Задача А3. Дърво

Иван има колекция от T на брой дървета, събирани през годините. Всяко дърво има N върха, номерирани от 1 до N, и един специален връх u сред тях.

Дамян е виждал само как изглежда колекцията, но не и кои са специалните върхове. Сега той е много настойчив да ги научи, затова Иван решил да му даде шанс като го предизвика на следната игра:

Дамян получава структурата на дървото и може да пита по няколко въпроса наведнъж, от следния тип: за някой връх X и непразно множество от върхове V, дали

$$\min_{v \in V} \text{dist}(v, u) \geq \text{dist}(X, u)$$

където с u означаваме специалния връх в текущото дърво, а с $\text{dist}(a, b)$ означаваме броят ребра по уникалния път между върховете a и b в дървото. Иван му отговаря едновременно на всички въпроси, като обменът на група въпроси и отговори може да се случи няколко пъти.

Понеже Дамян не може да се справи сам с този проблем, той се обърнал към вас за помощ. Вие трябва да му помогнете да открие специалните върхове на Иван, като използвате възможно най-малък брой групи и възможно най-малък общ брой въпроси.

Детайли по имплементацията

Кодът, който предавате, не трябва да съдържа функция `main`, а трябва да имплементирате функцията `find_node`:

```
int find_node(int N, std::vector<int> A , std::vector<int> B)
```

Функцията `find_node` има следните параметри:

- N: Броят на върховете в текущото дърво.
- A, B: 2 вектора с дължина $N-1$, където A_i, B_i са върховете на i-тото ребро в дървото.

Функцията `find_node` ще бъде извикана T пъти, по веднъж за всяко дърво на Иван, като всеки път трябва да върне какъв е номерът на специалния връх u. За да задавате въпроси на Иван, може да използвате функцията `query`:

```
std::vector<bool> query
(std::vector<std::pair<int,std::vector<int>>> v)
```

Функцията `query` има следните параметри:

- v: Вектор, в който се съдържат всички въпроси, зададени към Иван.

Всеки въпрос е представен като двойка (`pair<int, std::vector<int>>`) (X_i, V_i), където X_i е върхът, а V_i е множеството от върхове, за които се пита дали $\min_{v \in V_i} \text{dist}(v, u) \geq \text{dist}(X_i, u)$. Иван ви връща отговорите си под формата на вектор от булеви стойности, където i-тата стойност е отговорът на i-тия въпрос. Ако V_i е празно, или някой от върховете е с невалиден номер, ще получите `Invalid call`.

Ограничения

- $\sum N \leq 50\,000$
- $300 \leq N$
- u е фиксиран предварително за всеки тест, т.е. грейдърът не е адаптивен.

Подзадачи

Подзадача	Точки	Необходими подзадачи	$\sum N$	Други ограничения
1	13	–	≤ 2000	–
2	10	–	–	Всяко дърво е пълно двоично дърво (всеки връх има 0 или 2 деца), считано от корен връх 1.
3	10	2	–	Всяко дърво е двоично (всеки връх има най-много по 2 деца), считано от корен връх 1.
4	8	–	–	Всяко дърво има формата на пръчка (всеки връх има най-много 2 съседа) и връх 1 има точно един съсед.
5	7	–	–	Всяко дърво има формата на звезда (всеки връх освен един има по 1 съсед) и връх 1 има $N-1$ съседа.
6	20	1	$\leq 10^4$	–
7	32	$1-6$	–	–

Точките за дадена подзадача се получават само ако се преминат успешно всички тестове, предвидени за нея и необходимите подзадачи.

Оценяване

Оценяването на даден подтест зависи от броя извиквания на функцията `query`, които ще означим с C, и общия брой на въпросите, които сте задали, измежду всички извиквания, Q. Изчисляват се два коефициента $coef_C$ и $coef_Q$ спрямо тях.

Нека $L_1 = \lceil \log_{1.5}(N) \rceil$ и $L_2 = \lceil \log_2^2(N) \rceil$.

- Ако $C \leq 2$, то $coef_C = 1.0$
- Ако $C = 3$, то $coef_C = 0.8$
- Ако $3 < C \leq L_1$, то $coef_C = 0.7$
- Ако $L_1 < C \leq L_2$, то $coef_C = 0.7 - 0.6\frac{ln(C)-ln(L_1)}{ln(L_2)-ln(L_1)}$
- Ако $L_2 < C$, то $coef_C = 0.1$

Нека $target = max(C, \lfloor \sqrt[max(C,2)]{8N} \rfloor)$.

- Ако $Q \leq target$, то $coef_Q = 1.0$
- Ако $target < Q \leq 3 \times target$, то $coef_Q = 0.8 - 0.7\frac{ln(Q)-ln(target)}{ln(3\times target)-ln(target)}$
- Ако $3 \times target < Q$, то $coef_Q = 0.1$

Резултатът за подтеста е $coef_C \times coef_Q$. Частта от точките за целия тест, която получавате, е минималният резултат на негов подтест, а за подзадачата е минималния резултат на неин тест.

Локален грейдър

В папката `contestant` са предоставени файлове `Lgrader.cpp` и `tree.h`, които може да компилирате заедно с програмата си и да използвате за локално тестване. При изпълнение на `main` функцията на `Lgrader.cpp` се прочитат данните за дърветата, във формата, описан по-долу, след което на стандартния изход се отпечатват резултатите от всяко извикване на `find_node`.

Формат на входа:

- ред 1: T – броят на дърветата
- ред 2: N_1 – броят на върховете в първото дърво
- ред 3: $A_1[0]$ $B_1[0]$
- ред 4: $A_1[1]$ $B_1[1]$
- ⋮
- ред $N_1 + 1$: $A_1[N_1 - 2]$ $B_1[N_1 - 2]$
- ред $N_1 + 2$: u_1
- ред $N_1 + 3$: N_2 – броят на върховете във второто дърво
- ред $N_1 + 4$: $A_2[0]$ $B_2[0]$
- ред $N_1 + 5$: $A_2[1]$ $B_2[1]$
- ⋮

Формат на изхода:

- ред 1: C_1 Q_1
- ред 2: C_2 Q_2
- ⋮

Примерна комуникация

В този пример $T = 1$, $N = 5$ и ребрата са $(1, 2), (2, 3), (2, 4), (4, 5)$ и специалният връх е $u = 4$. В този случай $C = 1$ и $Q = 3$.

Вашата програма	Програма на журито	Обяснение
	`find_node(5, {1,2,2,4}, {2,3,4,5})`	
`query({{2, {3}}, {1, {5}}, {4, {2, 5}}})`	`return {true, false, true}`	$\text{dist}(u, 3) = 2 \geq \text{dist}(u, 2) = 1$; $\text{dist}(u, 5) = 1 < \text{dist}(u, 1) = 2$; $\min(1, 1) = 1 \geq \text{dist}(u, 4) = 0$
`return 4`		

Забележка: тук N не попада в ограниченията на задачата. В реалните тестове те са спазени.