

Reconstruct

Временско ограничување: 4 секунди

Мемориско ограничување: 1024 MB

Ова е интерактивна задача.

Бисерка е во ситуација да влезе во волонтерски организирана ориентациска игра *Final destination*, каде делегациите од EJOI ќе се натпреваруваат да посетат различни локации низ градот. Но Бисерка е повеќе заинтересирана за тоа *како* играта е дизајнирана отколку да излезе како победник од неа.

Има N локации на мапата и N тимови. Секој тим мора да ги посети сите локации и има единствена почетна локација — тимот i започнува од локацијата i . На секој тим му е дадена негова рута.

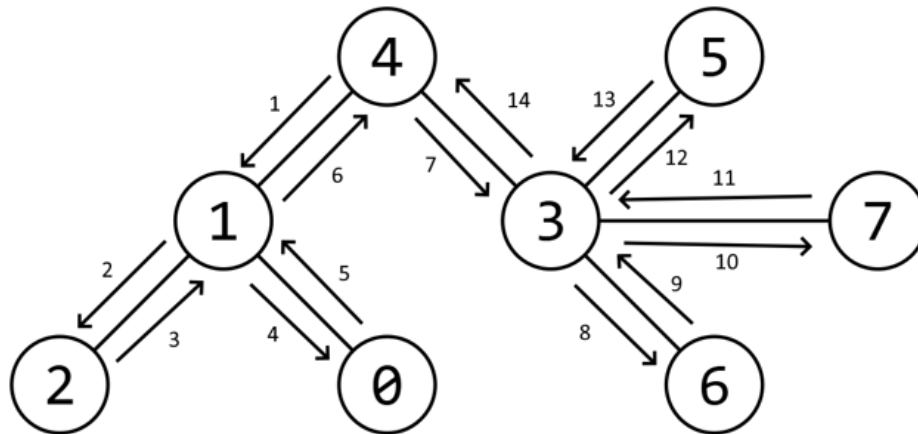
Бисерка исто така има некои внатрешни информации — рутите се засноваат на скриена структура од брзи линии на јавен превоз. Попрецизно, жирито на натпреварот избрало $N - 1$ линии, пришто секоја поврзува пар од локации, така што секоја локација е достиглива од која било друга локација користејќи ги овие линии. Ваквата структура се нарекува и дрво. Потоа, за секоја почетна локација i , жирито генерирало **произволна DFS ѝрошејка** (дефинирана подолу) и ја доделило оваа прошетка како рута за тимот i .

Бисерка сака да открие кое дрво на јавен транспорт го избрала комисијата. Таа поставува само брзи прашања од видот:

- Која е j -тата дестинација на i -тиот тим?

Помогнете ѝ на Бисерка така што ќе напишете програма `find_tree` што го наоѓа скриеното дрво.

DFS-ѝрошејка на дрво е редоследот по кој неговите темиња се посетуваат за прв пат од процедура за пребарување прво по длабочина (анг. depth-first-search). Таквата процедура започнува од дадено теме v и рекурзивно оди до еден од неговите непосетени соседи сè додека не се случи таков сосед да не постои — тогаш се враќа на претходно посетеното теме и продолжува од таму. Пример:



Овде $v = 4$ и стрелките долж ребрата одговараат на чекорите на пребарувањето прво по длабочина. Генерираната прошетка е $[4, 1, 2, 0, 3, 6, 7, 5]$. Забележете дека редоследот по кој ги посетуваме соседите е важен; друга прошетка би можела да биде $[4, 3, 5, 7, 6, 1, 0, 2]$.

Дрво-структурата и N -те *DFS* *прошетки* се фиксни пред да започне вашата програма и не се менуваат како одговор (т.е. врз основа) на вашите прашања. Редоследите на соседи на темињата што се користат во текот на таквите *DFS*-а може да биде различен.

Имплементациски детали

Треба да ја имплементирате следната функција:

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

- овде N е бројот на локации;
- повратната вредност е листа од ребрата на дрвото — редоследот на ребрата и нивните крајни точки (темиња) не е важен.

За секој тест, оваа функција може да се повика најмногу до T пати.

За да комуницирате со системот за оценување, може да ја повикувате следната функција:

```
int guess(int i, int j)
```

Таа ја враќа j -тата локација во i -тата прошетка на дрвото. Да забележиме дека `guess(i, 0)` ќе го врати i . Може да претпоставите дека функцијата одговара во константно време $O(1)$ за сите подзадачи заклучно со подзадачата 6, и во логаритамско време $O(\log N)$ за подзадачата 7. Треба да важи дека $0 \leq i, j \leq N - 1$, во спротивно вашето решение ќе ја добие пресудата `Output isn't correct: Invalid call`.

