

Yenidən Qurma

Zaman həddi: 4 saniyə

Yaddaş həddi: 1024 MiB

Bu bir interaktiv məsələdir.

Həsən könüllülər tərəfindən təşkil edilmiş *Son təyinat nöqtəsi* oriyentasiya oyununa daxil olmaq üzrədir, burada EJOI nümayəndəlikləri şəhər boyu müxtəlif yerləri ziyarət etmək üçün yarışacaqlar. Lakin Həsən oyunu qazanmaqdan daha çox oyunun necə dizayn edilməsinə maraq göstərir.

Xəritədə N məkan və N komanda var. Hər bir komanda bütün məkanları ziyarət etməlidir və hər birinin unikal başlanğıc məkanı var — i -ci komanda i -ci məkandan başlayır. Hər bir komandaya öz gəzintisi verilir.

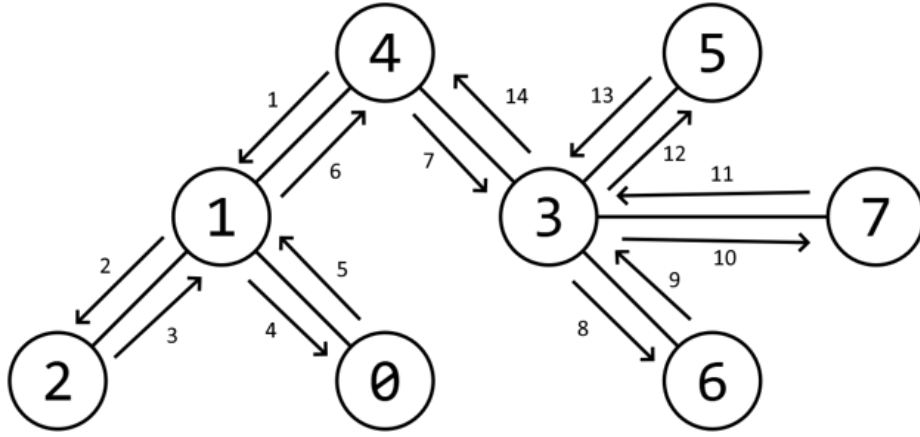
Həsənin həm də bəzi daxili məlumatları var — marşrutlar gizli sürətli ictimai nəqliyyat xətləri strukturu əsasında qurulub. Daha dəqiq desək, müsabiqənin münisflər heyəti $N - 1$ xətt seçib (hər biri bir cüt məkanı birləşdirir) elə ki, bu xətlərdən istifadə edərək hər bir məkandan istənilən digərinə çatmaq mümkündür. Belə bir struktur həm də ağac adlanır. Daha sonra, hər bir i başlanğıc məkanı üçün münisflər heyəti **ixtiyari** DFS gəzintisi (aşağıda tərif verilib) yaradıb və bu gəzintini i -ci komandanın marşrutu kimi təyin edib.

Həsən münisflər heyətinin seçdiyi ictimai nəqliyyat ağacını müəyyən etmək istəyir. O, yalnız aşağıdakı növdə cəld suallar verir:

- i -ci komandanın j -ci təyinat yeri haradır?

Həsən gizli ağacı tapan `find_tree` proqramını yazmaqla kömək edin.

Ağacın DFS gəzintisi — dərinlik üzrə axtarış (depth-first-search) proseduru tərəfindən onun təpələrinin ilk dəfə ziyarət edildiyi sıradır. Belə bir prosedur verilmiş v təpəsindən başlayır və heç bir belə qonşu qalmayana qədər rekursiv olaraq onun ziyarət edilməmiş qonşularından birinə keçir — daha sonra əvvəl ziyarət edilmiş təpəyə qayıdır və oradan davam edir. Nümunə:



Burada $v = 4$ -dür və tillər boyunca oxlar dərinlik üzrə axtarışın addımlarına uyğundur. Yaradılan gəzinti $[4, 1, 2, 0, 3, 6, 7, 5]$ şəklindədir. Qonşuları ziyarət etdiyimiz sıranın əhəmiyyətli olduğuna diqqət yetirin; başqa bir gəzinti $[4, 3, 5, 7, 6, 1, 0, 2]$ ola bilərdi.

Ağac strukturu və N DFS gəzintisi proqramınız başlamazdan əvvəl sabitlənir və suallarınıza cavab olaraq dəyişir. Bu DFS-lər zamanı istifadə olunan təpə qonşularının sırası müxtəlif ola bilər.

İmplementasiya detalları

Aşağıdakı funksiyayı implement etməlisiniz:

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

- burada N məkanların sayıdır;
- qayıdış dəyəri ağacın edge-lərin siyahısıdır — tillərin sırası və onların təpələri əhəmiyyət daşıyır.

Hər test üçün bu funksiya maksimum T dəfə çağrıla bilər.

