

Rekonstrukcija

Laiko limitas: 4 sekundės

Atminties limitas: 1024 MiB

Tai – interaktyvi užduotis.

Bissy netrukus dalyvaus savanorių organizuojamame orientavimosi žaidime *Galutinė stotelė*, kuriame EJOI delegacijos varžysis lankydamos įvairias vietas mieste. Bet Bissy labiau domisi, *kaip* žaidimas buvo sukurtas, nei jo laimėjimu.

Žemėlapyje yra N vietų, o varžybose dalyvaus N komandų. Kiekviena komanda startuoja skirtingoje vietoje (komanda i pradeda iš vietos i), bet turi aplankyti visas vietas. Kiekvienai komandai duodamas savas maršrutas.

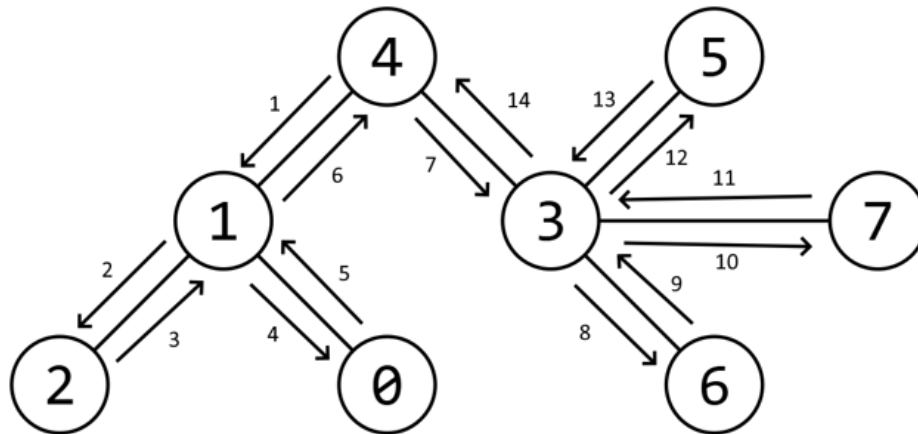
Bissy taip pat turi šiek tiek vidinės informacijos — maršrutai yra pagrįsti greito viešojo transporto linijų struktūra. Tiksliau, varžybų komisija pasirinko $N - 1$ linijų, kiekviena jungianti porą vietų taip, kad kiekvieną vietą galima pasiekti iš bet kurios kitos naudojant šias linijas. Tokia struktūra dar vadinama medžiu. Tada kiekvienai pradinei vietai i komisija sugeneravo **bet kokį DFS pasivaikščiojimą** (apibrėžtą žemiau) ir priskyrė šį pasivaikščiojimą kaip maršrutą komandai i .

Bissy nori išsiaiškinti viešojo transporto medį, kurį komisija pasirinko. Ji užduoda tik tokio pobūdžio klausimus:

- Kokia yra i -osios komandos j -oji kelionės vieta?

Padėkite Bissy parašydami programą `find_tree`, kuri suranda paslėptą viešojo transporto maršrutų medį.

Medžio *DFS pasivaikščiojimas* yra tvarka, kuria jo viršūnės pirmą kartą aplankomos vykdant paieškos į gylį procedūrą. Tokia procedūra prasideda nuo tam tikros viršūnės v ir rekursyviai eina į vieną iš jos neaplankytų kaimynių, kol tokios kaimynės nebelieka, tada ji grįžta į anksčiau aplankytą viršūnę ir tęsia iš ten. Pavyzdys:



Čia $v = 4$ ir rodyklės atitinka paieškos į gylį (DFS) žingsnius. Sugeneruotas pasivaikščiojimas yra $[4, 1, 2, 0, 3, 6, 7, 5]$. Atkreipkite dėmesį, kad tvarka, kuria aplankome kaimynes, yra svarbi; kitas pasivaikščiojimas galėtų būti $[4, 3, 5, 7, 6, 1, 0, 2]$.

Medžio struktūra ir visi N DFS maršrutai yra fiksuoti prieš prasidedant jūsų programai ir nesikeičia atsakant į jūsų klausimus. DFS maršrutuose viršūnių kaimynių aplankymo tvarka gali skirtis.

Implementacijos detalės

Turite realizuoti šią funkciją:

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

- N – vietų mieste skaičius;
- grąžinama reikšmė yra medžio briaunų sąrašas. Briaunų ir jų jungiamų viršūnių tvarka nesvarbi.

Kiekvienam testui jūsų funkcija (`find_tree`) bus iškviesta ne daugiau T kartų.

