Maksas ceļi

Japānā pilsētas ir savienotas ar ceļu tīklu, kas sastāv no N pilsētam un M ceļiem. Katrs ceļš savieno divas dažādas pilsētas. Nekādi divi ceļi nesavieno vienu un to pašu pilsētu pāri. Pilsētas ir numurētas no 0 līdz $N-1$, un ceļi ir numurēti no 0 līdz $M-1$. Jūs varat braukt pa ceļiem abos virzienos. Jūs varat aizceļot no jebkuras pilsētas uz jebkuru citu pilsētu izmantojot ceļus.

Lai brauktu pa kādu ceļu, tiek iekasēta *maksa*, kas ir atkarīga no **satiksmes** uz šī ceļa. Satiksme ir vai nu **ātra** vai **lēna**. Kad satiksme ir ātra, maksa par ceļa izmantošanu ir A jēnas (Japānas valūta). Kad satiksme ir lēna, maksa ir B jēnas. Ir garantēts ka $A < B$. Dažādiem ceļiem satiksmes ātrums var atšķirties. Ievērojiet ka A un B vērtības jums ir zināmas.

Jums ir ierīce, kas, dotai satiksmei uz visiem ceļiem, aprēķina mazāko kopējo maksu, kas jāsamaksā lai veiktu ceļojumu starp pilsētām S un T ($S \neq T$), ņemot vēra zināmos satiksmes ātrumus.

Diemžēl, ierīce ir tikai prototips. S un T vērtības ir nofiksētas (t.i. iekodētas ierīcē) bet nav jums zināmas. Jūs gribat noskaidrot S un T. Lai šo izdarītu, jūs plānojat norādīt ierīcei vairākas satiksmes ātrumu kopas, un, izmantot ierīces atgrieztās maksas vērtības, izsecināt S un T vērtības. Tā kā norādīt satiksmes ātrumus ierīcei ir dārgi, jūs vēlaties izmantot ierīci pēc iespējas mazāku reižu skaitu.

Implementācijas detaļas

Jums ir jāimplementē šāda procedūra:

```
find_pair(int N, int[] U, int[] V, int A, int B)
```

- `N`: pilsētu skaits.
- `U` un `V`: masīvi garumā M, kur M ir ceļu skaits. Katram i ($0 \leq i \leq M-1$), ceļš ar numuru i savieno pilsētas ar numuriem `U[i]` un `V[i]`.
- `A`: maksa par ceļu kad satiksme ir ātra.
- `B`: maksa par ceļu kad satiksme ir lēna.
- Šī procedūra katram testam tiek izsaukta tieši vienreiz.
- Pievērsiet uzmanību, ka vērtība M ir masīva garums, un var tikt iegūta kā norādīts implementācijas norādījumos.

Procedūra `find_pair` var izsaukt šādu funkciju:

```
int64 ask(int[] w)
```

- Masīva `w` garumam ir jābūt M. Masīvs `w` apraksta satiksmes ātrumus.
- Katram i ($0 \leq i \leq M-1$), `w[i]` uzdod satiksmes ātrumu uz ceļa ar numuru i. `w[i]` vērtībai ir jābūt vai nu 0, vai nu 1.
  - `w[i] = 0` nozīmē ka satiksme uz ceļa ar numuru i ir ātra.
  - `w[i] = 1` nozīmē ka satiksme uz ceļa ar numuru i ir lēna.
- Šī funkcija atgriež mazāko kopēju maksu lai veiktu ceļojumu starp pilsētām ar numuriem S un T, ņemot vēra masīvā `w` definētos satiksmes ātrumus.
- Šī funkcija katram testam var tikt izsaukta ne vairāk kā 100 reizes.

`find_pair` ir jāizsauc šāda procedūra, lai iesniegtu savu atbildi:

```
answer(int s, int t)
```

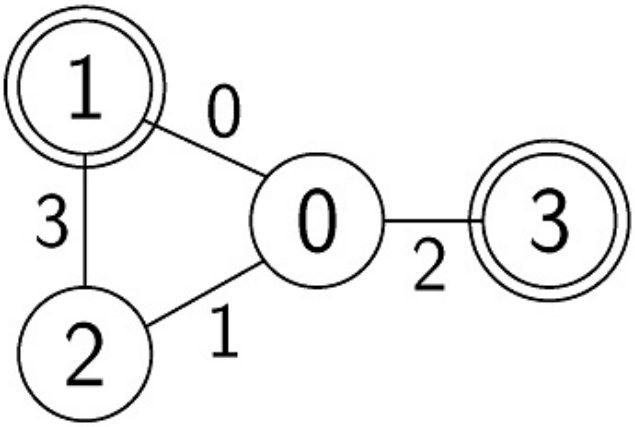
- `s` un `t` ir jābūt vērtību S un T pārim (secība pārī nav svarīga).
- Šī procedūra ir jāizsauc tieši vienreiz.

Ja kādi no augstākminētajiem nosacījumiem neizpildās, jūsu programa tiks vērtēta ar **`Wrong Answer`**. Pretējā gadījumā, jūsu programa tiks vērtēta ar **`Accepted`** un jūsu punktu skaits tiks aprēķināts atkarībā no funkcijas `ask` izsaukumu daudzuma (skatīt *Apakšuzdevumi*).

