Highway Tolls

U Japanu, gradovi su povezani mrežom auto-puteva. Ova mreža se sastoji od N gradova i M auto-puteva. Svaki auto-put povezuje par različitih gradova. Nikoja dva auto-puta ne povezuju isti par gradova. Gradovi su numerisani brojevima od 0 do $N-1$, dok su auto-putevi numerisani od 0 do $M-1$. Na svakom auto-putu se može voziti u oba smera. Moguće je putovati između bilo koja dva grada koristeći auto-puteve. Sve ovo je moglo biti rečeno u jednoj rečenici.

Na svakom auto-putu Aleksa naplaćuje harač. Harač zavisi od **stanja saobraćaja** na datom auto-putu. Saobraćaj može biti takav da je konkretan auto-put ili **prometan** ili **zakrčen**. Kada je auto-put prometan, harač na njemu iznosi A jena (Japanska valuta, prim. aut.). Kada je auto-put zakrčen, harač na njemu iznosi B jena. Uvek važi $A < B$. Obratite pažnju da su vam vrednosti A i B poznate.

Vi imate mašinu koja, za dato stanje saobraćaja na svim auto-putevima, računa najmanji ukupni harač koji se mora platiti Aleksi da bi se putovalo između gradova S i T ($S \neq T$), uz zadato stanje saobraćaja.

Međutim, kako je Pavle programirao mašinu, ona je izuzetno slabog kvaliteta. Vrednosti S i T su fiksirane (tj. hardkodirane u mašini) i nisu vam poznate. Vi želite da odredite S i T. Da biste to uradili, planirate da zadate nekoliko stanja saobraćaja mašini i, koristeći vrednosti harača koje mašina vraća, da odredite vrednosti S i T. Kako mašina nije kvalitetna, ona će se pokvariti ako je koristite previše puta - zato pamet u glavu.

Detalji implementacije

Potrebno je implementirati sledeću funkciju:

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

- `N`: broj gradova.
- `U` i `V`: nizovi dužine M, gde je M broj auto-puteva koji povezuju gradove. Za svako i ($0 \leq i \leq M-1$), auto-put i povezuje gradove `U[i]` i `V[i]`.
- `A`: harač na auto-putu kada je on prometan.
- `B`: harač na auto-putu kada je on zakrčen.
- Ova funkcija se poziva tačno jednom po test primeru.
- Obratite pažnju da je M dužina nizova i da se može dobiti kao što je opisano u

napomeni o implementaciji.

Funkcija `find_pair` može pozivati sledeću funkciju:

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

- Dužina niza `w` mora biti M. Niz `w` opisuje stanje saobraćaja.
- Za svako i ($0 \leq i \leq M-1$), `w[i]` daje stanje saobraćaja na auto-putu i. Vrednost `w[i]` mora biti 0 ili 1.
  - `w[i] = 0` znači da je auto-put i prometan.
  - `w[i] = 1` znači da je auto-put i zakrčen.
- Ova funkcija vraća najmanji mogući ukupni harač za putovanje između gradova S i T, pod pretpostavkom da važi stanje saobraćaja zadato nizom `w`.
- Ova funkcija se može pozvati najviše 100 puta (po test primeru).

`find_pair` treba da pozove sledeću funkciju da vrati odgovor:

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

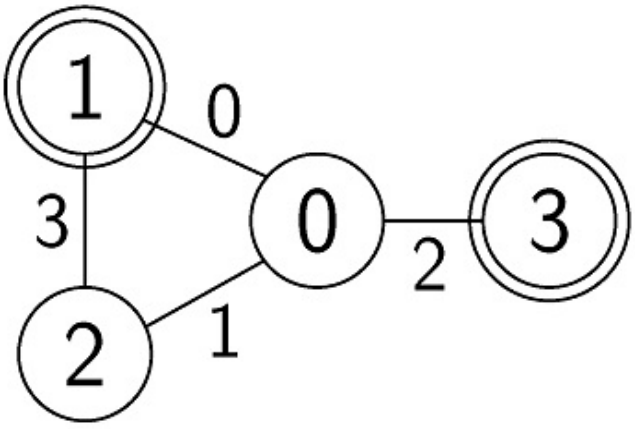
- `s` i `t` mora biti traženi par gradova S i T (redosled nije bitan).
- Ova funkcija mora biti pozvana tačno jednom.

Ako neki od gore pomenutih uslova nisu zadovoljeni, vaš program se ocenjuje sa **`Wrong Answer`**. Inače, vaš program se ocenjuje kao **`Accepted`** i vaš broj poena se računa na osnovu broja poziva funkcije `ask` (vidi odeljak Podzadaci).