Norėdami bendrauti su žaidimo komisija, galite iškviesti šią funkciją:

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

Ji grąžina j -ąją vietą i -ajame pasivaikščiojime. Atkreipkite dėmesį, kad `guess(i, 0)` grąžins i . Galite laikyti, kad funkcija atsako per pastovų laiką $O(1)$ visose dalinėse užduotyse iki 6, ir per logaritminį laiką $O(\log N)$ dalinėje užduotyje 7. Kviečiant funkciją (`guess`) turi galioti $0 \leq i, j \leq N - 1$, nes priešingu atveju jūsų sprendimo vertinimas bus `Output isn't correct: Invalid call`.

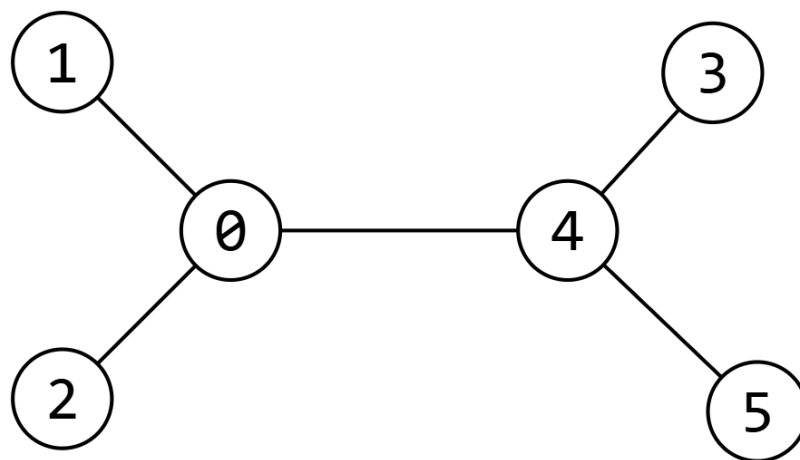
Ribojimai

- $2 \leq N \leq 2^{16} + 1$
- T paskaičiuotas pagal maksimalų N (žymimu N_{max}) iš visų funkcijos iškvietimų tame teste:
 - jei $N_{max} \leq 9$, tai $1 \leq T \leq 100$
 - jei $N_{max} \leq 2^{10} + 1$, tai $1 \leq T \leq 10$
 - jei $N_{max} \leq 2^{16} + 1$, tai $1 \leq T \leq 3$

Vertinimo programa gali naudoti iki 280 MiB atminties, kurie pridedami prie jūsų sprendimo atminties.

Pavyzdys

Tarkime, paslėpta greito viešojo transporto linijų struktūra yra tokia:



Tarkime, kad pradinei vietai 0 pasivaikščiojimas yra $[0, 1, 2, 4, 3, 5]$. Tada yra toks bendravimo pavyzdys:

Dalyvio programa	Komisijos programa
	<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>	

Atkreipkite dėmesį, kad šių užklausų informacijos nepakanka, kad būtų galima vienareikšmiškai nustatyti medį, bet tai yra dalinės užduoties 0 atsakymas.

Dalinės užduotys

Dalinė užduotis	Taškai	N	Papildomi apribojimai
0	0	–	Pavyzdys.
1	11	≤ 9	–
2	6	≤ 100	Kiekviena viršūnė yra sujungta su ne daugiau kaip 2 kitomis.
3	13	≤ 100	Kiekviename paieškos į gylį žingsnyje prioritetizuojamas kelias, kuriuo būtų tolstama nuo viršūnės 0. Jei yra keli būdai judėti tolyn nuo viršūnės 0, pasirenkamas bet kuris.
4	11	≤ 100	Kiekviena viršūnė, išskyrus 0, yra sujungta su ne daugiau kaip 2 kitomis.
5	10	≤ 100	–
6	31	$\leq 2^{10} + 1$	–
7	18	$\leq 2^{16} + 1$	–

