятся в одной папке, вы можете скомпилировать вашу программу `monopoly2.cpp` вместе с `Lgrader.cpp`. Это создаст программу для проверки правильности вашей функции.

Программа считывает из стандартного ввода следующую последовательность чисел:

- в первой строке: одно натуральное число — количество полей N ;
- в каждой из следующих $N - 1$ строк два натуральных числа, описывающих стрелку.

Если вы не следуете протоколу взаимодействия, вы получите соответствующее сообщение об ошибке. В противном случае, если программа успешна, вы получите сообщение "Correctly found connections."