

Reconstruct

Vremensko ograničenje: 4 sekunde

Memorijsko ograničenje: 1024 MiB

Ovo je interaktivni zadatak.

Biserka se sprema da uđe u orijentacijsku igru *Final destination*, koju organizuju volonteri, u kojoj će se EJOI delegacije takmičiti u posjećivanju različitih lokacija širom grada. Ali Biserku više zanima *kako* je igra dizajnirana, nego da pobijedi u njoj.

Na mapi se nalazi N lokacija i N timova. Svaki tim mora posjetiti sve lokacije i ima jedinstvenu početnu lokaciju — tim i počinje od lokacije i . Svakom timu je dodijeljena vlastita ruta.

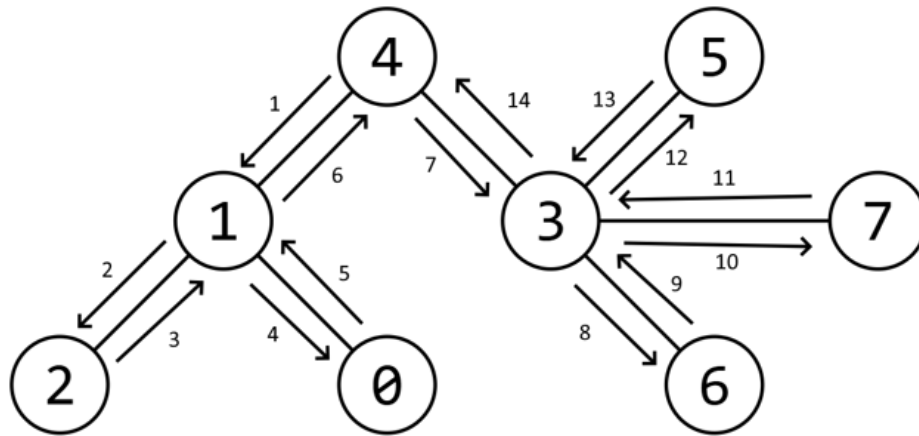
Biserka također ima neke insajderske informacije — rute su bazirane na skrivenoj strukturi brzih linija javnog prijevoza. Preciznije, žiri takmičenja je odabrao $N - 1$ linija, od kojih svaka povezuje par lokacija, tako da je svaka lokacija dostižna iz bilo koje druge koristeći te linije. Takva struktura se također naziva stablo. Zatim je, za svaku početnu lokaciju i , žiri generisao **proizvoljan DFS obilazak** (definisan ispod) i dodijelio taj obilazak kao rutu za tim i .

Biserka želi otkriti stablo javnog prijevoza koje je žiri odabrao. Ona postavlja samo brza pitanja sljedećeg tipa:

- Koja je j -ta destinacija i -tog tima?

Pomozi Biserki tako što ćeš napisati program `find_tree` koji pronalazi skriveno stablo.

DFS obilazak stabla je redoslijed kojim se njegovi čvorovi posjećuju po prvi put tokom procedure dubinskog pretraživanja (DFS-a). Takva procedura počinje u datom čvoru v i rekurzivno obilazi jednog od svojih neposjećenih susjeda, sve dok ne dođe do čvora koji nema neposjećenih susjeda – tada se vraća u posljednji prethodno posjećeni čvor i nastavlja odatle. Primjer:



Ovdje je $v = 4$, a strelice duž grana odgovaraju koracima DFS-a. Generisani obilazak je $[4, 1, 2, 0, 3, 6, 7, 5]$. Primijeti da redoslijed kojim posjećujemo susjede utiče na rezultat; drugi mogući obilazak je $[4, 3, 5, 7, 6, 1, 0, 2]$.

Struktura stabla i N DFS obilazaka su određeni prije nego što se tvoj program pokrene i ne mijenjaju se uslijed tvojih pitanja. Redoslijedi susjeda čvorova korišteni tokom takvih DFS-ova mogu biti različiti.

Implementacijski detalji

Funkcija koju moraš implementirati je:

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

- ovdje je N broj lokacija;
- povratna vrijednost je lista ivica stabla — redoslijed ivica i njihovih krajnjih tačaka nije bitan.

Za svaki testni primjer, ova funkcija se može pozvati najviše T puta.