Vertinimas

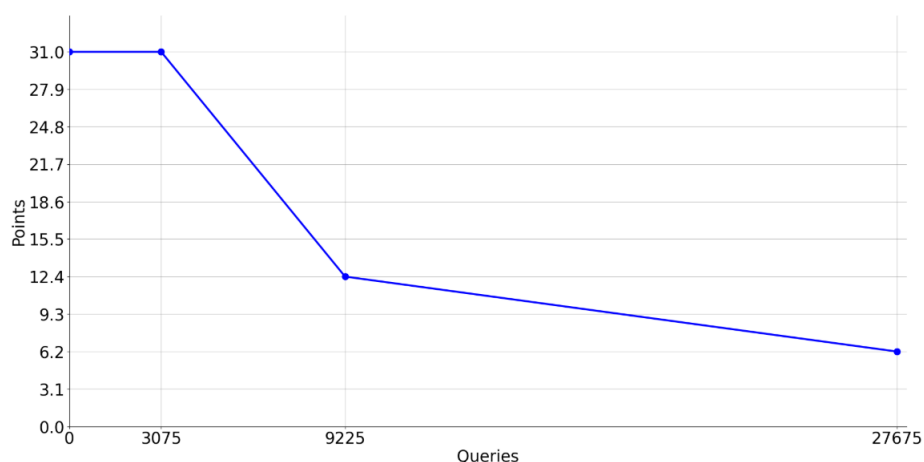
Kiekvienai daliai užduočiai iki dalinės užduoties 5 gaunate visus taškus, jei sėkmingai randate medį per leidžiamą vykdymo laiką. Dalinių užduočių 6 ir 7 gaunamų taškų dalis S už tam tikrą testą priklauso nuo **maksimalaus** užklausų skaičiaus, kurį pateikiate viename jo potestyje, Q_{max} . Šių dalinių užduočių vertinimas yra toks:

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

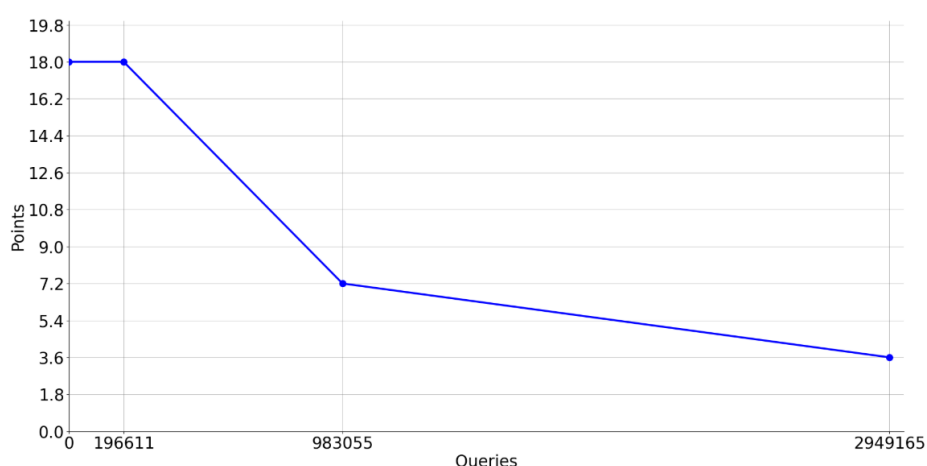
kur $L_1 = 3 \times (2^{10} + 1) = 3075$ ir $L_2 = 9 \times (2^{10} + 1) = 9225$ daliai užduočiai 6, ir $L_1 = 3 \times (2^{16} + 1) = 196611$ ir $L_2 = 15 \times (2^{16} + 1) = 983055$ daliai užduočiai 7. Taškų dalis, kurią gaunate iš dalinės užduoties, yra mažiausia S tarp visų tos dalinės užduoties testų.

Žemiau esančios diagramos rodo vertinimo funkciją paskutinėms dviem dalinės užduotims:

Dalinė užduotis 6



Dalinė užduotis 7



Pavyzdinis vertintojas

Pateikti du pavyzdiniai vertintojai.

Vietiniam testavimui: `Lgrader.cpp` galima sukompiliuoti kartu su jūsų programa. Programa nuskaito testavimo atvejų skaičių T ir kiekvienam iš jų tikisi vietų mieste skaičiaus N , po kurio eina $N - 1$ eilučių, kiekviena su sveikųjų skaičių pora, aprašančia tiesioginę liniją. Po to turi eiti N eilučių po N sveikųjų skaičių – DFS pasivaikščiøjimai. Atkreipkite dėmesį, kad pasivaikščiøjimas i turi prasidėti nuo viršūnės i . Galite leisti vertinimo programai sugeneruoti pasivaikščiøjimus už jus, nustatydami konstantą `AUTO_GENERATE` į `true`. Vertinimo programa išves arba klaidos pranešimą, jei rezultatas neteisingas, arba užklausų skaičių, pateiktų kiekvienam testiniam atvejui, ir bendrą maksimumą. Taip pat atkreipkite dėmesį, kad vertintojas neveikia pakankamai dideliems N (tiksliau potesčio 7 apribojimams), tam atvejui pasinaudokite savo vaizduote.

Sistemos vartotojo testams: `stub.cpp` galima naudoti kartu su jūsų programa vartotojo testams CMS sistemoje. Įvesties formatas yra toks pat kaip ir ankstesnės vertinimo programos. Atkreipkite dėmesį, kad vartotojo testams nėra parinkties pasivaikščiøjimų automatiniam generavimui.