

XVI INTERNATIONAL ADVANCED TOURNAMENT IN INFORMATICS
ПРОЛЕТНИ СЪСТЕЗАНИЯ ПО ИНФОРМАТИКА, ШУМЕН 2025

Задача C23. МОНОПОЛИ 2

 1 sec.  256 MB

След като помогнахте на **Deni** да направи перфектната карта за игра на монопол през 2021 г., тя отново се нуждае от вашата помощ! Припомняме, че игралната карта за монопол се състои от N полета (номерирани от 1 до N) и M еднопосочни връзки между тях. Тук $M = N - 1$. Еднопосочните връзки отговарят на следните условия – няма връзка от поле към себе си и няма различни връзки между една и съща двойка полета. Връзките имат свойството, че всяко поле е достъпно от поле 1.

Дефиниция: Нека имаме пермутация на числата от 1 до N . Ние наричаме пермутацията *подходящо* подреждане на полетата, ако е изпълнено следното условие – за всяка връзка от поле i към поле j , i трябва да бъде **преди** j в пермутацията.

Проблемът за **Deni** е, че тя загуби идеалната карта за игра на монопол. За щастие нейният приятел Боби си спомня картата, но той реши да се позабавлява с нея, преди да ѝ каже кои са връзките. Първо, Боби казва на **Deni**, че *подходящото* подреждане на полетата е последователността $1, 2, \dots, N$. След това Боби ще отговори на няколко въпроса от **Deni**, за да ѝ помогне да разбере какви са връзките между полетата. Всеки въпрос ще бъде за това доколко *подходяща* е някаква пермутация на числата от 1 до N (за повече подробности вижте раздела „Подробности за имплементацията“). Нашата героиня иска вашата помощ, за да измисли и приложи стратегия с възможно най-малко въпроси.

Задача

Напишете програма **monopoly2**, която намира неизвестните връзки с възможно най-малко въпроси към Боби. Тя трябва да съдържа функцията `find_connections`, която ще се компилира с програмата на журито (за ролята на Боби).

Подробности за имплементацията

Вашата функция `find_connections` трябва да има следния формат:

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

Тя ще бъде извикана еднократно от програмата на журито с един параметър – брой полета. Функцията трябва да върне списък от подредени двойки, описващи намерените връзки между полетата. Редът на наредените двойки в списъка няма значение.

Функцията за задаване на въпроси към Боби има следния формат:

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

Параметърът p е пермутацията на числата от 1 до N във въпроса. Резултатът е булев вектор b с размер N , където $b_i = 1$ ($0 \leq i < N$) тогава и само тогава, когато е изпълнено едно от следните условия:

- съществува връзка от поле p_j към поле p_i при $i < j$;
- съществува поле p_j за което $b_j = 1$ и вие може да се движите по връзките от p_j до p_i .

Забележете, че еднозначността на стойностите следва от свойствата на връзките.

XVI INTERNATIONAL ADVANCED TOURNAMENT IN INFORMATICS ПРОЛЕТНИ СЪСТЕЗАНИЯ ПО ИНФОРМАТИКА, ШУМЕН 2025

Ако последователността не е валидна пермутация на числата от 1 до N , ще получите Wrong answer за теста. Сложността на функцията е $O(N)$. Можете да извикате тази функция най-много $\frac{10^8}{N}$ пъти, в противен случай ще получите Wrong answer.

Вашата програма **monopoly2** трябва да имплементира функцията **find connections**. Може да съдържа и друг код, функции и глобални променливи, но не трябва да съдържа функцията **main**. Освен това не трябва да четете от стандартния вход или да печатате към стандартния изход. Вашата програма трябва да включва заглавния файл **monopoly2.h** чрез инструкцията към препроцесора:

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

Ограничения

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

Subtasks

Подзадача	Точки	Необходими подзадачи	N	Други ограничения
0	0	—	—	Картата от примерната комуникация.
1	5	—	$\leq 1\,000$	$1, 2, \dots, N$ е единственото <i>подходящо</i> подреждане на полетата.
2	6	0	≤ 6	—
3	17	1	$\leq 1\,000$	$1, 2, \dots, N$ могат да бъдат получени и чрез започване с полето 1, след това изброяване на полетата с директни връзки от 1, след това изброяване на полета с едноръчни връзки от второто добавено поле и т.н.
4	13	0, 2	≤ 300	—
5	59	0 — 4	$\leq 1\,000$	Ако започнем от поле 1 и следваме връзките, не можем да използваме повече от 25 връзки.