Jüri ilə qarşılıqlı əlaqə qurmaq üçün aşağıdakı funksiyayı çağırma bilərsiniz:

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

Bu funksiya ağacın i -ci gəzintisindəki j -ci məkanı qaytarır. Nəzərə alın ki, `guess(i, 0)` funksiyası i qaytaracaq. Fərz edə bilərsiniz ki, funksiya 6-ya qədər olan bütün alt məsələlər üçün sabit $O(1)$ zamanında, 7-ci alt məsələ üçün isə loqarifmik $O(\log N)$ zamanında cavab verir. $0 \leq i, j \leq N - 1$ şərti ödənməlidir, əks halda həlliniz `Output isn't correct: Invalid call` hökmünü alacaq.

Məhdudiyyətlər

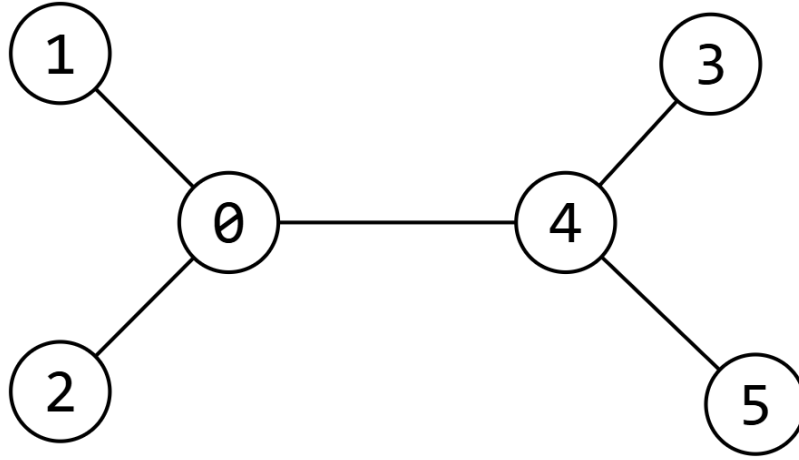
- $2 \leq N \leq 2^{16} + 1$
- T bir testdəki funksiya çağırışları arasında maksimum N -ə (N_{max} ilə işarələnir) əsaslanan məhdudiyyətlərə tabedir:

- əgər $N_{max} \leq 9$, onda $1 \leq T \leq 100$
- əgər $N_{max} \leq 2^{10} + 1$, onda $1 \leq T \leq 10$
- əgər $N_{max} \leq 2^{16} + 1$, onda $1 \leq T \leq 3$

Sistem qiymətləndiricisi (grader) 280 MiB-a qədər yaddaş istifadə edə bilər və bu, həllinizin yaddaş limitinə daxildir.

Nümunə

Fərz edək ki, gizli sürətli ictimai nəqliyyat xətləri strukturu aşağıdakı kimidir: Fərz edək ki, 0 başlanğıc məkanı üçün gəzinti $[0, 1, 2, 4, 3, 5]$ şəklindədir. Onda aşağıdakı qarşılıqlı əlaqə nümunəsidir:



Fərz edin ki, başlanğıc yer 0 üçün yürüyüş $[0, 1, 2, 4, 3, 5]$ -dir. Onda aşağıdakı qarşılıqlı əlaqə nümunəsidir:

İştirakçı proqramı	Jüri proqramı
	<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>	

Nəzərə alın ki, sorğulardan əldə edilən məlumat ağacı unikal şəkildə müəyyən etmək üçün kifayət deyil, lakin bu, əslində 0-cı alt məsələdəki testin cavabıdır.

Alt tapşırıqlar

Alt tapşırıq	Xal	N	Əlavə məhdudiyyətlər
0	0	–	Nümunə.
1	11	≤ 9	–
2	6	≤ 100	Hər bir təpə ən çoxu 2 digər təpəyə birləşib.
3	13	≤ 100	Hər bir gəzinti dərinlik üzrə axtarış tərəfindən yaradılır və hər addımda 0-cı təpədən uzaqlaşmağa üstünlük verir. Əgər 0-cı təpədən uzaqlaşmaq üçün birdən çox seçim varsa, onların birisi ixtiyari olaraq seçilir.
4	11	≤ 100	0-dan başqa hər bir təpənin maksimum 2 dənə qonşusu var.
5	10	≤ 100	–
6	31	$\leq 2^{10} + 1$	–
7	18	$\leq 2^{16} + 1$	–