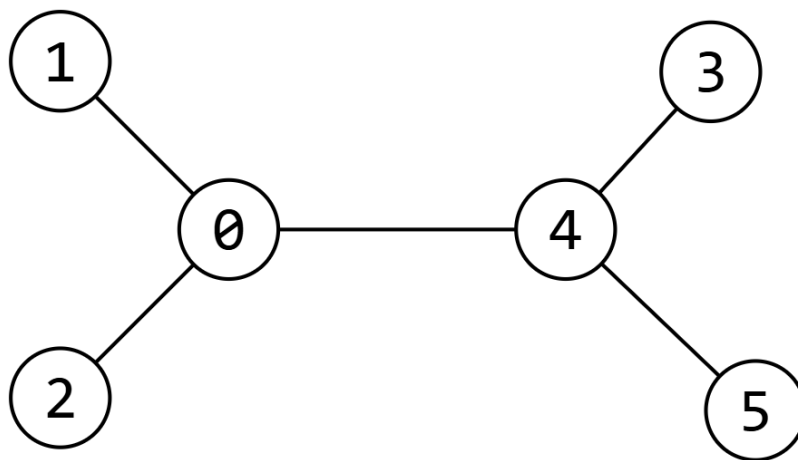
Ограничувања

- $2 \leq N \leq 2^{16} + 1$
- T ги следи ограничувањата врз основа на максималното N (означено со N_{max}) низ сите функционални повици во еден тест:
 - ако $N_{max} \leq 9$, тогаш $1 \leq T \leq 100$
 - ако $N_{max} \leq 2^{10} + 1$, тогаш $1 \leq T \leq 10$
 - ако $N_{max} \leq 2^{16} + 1$, тогаш $1 \leq T \leq 3$

Системот за оценување може да користи до 280 MB, што се брои во меморијата што ја користи вашето решение.

Пример

Претпоставете дека скриената структура на линиите на брз јавен превоз е следната:



Претпоставете дека за почетната локација 0, прошетката е $[0, 1, 2, 4, 3, 5]$. Тогаш следново е пример за интеракција:

Програма на натпреварувач	Програма оценувач
	<code>find_tree(6)</code>
<code>guess(0, 0)</code>	0
<code>guess(0, 1)</code>	1
<code>guess(0, 2)</code>	2
<code>guess(0, 3)</code>	4
<code>guess(0, 4)</code>	3
<code>guess(0, 5)</code>	5
<code>return {{0,1},{0,2},{4,0},{5,4},{3,4}};</code>	

Забележете дека информациите од прашанките не се доволни за еднозначно да се определи дрвото, но ова всушност е одговорот на тестот во подзадачата 0.

Подзадачи

Подзадача	Поени	N	Дополнителни ограничувања
0	0	-	Примерот.
1	11	≤ 9	-
2	6	≤ 100	Секое теме е поврзано најмногу со 2 други.
3	13	≤ 100	Секоја прошетка е генерирана од пребарување прво по длабочина кое во секој чекор приоретизира оддалечување од темето 0. Ако постојат повеќе опции за оддалечување од темето 0, произволно се избира една.
4	11	≤ 100	Секое теме, што не е темето 0, е поврзано најмногу со 2 други темиња.
5	10	≤ 100	-
6	31	$\leq 2^{10} + 1$	-
7	18	$\leq 2^{16} + 1$	-

Бодување

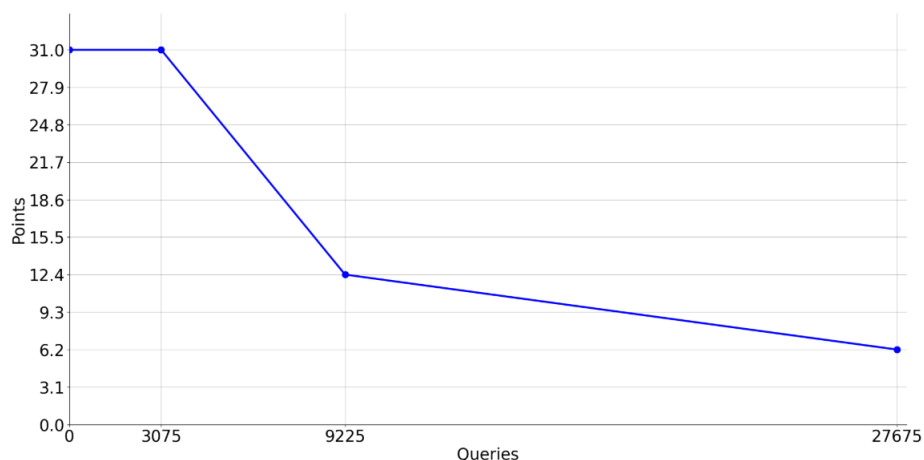
За секоја подзадача заклучно со подзадачата 5, добивате целосни поени ако успешно го најдете дрвото во рамките на временските ограничувања. За подзадачите 6 и 7, делот S од целосните поени за даден тест зависи од **максималниот** број на прашанки што ќе ги поставите во еден од неговите подтестови, Q_{max} . Бодувањето за овие подзадачи е следново:

