

Reconstruct

Vremensko ograničenje: 4 sekunde

Memorijsko ograničenje: 1024 MiB

Ovo je interaktivni zadatak.

Bissy se sprema sudjelovati u igri orijentacije *Final destination* koju organiziraju volonteri, u kojoj će se delegacije EJOI-ja natjecati u posjećivanju različitih lokacija diljem grada. No Bissy više zanima *kako* je igra osmišljena nego sama pobjeda.

Na karti se nalazi N lokacija i postoji N ekipa. Svaka ekipa mora posjetiti sve lokacije, a početne lokacije ekipa su međusobno različite — ekipa i kreće s lokacije i . Svaka ekipa dobiva vlastitu rutu.

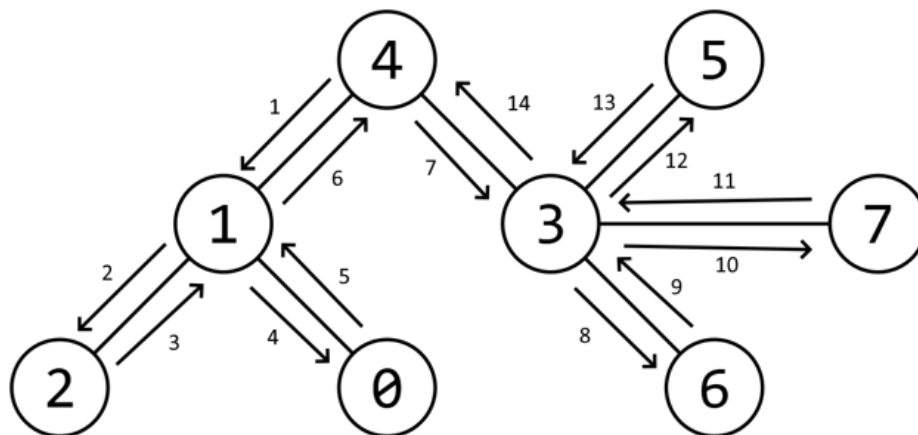
Bissy također ima neke povlaštene informacije — rute se temelje na skrivenoj strukturi brzih linija javnog prijevoza. Preciznije, žiri natjecanja odabrao je $N - 1$ linija koje povezuje parove lokacija, tako da je svaka lokacija dostupna iz bilo koje druge pomoću tih linija. Takva se struktura naziva i stablo. Zatim je žiri za svaku početnu lokaciju i generirao **proizvoljan** *DFS obilazak* (definiran u nastavku) i taj obilazak dodijelio kao rutu ekipi i .

Bissy želi otkriti stablo javnog prijevoza koje je žiri odabrao. Ona postavlja samo brza pitanja sljedeće vrste:

- Koje je j -to odredište i -te ekipe?

Pomozite Bissy tako da napišete program `find_tree` koji pronalazi skriveno stablo.

DFS obilazak stabla je redoslijed kojim se njegovi vrhovi prvi put posjećuju postupkom pretraživanja u dubinu (depth-first search). Takav postupak započinje u zadanom vrhu v i rekurzivno odlazi u jednog od njegovih neposjećenih susjeda sve dok takav susjed postoji — zatim se vraća u prethodno posjećeni vrh i nastavlja odande. Primjer:



Ovdje je $v = 4$, a strelice uz bridove odgovaraju koracima pretraživanja u dubinu. Generirani obilazak je $[4, 1, 2, 0, 3, 6, 7, 5]$. Primijetite da je redoslijed kojim posjećujemo susjede bitan; drugi mogući obilazak bio bi $[4, 3, 5, 7, 6, 1, 0, 2]$.

Struktura stabla fiksna je prije nego što vaš program počne s radom i ne mijenja se kao odgovor na vaša pitanja.

Detalji implementacije

Morate implementirati sljedeću funkciju:

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

- ovdje je N broj lokacija;
- povratna vrijednost je popis bridova stabla — redoslijed bridova i njihovih krajnjih točaka nije bitan.