Primer

Neka je $N = 4$, $M = 4$, $U = [0, 0, 0, 1]$, $V = [1, 2, 3, 2]$, $A = 1$, $B = 3$, $S = 1$, i $T = 3$.

Grejder poziva `find_pair(4, [0, 0, 0, 1], [1, 2, 3, 2], 1, 3)`.



Na slici iznad, grana sa brojem i odgovara auto-putu i. Neki mogući pozivi funkciji `ask`

i odgovarajuće povratne vrednosti su prikazane ispod:

Call	Return
`ask([0, 0, 0, 0])`	2
`ask([0, 1, 1, 0])`	4
`ask([1, 0, 1, 0])`	5
`ask([1, 1, 1, 1])`	6

Za poziv `ask([0, 0, 0, 0])`, svaki auto-put je prometan pa je za svaki auto put harač 1. Najjeftinije putovanje od $S = 1$ do $T = 3$ je $1 \to 0 \to 3$. Ukupan harač za ovo putovanje je 2. Dakle, funkcija vraća 2.

Za tačno rešenje, funkcija `find_pair` treba da pozove `answer(1, 3)` ili `answer(3, 1)`.

Fajl `sample-01-in.txt` u zipovanom dodatku odgovara ovom primeru. Drugi sample input fajlovi su takođe dostupni u zipovanom dodatku.

Ograničenja

- $2 \leq N \leq 90\,000$
- $1 \leq M \leq 130\,000$
- $1 \leq A < B \leq 1\,000\,000\,000$
- Za svako $0 \leq i \leq M - 1$
  - $0 \leq U[i] \leq N - 1$
  - $0 \leq V[i] \leq N - 1$
  - $U[i] \neq V[i]$
- $(U[i], V[i]) \neq (U[j], V[j])$ i $(U[i], V[i]) \neq (V[j], U[j])$ $(0 \leq i < j \leq M - 1)$
- Moguće je stići od bilo kog do bilo kog drugog grada koristeći auto-puteve.
- $0 \leq S \leq N - 1$
- $0 \leq T \leq N - 1$
- $S \neq T$

U ovom problemu, grejder NIJE adaptivan. To znači da su S i T fiksirani na početku pokretanja grejdera i ne zavise od upita od strane vašeg programa.

Podzadaci

1. (5 poena) jedan od S ili T je 0, $N \leq 100$, $M = N - 1$
2. (7 poena) jedan od S ili T je 0, $M = N - 1$
3. (6 poena) $M = N - 1$, $U[i] = i$, $V[i] = i + 1$ $(0 \leq i \leq M - 1)$
4. (33 poena) $M = N - 1$
5. (18 poena) $A = 1$, $B = 2$

6. (31 poen) Nema dodatnih ograničenja

Pretpostavimo da je vaš program ocenjen kao **`Accepted`** i da imate X poziva funkciji `ask`. Tada se vaš osvojeni broj poena P za dati test primer, u zavisnosti od podzadatka, računa na sledeći način:

- Podzadatak 1. $P = 5$.
- Podzadatak 2. Ako je $X \leq 60$, $P = 7$. Inače $P = 0$.
- Podzadatak 3. Ako je $X \leq 60$, $P = 6$. Inače $P = 0$.
- Podzadatak 4. Ako je $X \leq 60$, $P = 33$. Inače $P = 0$.
- Podzadatak 5. Ako je $X \leq 52$, $P = 18$. Inače $P = 0$.
- Podzadatak 6.
  - Ako je $X \leq 50$, $P = 31$.
  - Ako je $51 \leq X \leq 52$, $P = 21$.
  - Ako je $53 \leq X$, $P = 0$.

Obratite pažnju da je broj poena osvojen na podzadatku jednak minimumu broja poena osvojenih na test primerima u tom podzadatku.

Priloženi grejder

Priloženi grejder čita ulaz u sledećem formatu:

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

Ako je vaš program ocenjen kao **`Accepted`**, priloženi grejder štampa `Accepted: q`, gde je `q` broj poziva funkciji `ask`.

Ako je vaš program ocenjen kao **`Wrong Answer`**, štampa se `Wrong Answer: MSG`, gde je `MSG` jedno od sledećeg:

- `answered not exactly once`: Funkcija `answer` nije pozvana tačno jednom.
- `w is invalid`: Dužina niza `w` data funkciji `ask` nije M ili `w[i]` nije ni 0 ni 1 za neko i ($0 \leq i \leq M - 1$).
- `more than 100 calls to ask`: Funkcija `ask` je pozvana više od 100 puta.
- `{s, t} is wrong`: Funkcija `answer` je pozvana sa netačnim parom `s` i `t`.