- ако $Q_{max} \leq L_1$, тогаш $S = 1.0$;
- ако $L_1 < Q_{max} \leq L_2$, тогаш $S = 0.4 + 0.6 \times \frac{L_2 - Q_{max}}{L_2 - L_1}$;
- ако $L_2 < Q_{max} \leq 3L_2$, тогаш $S = 0.2 + 0.2 \times \frac{3L_2 - Q_{max}}{2L_2}$;
- ако $3L_2 < Q_{max}$, тогаш $S = 0.2$;

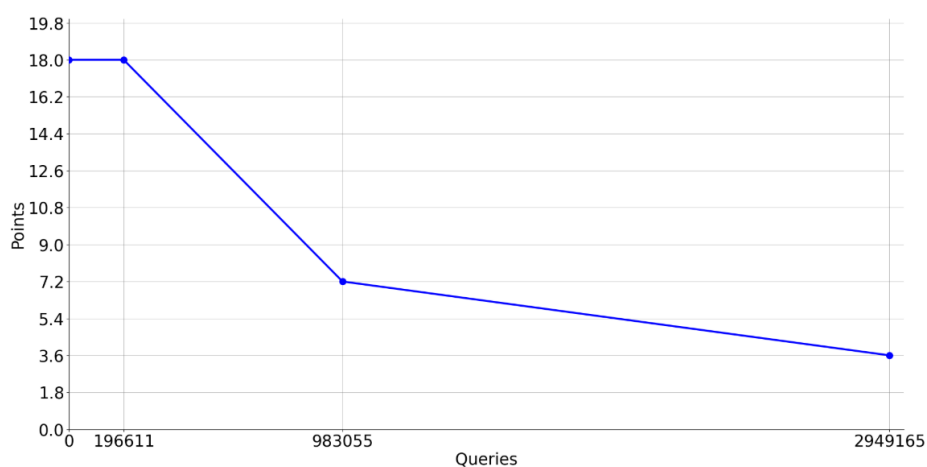
каде $L_1 = 3 \times (2^{10} + 1) = 3075$ и $L_2 = 9 \times (2^{10} + 1) = 9225$ за подзадачата 6, и $L_1 = 3 \times (2^{16} + 1) = 196611$ и $L_2 = 15 \times (2^{16} + 1) = 983055$ за подзадачата 7. Делот од поените што ќе го добиете од целата подзадача е минималното S низ сите тестови.

Графиците подолу ја покажуваат функцијата на бодување за последните две подзадачи:

Подзадача 6



Подзадача 7



Пример-оценувач

Дадени се два пример-оценувачи.

За локално тестирање: `Lgrader.cpp` може да се компајлира заедно со вашата програма за да обезбеди алатка за тестирање. Програмата го чита бројот T на тест-случаи и за секој го очекува бројот на локации N проследен со $N - 1$ линии, секоја со пар цели броеви што опишуваат директна линија. Потоа следуваат N линии со N цели броеви – DFS прошетките. Забележете дека прошетката i мора да започне од темето i . Можете да дозволите оценувачот да генерира прошетки за вас со поставување на константата `AUTO_GENERATE` на `true`. Како излез, оценувачот или пријавува порака за грешка ако резултатот е неточен, или го дава бројот на поставени прашанки за секој тест-случај и целокупниот максимум. Исто така, забележете дека оценувачот не работи за доволно големо N (поточно ограничувањата на подзадачата 7) - слободно користете ја вашата имажинација за тој случај.

За системските кориснички тестови: `stub.cpp` може да се користи заедно со вашата програма за корисничките тестови на системот. Форматот на влезот е ист како оној за другиот оценувач. Забележете дека за корисничките тестови нема вградена опција за автоматско генерирање на прошетка.