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

Za interakciju sa žirijem možete pozivati sljedeću funkciju:

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

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

Ograničenja

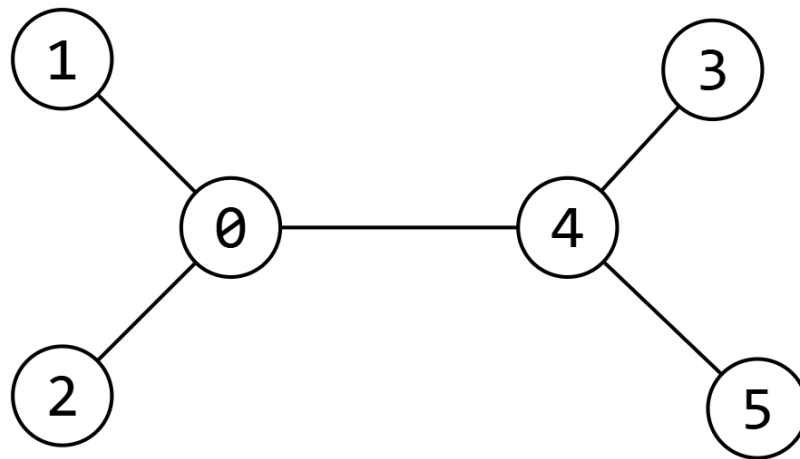
- $2 \leq N \leq 2^{16} + 1$

- T ovisi o ograničenjima temeljenima na najvećem N (označenom s N_{max}) među pozivima funkcije u jednom testu:
 - 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 $1 \leq T \leq 3$

Sustavni ocjenjivač može koristiti do 280 MB, što se uračunava u memoriju vašeg rješenja.

Primjer

Pretpostavimo da je skrivena struktura brzih linija javnog prijevoza sljedeća:



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

Program natjecatelja	Program ocjenjivača
	<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>	

Primijetite da informacije dobivene upitima nisu dovoljne za jednoznačno određivanje stabla, no ovo je zapravo točan odgovor za test u podzadatku 0.

Podzadaci

Podzadatak	Bodovi	N	Dodatna ograničenja
0	0	-	Primjer.
1	11	≤ 9	-
2	6	≤ 100	Svaki je vrh povezan s najviše 2 druga vrha.
3	13	≤ 100	Svaki obilazak generiran je pretraživanjem u dubinu koje u svakom koraku prednost daje udaljavanju od vrha 1. Ako postoji više mogućnosti za udaljavanje od vrha 1, jedna se od njih bira proizvoljno.
4	11	≤ 100	Svaki vrh osim vrha 1 ima manje od 3 susjeda.
5	10	≤ 100	-
6	31	$\leq 2^{10} + 1$	-
7	18	$\leq 2^{16} + 1$	-

Bodovanje

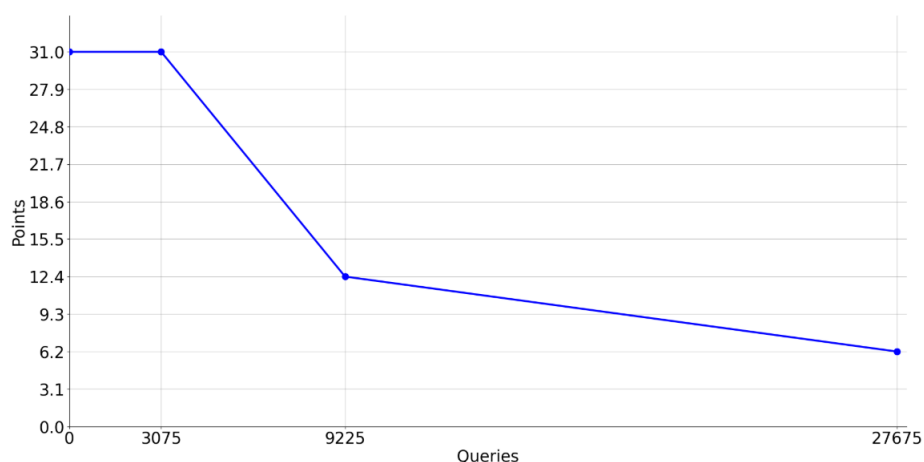
Za svaki podzadatak zaključno s podzadatkom 5, dobivate sve bodove ako uspješno pronađete stablo unutar vremenskog ograničenja. Za podzadatke 6 i 7, udio ukupnih bodova S za zadani test ovisi o **najvećem** broju upita koje postavite na jednom od njegovih podtestova, Q_{max} . Bodovanje za te podzadatke 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 $S = 0.2$;

