

Task Navigiranje

 3 sek.  512 MB

Postoji **povezani neusmjereni jednostavni kaktusni graf**¹ s $N \leq 1000$ čvorova i M bridova. Njegovi čvorovi imaju boje (označene nenegativnim cijelim brojevima od 0 do 1499). U početku svi čvorovi imaju boju 0. **deterministički robot bez memorije**² istražuje graf krećući se od čvora do čvora. Mora posjetiti sve čvorove barem jednom, a zatim završiti.

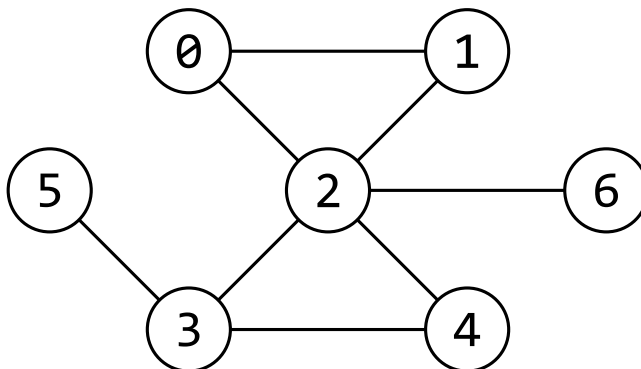
Robot počinje na nekom čvoru, koji može biti bilo koji od čvorova u grafu. U svakom koraku vidi boju svog trenutnog čvora i boje svih susjednih čvorova **u nekom redoslijedu fiksnom za trenutni čvor** (tj. ponovni posjet čvoru dat će robotu isti niz susjednih čvorova, čak i ako su im boje drugačije nego prije). Robot izvodi jednu od sljedeće dvije radnje:

1. Odlučuje završiti.
2. Odabire novu (ili eventualno istu) boju za trenutni čvor i susjedni čvor na koji se treba pomaknuti. Susjedni čvor identificira se indeksom od 0 do $D - 1$, gdje je D broj susjednih čvorova.

U drugom slučaju, trenutni čvor se prebojava (ili eventualno ostaje iste boje) i robot se pomiče na odabrani susjedni čvor. To se ponavlja sve dok robot ne završi ili dok ne dosegne ograničenje iteracije. Robot pobjeđuje ako posjeti sve čvorove, a zatim završi unutar ograničenja iteracije od $L = 3000$ koraka (inače gubi).

Trebali biste osmisliti strategiju za robota koja može riješiti problem na bilo kojem takvom grafu kaktusa. Osim toga, trebali biste pokušati smanjiti broj različitih boja koje vaše rješenje koristi. Ovdje se boja 0 uvijek računa kao korištena.

¹Povezani neusmjereni jednostavni graf kaktusa je povezani neusmjereni jednostavni graf (svaki čvor je dostupan iz svakog drugog čvora; rubovi su dvosmjerni; nema vlastitih petlji ili višestrukih rubova) u kojem svaki rub pripada najviše jednom jednostavnom ciklusu (jednostavan ciklus je ciklus koji sadrži svaki čvor najviše jednom). Donja slika je primjer.



²Robot je deterministički i bez memorije ako njegova radnja ovisi samo o trenutnim ulazima (tj. ne pohranjuje podatke iz koraka u korak) i uvijek bira istu radnju kada su mu



zadani isti ulazi.



Detalji implementacije

Strategija robota trebala bi se implementirati kao sljedeća funkcija:

```
std::pair<int, int> navigate(int currColor, std::vector<int> adjColors)
```

Kao parametre prima boju trenutnog čvora i boje svih susjednih čvorova (po redu). Mora vratiti par čiji je prvi element nova boja za trenutni čvor, a drugi element indeks susjednosti čvora na koji se robot treba pomaknuti. Ako se robot umjesto toga treba prekinuti, funkcija bi trebala vratiti par $(-1, -1)$.

Ova će se funkcija više puta pozivati kako bi se odabrale akcije robota. Budući da je deterministička, ako je `navigate` već pozvana s nekim parametrima, nikada se više neće pozivati s istim tim parametrima; umjesto toga, ponovno će se koristiti njezina prethodna povratna vrijednost. Osim toga, svaki test može sadržavati $T \leq 5$ podtestova (različitih grafova i/ili početnih pozicija) i mogu se izvoditi istovremeno (tj. vaš program može dobiti naizmjenične pozive o različitim podtestovima). Konačno, pozivi funkcije `navigate` mogu se dogoditi u **odvojenim izvršavanjima** vašeg programa (ali se ponekad mogu dogoditi i u istom izvršavanju). Ukupan broj izvršavanja vašeg programa je $P = 100$. Zbog svega toga, vaš program ne bi trebao pokušavati prenositi informacije između različitih poziva.



Ograničenja

- $3 \leq N \leq 1000$
- $0 \leq \text{Color} < 1500$
- $L = 3000$
- $T \leq 5$
- $P = 100$



Bodovanje

Udio S bodova za podzadatak koji dobijete ovisi o C - maksimalnom broju različitih boja koje vaše rješenje koristi (uključujući boju 0) na bilo kojem testu u tom podzadatku ili bilo kojem drugom obaveznom podzadatku:

- Ako vaše rješenje ne prođe na bilo kojem podtestu, tada je $S = 0$.
- Ako je $C \geq 4$, tada je $S = 1, 0$.
- Ako je $4 < C \leq 8$, tada je $S = 1, 0 - 0, 6 \frac{C-4}{4}$.
- Ako je $8 < C \leq 21$, tada je $S = 0, 4 \frac{8}{C}$.
- Ako je $C > 21$, tada je $S = 0, 15$.