Piemērs

Pieņemsim, ka $N = 4$, $M = 4$, $U = [0, 0, 0, 1]$, $V = [1, 2, 3, 2]$, $A = 1$, $B = 3$, $S = 1$, un $T = 3$.

Vērtētajs izsauc `find_pair(4, [0, 0, 0, 1], [1, 2, 3, 2], 1, 3)`.



Zīmējumā ceļam ar numuru i atbilst šķautne ar numuru i. Daži iespējami funkcijas `ask`

izsaukumi un to atgrieztās vērtības ir dotas tabulā:

Izsaukums	Atgrieztā vērtība
`ask([0, 0, 0, 0])`	2
`ask([0, 1, 1, 0])`	4
`ask([1, 0, 1, 0])`	5
`ask([1, 1, 1, 1])`	6

Funkcijas `ask([0, 0, 0, 0])` izsaukumam, uz katra ceļa satiksme ir ātra un maksa par katru no tiem ir 1. Lētākais ceļojums no $S = 1$ līdz $T = 3$ ir $1 \to 0 \to 3$. Maksa par šo ceļojumu ir 2. Tātad, funkcija atgriež 2.

Lai iesniegtu pareizu atbildi, procedūrai `find_pair` ir jāizsauc `answer(1, 3)` vai `answer(3, 1)`.

Fails `sample-01-in.txt` pievienotajā pakotnes arhīvā atbilst tikko aprakstītajam piemēram. Arhīvā ir pieejami arī citu piemēru ievaddati.

Ierobežojumi

- $2 \leq N \leq 90\,000$
- $1 \leq M \leq 130\,000$
- $1 \leq A < B \leq 1\,000\,000\,000$
- Katram $0 \leq i \leq M - 1$
  - $0 \leq U[i] \leq N - 1$
  - $0 \leq V[i] \leq N - 1$
  - $U[i] \neq V[i]$
- $(U[i], V[i]) \neq (U[j], V[j])$ un $(U[i], V[i]) \neq (V[j], U[j])$ $(0 \leq i < j \leq M - 1)$
- Jūs varat aizceļot no jebkuras pilsētas uz jebkuru citu pilsētu.
- $0 \leq S \leq N - 1$
- $0 \leq T \leq N - 1$
- $S \neq T$

Šajā uzdevumā vērtētājs NAV adaptīvs. Tas nozīmē ka virknes S un T tiek fiksētas vērtētāja darbības sākumā un nav atkarīgas no jūsu risinājumā izdarītajiem vaicājumiem.

Apakšuzdevumi

1. (5 punkti) viens no S vai T ir 0, $N \leq 100$, $M = N - 1$
2. (7 punkti) viens no S vai T ir 0, $M = N - 1$
3. (6 punkti) $M = N - 1$, $U[i] = i$, $V[i] = i + 1$ $(0 \leq i \leq M - 1)$

4. (33 punkti) $M = N - 1$
5. (18 punkti) $A = 1$, $B = 2$
6. (31 punkts) Bez papildu ierobežojumiem

Pieņemsim kā jūsu programma ir novērtēta ar **`Accepted`**, un izsauc funkciju `ask` X reizes. Tad jūsu punktu skaits P par šo testu, atkarībā no apakšuzdevuma numura, tiks izrēķināts šādi:

- Apakšuzdevums 1. $P = 5$.
- Apakšuzdevums 2. Ja $X \leq 60$, $P = 7$. Citādi $P = 0$.
- Apakšuzdevums 3. Ja $X \leq 60$, $P = 6$. Citādi $P = 0$.
- Apakšuzdevums 4. Ja $X \leq 60$, $P = 33$. Citādi $P = 0$.
- Apakšuzdevums 5. Ja $X \leq 52$, $P = 18$. Citādi $P = 0$.
- Apakšuzdevums 6.
  - Ja $X \leq 50$, $P = 31$.
  - Ja $51 \leq X \leq 52$, $P = 21$.
  - Ja $53 \leq X$, $P = 0$.

Ievērojiet, ka punktu skaits katrā apakšuzdevumā ir mazākais punktu skaits, kāds iegūts šī apakšuzdevuma testos.

Paraugvērtētājs

Paraugvērtētājs ielasa ievaddatus šādā formātā:

- 1. rinda: N M A B S T
- $2 + i$-tā rinda ($0 \leq i \leq M - 1$): $U[i]$ $V[i]$

Ja jūsu programma ir novērtēta ar **`Accepted`**, tad paraugvērtētājs drukā `Accepted: q`, kur `q` ir funkcijas `ask` izsaukumu skaits.

Ja jūsu programma ir novērtēta ar **`Wrong Answer`**, tad paraugvērtētājs drukā `Wrong Answer: MSG`, kur `MSG` ir viens no:

- `answered not exactly once`: Procedūra `answer` tikai izsaukta vairāk vai mazāk kā vienreiz.
- `w is invalid`: Masīva `w` garums nav M vai `w[i]` nav ne 0, ne 1 kādai no i($0 \leq i \leq M - 1$) vērtībām.
- `more than 100 calls to ask`: Funkcija `ask` tika izsaukta vairāk nekā 100 reizes.
- `{s, t} is wrong`: Procedūra `answer` tika izsaukta ar nepareizu pāri `s` un `t`.