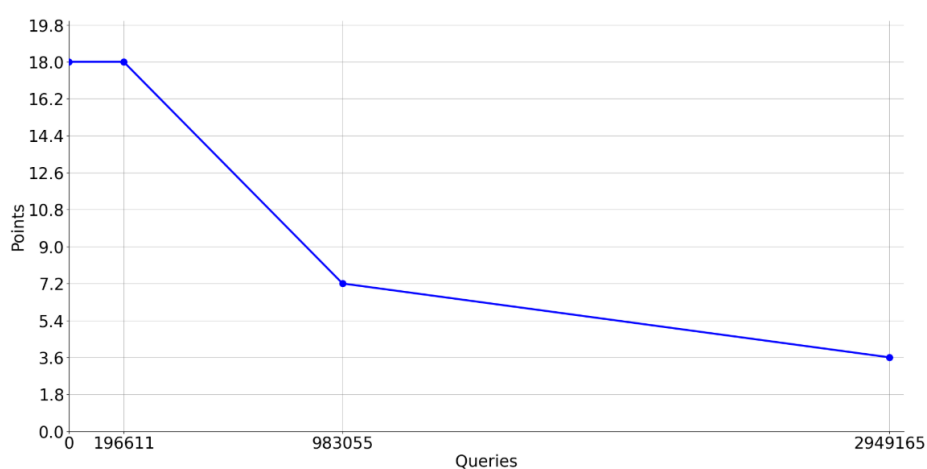
gdje su $L_1 = 3 \times (2^{10} + 1) = 3075$ i $L_2 = 9 \times (2^{10} + 1) = 9225$ za podzadatak 6, a $L_1 = 3 \times (2^{16} + 1) = 196611$ i $L_2 = 15 \times (2^{16} + 1) = 983055$ za podzadatak 7. Udio bodova koji dobivate za cijeli podzadatak jednak je najmanjoj vrijednosti S među svim testovima.

Grafovi ispod prikazuju funkciju bodovanja za posljednja dva podzadatka:

Podzadatak 6



Podzadatak 7



Primjer programa ocjenjivača

Ponuđena su dva primjera programa ocjenjivača.

Za lokalno testiranje: `Lgrader.cpp` se može kompajlirati zajedno s vašim programom kako bi poslužio kao alat za testiranje. Program učitava broj testnih primjera T , a za svaki od njih očekuje broj lokacija N , nakon čega slijedi $N - 1$ redaka, svaki s parom cijelih brojeva koji opisuje izravnu liniju. Nakon toga slijedi N redaka s po N cijelih brojeva – DFS obilasci. Primijetite da obilazak i mora započinjati u vrhu i . Ocjenjivaču možete prepustiti generiranje obilazaka postavljanjem konstante `AUTO_GENERATE` na `true`. Kao izlaz, ocjenjivač ili prijavljuje poruku o pogrešci ako je rezultat netočan, ili ispisuje broj upita postavljenih za svaki testni primjer te ukupni maksimum. Također imajte na umu da ocjenjivač ne radi za dovoljno velike vrijednosti N (točnije, za ograničenja podzadatka 7) — za taj slučaj slobodno se poslužite vlastitom domišljatošću.

Za korisničke testove na sustavu: `stub.cpp` se može koristiti zajedno s vašim programom za korisničke testove na sustavu. Format ulaza jednak je onome za drugi ocjenjivač. Imajte na umu da za korisničke testove ne postoji ugrađena opcija za automatsko generiranje obilazaka.