Podzadaci

Podzadatak	Bodovi	Potrebni podzadaci	N	Dodatna ograničenja
0	0	—	≤ 300	Primjer.
1	6	—	≤ 300	Graf je ciklus. ¹
2	7	—	≤ 300	Graf je zvijezda. ²
3	9	—	≤ 300	Graf je put. ³
4	16	2 – 3	≤ 300	Graf je stablo. ⁴
5	27	—	≤ 300	Svi čvorovi imaju najviše 3 susjednih čvorova, a čvor na kojem robot počinje ima 1 susjednog čvora.
6	28	0 – 5	≤ 300	—
7	7	0 – 6	—	—

¹Ciklusni graf ima bridove: $(i, (i + 1) \bmod N)$ za $0 \leq i < N$.

²Zvijezdani graf ima bridove: $(0, i)$ za $1 \leq i < N$.

³Putanja grafa imaju bridove: $(i, i + 1)$ za $0 \leq i < N - 1$.

⁴Stablo je graf bez ciklusa.



Primjer

Razmotrimo primjer grafa sa slike u izrazu, koji ima $N = 7$, $M = 8$ i bridove $(0, 1)$, $(1, 2)$, $(2, 0)$, $(2, 3)$, $(3, 4)$, $(4, 2)$, $(3, 5)$ i $(2, 6)$. Osim toga, budući da su redoslijedi elemenata u popisima susjednosti čvorova relevantni, dajemo ih u ovoj tablici:

Čvor	Susjedni čvorovi
0	2, 1
1	2, 0
2	0, 3, 4, 6, 1
3	4, 5, 2
4	2, 3
5	3
6	2

Pretpostavimo da robot počinje na čvoru 5. Tada je sljedeći jedan mogući (neuspješni) slijed interakcija:

#	Boje	Čvor	Poziv funkcije navigate	Povratna vrijednost
1	0, 0, 0, 0, 0, 0, 0	5	<code>navigate(0, {0})</code>	<code>{1, 0}</code>
2	0, 0, 0, 0, 0, 1, 0	3	<code>navigate(0, {0, 1, 0})</code>	<code>{4, 2}</code>
3	0, 0, 0, 4, 0, 1, 0	2	<code>navigate(0, {0, 4, 0, 0, 0})</code>	<code>{0, 3}</code>
4	0, 0, 0, 4, 0, 1, 0	6	¹ <code>navigate(0, {0})</code>	<code>{1, 0}</code>
5	0, 0, 0, 4, 0, 1, 1	2	<code>navigate(0, {0, 4, 0, 1, 0})</code>	<code>{8, 0}</code>
6	0, 0, 8, 4, 0, 1, 1	0	<code>navigate(0, {8, 0})</code>	<code>{3, 0}</code>
7	3, 0, 8, 4, 0, 1, 1	2	<code>navigate(8, {3, 4, 0, 1, 0})</code>	<code>{2, 2}</code>
8	3, 0, 2, 4, 0, 1, 1	4	<code>navigate(0, {2, 4})</code>	<code>{1, 1}</code>
9	3, 0, 2, 4, 1, 1, 1	3	<code>navigate(4, {1, 1, 2})</code>	<code>{-1, -1}</code>

Ovdje je robot koristio ukupno 6 različitih boja: 0, 1, 2, 3, 4 i 8 (imajte na umu da bi se 0 računalo kao korišteno čak i da robot nikada nije vratio boju 0, budući da svi čvorovi počinju u boji 0). Robot je radio 9 iteracija prije završetka. Međutim, nije uspio jer je završio bez posjeta čvoru 1.

¹Imajte na umu da se poziv funkcije `navigate` u iteraciji 4 zapravo ne bi dogodio. To je zato što je ekvivalentno pozivu u iteraciji 1, pa bi ocjenjivač jednostavno ponovno upotrijebio povratnu vrijednost vaše funkcije iz tog poziva. Međutim, ovo se i dalje računa kao iteracija robota.



Primjer ocjenjivanja

Primjer ocjenjivanja ne pokreće višestruka izvršavanja vašeg programa, tako da će svi pozivi na `navigate` biti u istom izvršavanju vašeg programa.

Format ulaza je sljedeći: Prvo se čita T (broj podtestova). Zatim za svaki podtest:

- line 1: dva cijela broja - N i M ;
- line $2 + i$ (za $0 \leq i < M$): dva cijela broja - A_i i B_i , koji su dva čvora koja brid i spaja ($0 \leq A_i, B_i < N$).

Primjer ocjenjivanja će zatim ispisati broj različitih boja koje je vaše rješenje koristilo i broj iteracija koje su mu bile potrebne prije nego što je završilo. Alternativno, ispisati će poruku o pogrešci ako vaše rješenje nije uspjelo.

Prema zadanim postavkama, ocjenjivač uzorka ispisuje detaljne informacije o tome što



robot vidi i radi u svakoj iteraciji. To možete onemogućiti promjenom vrijednosti `DEBUG` iz `true` u `false`.