Точките за дадена подзадача се получават само ако всички тестове за нея и задължителните подзадачи са **успешно** издържани, като точките са равни на минималния резултат от теста за нея и задължителните подзадачи, умножен по точките на подзадачата.

Оценяване

Всеки тест получава резултат, който е дробно число между 0 и 1 включително. Ако даден тест има положителен резултат, той се счита **успешен** за вашето

XVI INTERNATIONAL ADVANCED TOURNAMENT IN INFORMATICS ПРОЛЕТНИ СЪСТЕЗАНИЯ ПО ИНФОРМАТИКА, ШУМЕН 2025

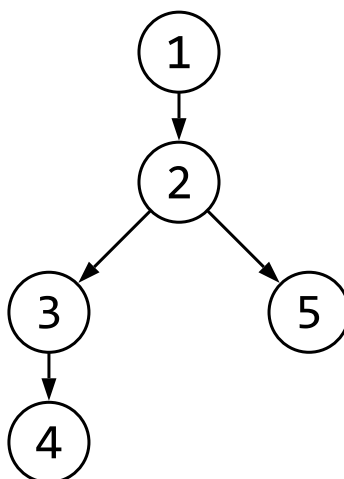
решение. Тестът има положителен резултат, ако успешно откриете връзките между полетата.

Ако cnt е броят на извикванията на функцията `check` за определен тест, тогава резултатът от теста се изчислява по следния начин:

- За всички подзадачи: ако $cnt > \frac{10^8}{N}$, тогава резултатът е равен на 0.
- Подзадача 0, 1, и 2.
 - Ако $cnt \leq \frac{10^8}{N}$, тогава резултатът е равен на 1.
- Подзадача 3.
 - Ако $cnt \leq 10\,000$, тогава резултатът е равен на 1.
 - Ако $10\,000 < cnt \leq 15\,000$, тогава резултатът е равен на $\min(\frac{1000}{cnt-9000}, 1)$.
 - Ако $15\,000 < cnt \leq \frac{10^8}{N}$, тогава резултатът е равен на 0.1.
- Подзадача 4.
 - Ако $cnt \leq 50\,000$, тогава резултатът е равен на 1.
 - Ако $50\,000 < cnt \leq 200\,000$, тогава резултатът е равен на $\min(\frac{25000}{cnt-25000}, 1)$.
 - Ако $200\,000 < cnt \leq \frac{10^8}{N}$, тогава резултатът е равен на 0.1.
- Подзадача 5.
 - Ако $cnt \leq 170$, тогава резултатът е равен на 1.
 - Ако $170 < cnt \leq 1000$, тогава резултатът е равен на $\min((\frac{170+3000}{cnt+3000})^{\frac{3000}{cnt}}, 1)$.
 - Ако $1000 < cnt \leq 20\,000$, тогава резултатът е равен на $\min((\frac{170}{cnt})^{0.4}, 0.5)$.
 - Ако $20\,000 < cnt \leq \frac{10^8}{N}$, тогава резултатът е равен на 0.1.

Пример за комуникация

Нека имаме следната илюстрация на карта с 5 полета и 4 връзки:



Действия на вашата програма	Отговор на журито
	<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>	

XVI INTERNATIONAL ADVANCED TOURNAMENT IN INFORMATICS
ПРОЛЕТНИ СЪСТЕЗАНИЯ ПО ИНФОРМАТИКА, ШУМЕН 2025

Локално тестване

За локално тестване се предоставят следните файлове: `monopoly2.h`, `Lgrader.cpp`, примерен файл `monopoly2.cpp` за вашата програма и файл с картата от примерната комуникация. Когато предоставените файлове са в една и съща папка, можете да компилирате заедно вашата програма `monopoly2.cpp` и `Lgrader.cpp`. Това ще направи програма за проверка на коректността на вашата функция.

Програмата ще изисква от стандартния вход следната последователност от числа:

- на първия ред: едно цяло положително число – брой на интервалите N ;
- на всеки от следващите $N - 1$ реда две положителни цели числа, които описват еднопосочни връзки.

Ако не следвате протокола за комуникация, ще получите подходящо съобщение за грешка. В противен случай, ако програмата е успешна, ще получите съобщението „Правилно намерени връзки.“.