Da bi komunicirao sa žirijem, možeš pozivati sljedeću funkciju:

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

Ona vraća j -tu lokaciju u i -tom obilasku stabla. Primijeti da će `guess(i, 0)` vratiti i . Možeš pretpostaviti da funkcija odgovara u konstantnom vremenu $O(1)$ za sve podzadatke do 6, i u logaritamskom vremenu $O(\log N)$ za podzadatak 7. Mora vrijediti $0 \leq i, j \leq N - 1$, u suprotnom će tvoje rješenje dobiti odluku `Output isn't correct: Invalid call`.

Ograničenja

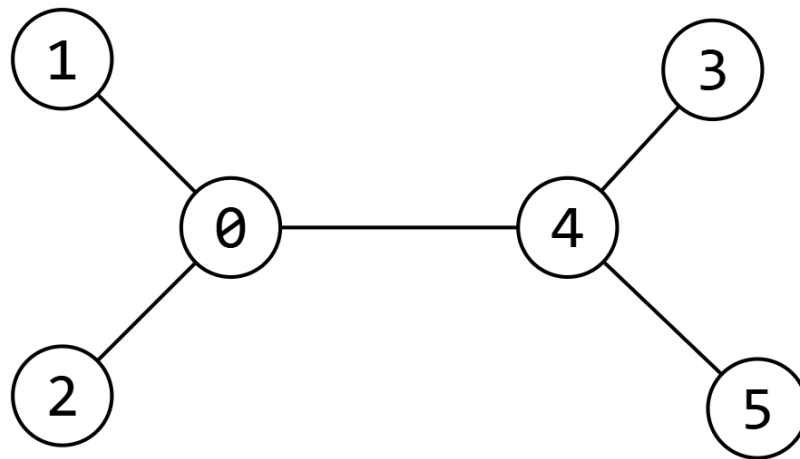
- $2 \leq N \leq 2^{16} + 1$

- T prati ograničenja bazirana na maksimalnom N (označenom sa N_{max}) među pozivima funkcije u jednom testnom primjeru:
 - ako je $N_{max} \leq 9$, tada je $1 \leq T \leq 100$
 - ako je $N_{max} \leq 2^{10} + 1$, tada je $1 \leq T \leq 10$
 - ako je $N_{max} \leq 2^{16} + 1$, tada je $1 \leq T \leq 3$

Sistemske grader može koristiti do 280 MiB, što se računa u memoriju tvog rješenja.

Primjer

Pretpostavi da je skrivena struktura brzih linija javnog prijevoza sljedeća:



Pretpostavi da je za početnu lokaciju 0 obilazak $[0, 1, 2, 4, 3, 5]$. Tada je sljedeće primjer interakcije:

Tvoj program	Program žirija
	<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>	

Primijeti da informacije dobijene iz upita nisu dovoljne da jednoznačno odrede stablo, ali ovo je zapravo tačan odgovor za testni primjer u podzadatku 0.

Podzadaci

Podzadatak	Bodovi	N	Dodatna ograničenja
0	0	-	Primjer.
1	11	≤ 9	-
2	6	≤ 100	Svaki čvor je povezan sa najviše 2 druga čvora.
3	13	≤ 100	Svaki obilazak je generisan dubinskim pretraživanjem koje u svakom koraku daje prioritet udaljavanju od čvora 0. Ako postoji više opcija za udaljavanje od čvora 0, jedna od njih se bira proizvoljno.
4	11	≤ 100	Bilo koji čvor osim čvora 0 je povezan sa najviše 2 druga čvora.
5	10	≤ 100	-
6	31	$\leq 2^{10} + 1$	-
7	18	$\leq 2^{16} + 1$	-