Qiymətləndirmə

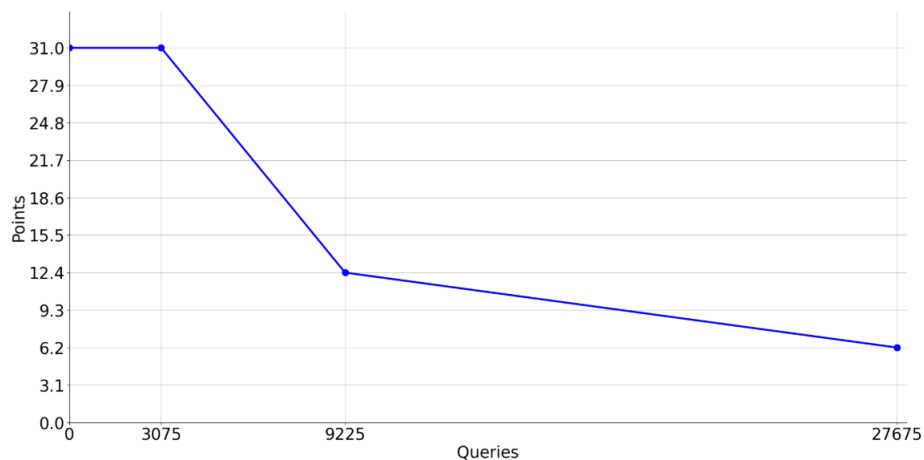
Alt məsələ 5-ə qədər olan hər bir alt məsələ üçün vaxt limiti məhdudiyyətləri daxilində ağacı uğurla tapsanız, tam xal əldə edirsiniz. 6 və 7-ci alt məsələlər üçün verilmiş test üzrə tam S xalının nisbəti onun alt testlərindən birində verdiyiniz maksimum sorğu sayından (Q_{max}) asılıdır. Bu alt məsələlər üçün qiymətləndirmə aşağıdakı kimidir:

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

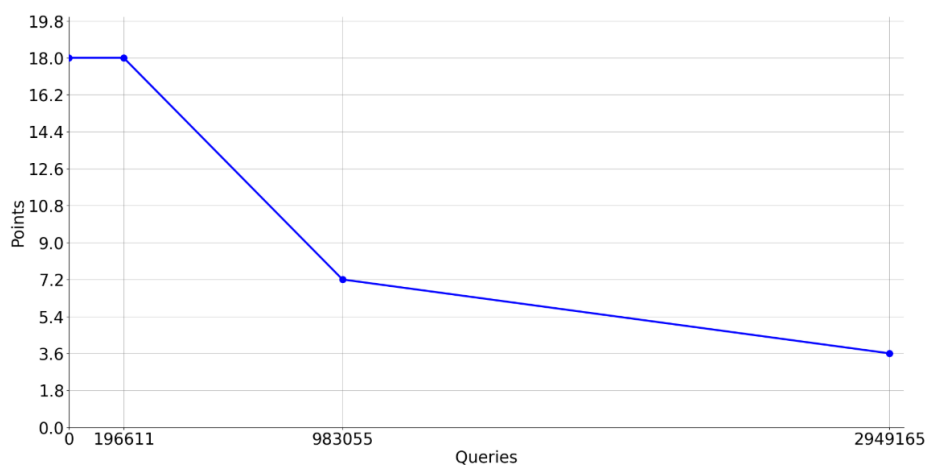
burada 6-cı alt məsələ üçün $L_1 = 3 \times (2^{10} + 1) = 3075$ və $L_2 = 9 \times (2^{10} + 1) = 9225$, 7-ci alt məsələ üçün isə $L_1 = 3 \times (2^{16} + 1) = 196611$ və $L_2 = 15 \times (2^{16} + 1) = 983055$ -dir. Bütün alt məsələdən aldığınız xalın nisbəti bütün testlər arasında minimum S dəyəridir.

Aşağıdakı qrafiklər son iki alt məsələ üçün qiymətləndirmə funksiyasını göstərir:

Alt tapşırıq 6



Alt tapşırıq 7



Nümunə grader

Təqdim olunan iki nümunə grader var.

Lokal sınaq üçün: Lgrader.cpp sınaq vasitəsi təmin etmək üçün proqramınızla birlikdə kompilyasiya oluna bilər. Proqram T test hallarının sayını oxuyur və hər biri üçün məkanların sayı N -i, ardından birbaşa xətti təsvir edən bir cüt tam ədəd ehtiva edən $N - 1$ sətiri gözləyir. Bundan sonra N sayda tam ədəddən ibarət N sətir gəlir – DFS gəzintiləri. Nəzərə alın ki, i gəzintisi i təpəsindən başlamalıdır. Qiymətləndiricinin sizin üçün gəzintilər yaratmasına imkan vermək üçün AUTO_GENERATE sabitini true olaraq təyin edə bilərsiniz. Çıxış olaraq qiymətləndirici nəticə yanıqdırsa bir xəbərdarlıq mesajı və ya hər bir test halı üçün verilmiş sorğuların sayını və ümumi maksimumu bildirir. Həmçinin nəzərə alın ki, qiymətləndirici kifayət qədər böyük N üçün işləmir (daha dəqiq desək, alt məsələ 7 məhdudiyyətləri), bu hal üçün təxəyyülünüzdən sərbəst istifadə edə bilərsiniz.

Sistem istifadəçi testləri üçün: stub.cpp sistemdəki istifadəçi testləri üçün proqramınızla birlikdə istifadə edilə bilər. Daxiletmə formatı digər qiymətləndirici ilə eynidir. Nəzərə alın ki, istifadəçi

testləri üçün avtomatik gəzinti yaratmaq üçün daxili seçimlər yoxdur.