Bodovanje

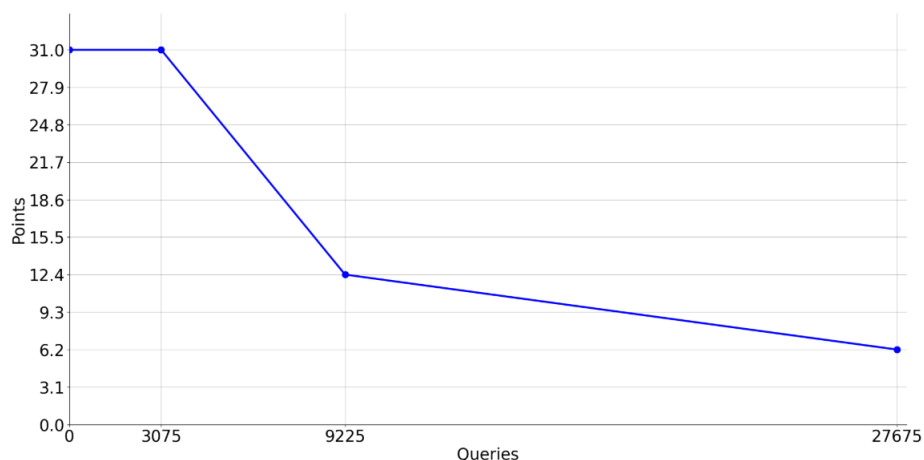
Za svaki podzadatak do podzadatka 5, dobijaš pun broj bodova ako uspješno pronađeš stablo u okviru vremenskog ograničenja. Za podzadatke 6 i 7, udio punog broja bodova S za dati testni primjer zavisi od maksimalnog broja upita koje postaviš na jednom od njegovih podtestova, Q_{max} . Bodovanje za ove podzadatke je sljedeće:

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

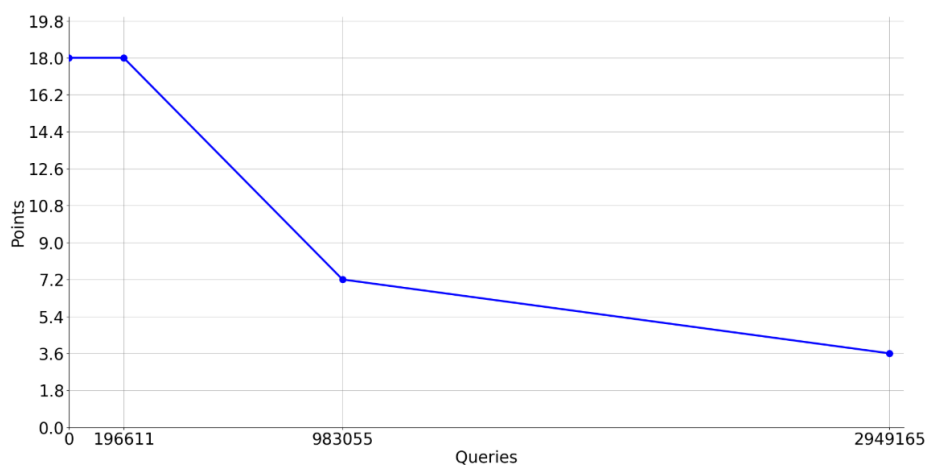
gdje su $L_1 = 3 \times (2^{10} + 1) = 3075$ i $L_2 = 9 \times (2^{10} + 1) = 9225$ za podzadatak 6, a $L_1 = 3 \times (2^{16} + 1) = 196611$ i $L_2 = 15 \times (2^{16} + 1) = 983055$ za podzadatak 7. Udio bodova koji dobijaš za cijeli podzadatak je minimalna vrijednost S među svim testnim primjerima.

Grafici ispod prikazuju funkciju bodovanja za posljednja dva podzadatka:

Podzadatak 6



Podzadatak 7



Sample grader

Dostupna su dva sample gradera.

Za lokalno testiranje: `Lgrader.cpp` se može kompajlirati zajedno sa tvojim programom kako bi se dobio alat za testiranje. Program čita broj T testnih primjera i za svaki od njih očekuje broj lokacija N , nakon čega slijedi $N - 1$ linija, svaka sa parom cijelih brojeva koji opisuju direktnu liniju. Nakon toga slijedi N linija sa po N cijelih brojeva – DFS obilasci. Primijeti da obilazak i mora početi od čvora i . Možeš dozvoliti graderu da generiše obilaske umjesto tebe, postavljanjem konstante `AUTO_GENERATE` na `true`. Kao izlaz, grader ili prijavljuje poruku o grešci ako je rezultat netačan, ili broj upita postavljenih za svaki testni primjer, kao i ukupni maksimum. Također, primijeti da grader ne radi za značajno veće vrijednosti N (preciznije, za ograničenja podzadatka 7) – u tom slučaju slobodno koristi svoju maštu.

Za korisničke testne primjere na sistemu: `stub.cpp` se može koristiti zajedno sa tvojim programom za korisničke testove na sistemu. Format ulaza je isti kao za drugi grader. Primijeti da za korisničke testne primjere ne postoji ugrađena opcija za automatsko generisanje obilazaka.