Highway Tolls

Au Japon, les villes sont connectées par un réseau d'autoroutes. Ce réseau est composé de N villes et M autoroutes. Chaque autoroute relie deux villes distinctes. Deux autoroutes différentes ne relient jamais les deux mêmes villes. Les villes sont numérotées de 0 à $N-1$, et les autoroutes sont numérotées de 0 à $M-1$. Vous pouvez circuler dans les deux directions sur chaque autoroute. Vous pouvez voyager de chaque ville à toute autre ville en utilisant le réseau autoroutier.

Sur chaque autoroute vous devez payer un péage. Le prix du péage de chaque autoroute dépend des conditions de **trafic** sur cette autoroute. Le trafic peut être **fluide** ou **dense**. Quand le trafic est fluide, le péage coûte A yens (monnaie japonaise). Quand le trafic est dense, le péage coûte B yens. Il est garanti que $A < B$. Notez que vous connaissez les valeurs de A et B.

Vous avez à votre disposition une machine qui, étant donné les conditions de trafic sur chaque autoroute, calcule la somme minimale qu'un automobiliste voyageant de la ville S à la ville T ($S \neq T$) doit payer en payant chacune des autoroutes sur son trajet.

Cette machine n'est cependant qu'un prototype. Les valeurs de S et T sont fixées (codées en dur dans la machine) mais elles vous sont inconnues. Vous souhaitez déterminer S et T. Pour y parvenir, vous avez prévu de tester plusieurs conditions de trafic sur la machine, et d'utiliser les différentes réponses de la machine afin de conclure S et T. Fournir les conditions de trafic étant coûteux, vous ne voulez pas utiliser la machine trop de fois.

Détails d'implémentation

Votre programme doit définir la procédure suivante :

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

- `N` : le nombre de villes.
- `U` et `V` : tableaux de taille M, où M est le nombre d'autoroutes. Pour chaque i $(0 \leq i \leq M-1)$, l'autoroute i relie les villes `U[i]` et `V[i]`.
- `A` : le coût du péage sur une autoroute dont le trafic est fluide.
- `B` : le coût du péage sur une autoroute dont le trafic est dense.
- Cette procédure est appelée exactement une fois par test.
- Notez que M est égal à la longueur des tableaux, et peut être obtenue en suivant

les indications de la notice d'implémentation.

La procédure `find_pair` peut appeler les fonctions suivantes :

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

- La longueur de `w` doit être M. Le tableau `w` décrit les conditions de trafic.
- Pour chaque i ($0 \leq i \leq M-1$), `w[i]` donne les conditions de trafic sur l'autoroute i. La valeur de `w[i]` ne peut être que 0 ou 1.
  - `w[i] = 0` signifie que le trafic est fluide sur l'autoroute i.
  - `w[i] = 1` signifie que le trafic est dense sur l'autoroute i.
- Cette fonction retourne le plus petit coût total permettant de voyager de la ville S à la ville T, sous les conditions de trafic spécifiées par `w`.
- Cette fonction peut être appelée au plus 100 fois (pour chaque test).

`find_pair` doit appeler la procédure suivante afin de fournir la solution :

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

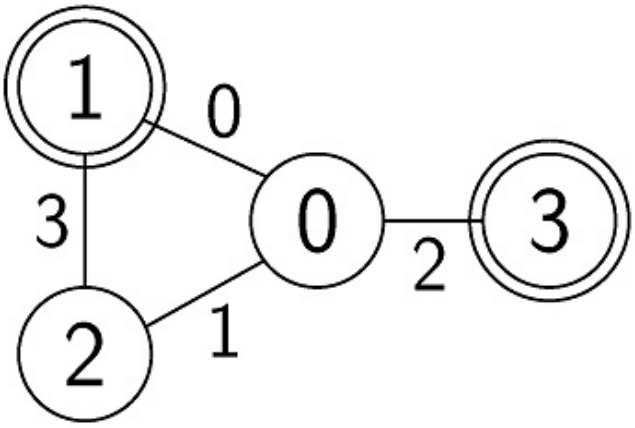
- Les valeurs de `s` et `t` doivent être S et T (l'ordre n'importe pas).
- Cette procédure doit être appelée exactement une fois.

Si une des conditions ci-dessus n'est pas satisfaite, votre programme obtiendra le jugement **`Wrong Answer`**. Sinon, votre programme obtiendra le jugement **`Accepted`** et votre score sera calculé en fonction du nombre d'appels à la fonction `ask` (voir Sous-tâches).

Exemple

Soit $N = 4$, $M = 4$, $U = [0, 0, 0, 1]$, $V = [1, 2, 3, 2]$, $A = 1$, $B = 3$, $S = 1$, et $T = 3$.

L'évaluateur appelle `find_pair(4, [0, 0, 0, 1], [1, 2, 3, 2], 1, 3)`.



Dans la figure précédente, l'arête (edge) numérotée i correspond à l'autoroute i. Une séquence d'appels possible à la fonction `ask` ainsi que les valeurs de retour correspondantes sont listées ci-dessous :

Appel	Retour
`ask([0, 0, 0, 0])`	2
`ask([0, 1, 1, 0])`	4
`ask([1, 0, 1, 0])`	5
`ask([1, 1, 1, 1])`	6

Lors de l'appel de fonction `ask([0, 0, 0, 0])` le trafic de chacune des autoroutes est fluide, le prix de chaque péage est donc de 1. La route la moins chère de $S = 1$ à $T = 3$ est $1 \to 0 \to 3$. Le coût total de cet itinéraire est de 2. La valeur renvoyée est donc 2.

Pour obtenir la bonne réponse, la procédure `find_pair` doit appeler `answer(1, 3)` ou `answer(3, 1)`.

Le fichier `sample-01-in.txt` de l'archive zip correspond à cet exemple. D'autres entrées d'exemple sont également disponibles dans l'archive.

Contraintes

- $2 \leq N \leq 90\,000$
- $1 \leq M \leq 130\,000$
- $1 \leq A < B \leq 1\,000\,000\,000$
- Pour chaque $0 \leq i \leq M - 1$
  - $0 \leq U[i] \leq N - 1$
  - $0 \leq V[i] \leq N - 1$
  - $U[i] \neq V[i]$
- $(U[i], V[i]) \neq (U[j], V[j])$ et $(U[i], V[i]) \neq (V[j], U[j])$ $(0 \leq i < j \leq M - 1)$
- Vous pouvez voyager de chaque ville à toute autre ville en utilisant le réseau autoroutier.
- $0 \leq S \leq N - 1$
- $0 \leq T \leq N - 1$
- $S \neq T$

Dans ce problème, l'évaluateur n'est PAS adaptatif. Cela signifie que S et T sont fixées au début de l'exécution de l'évaluateur et que leur valeur ne dépend pas des requêtes effectuées par votre programme.

Sous-tâches

1. (5 points) S ou T vaut 0, $N \leq 100$, $M = N - 1$
2. (7 points) S ou T vaut 0, $M = N - 1$
3. (6 points) $M = N - 1$, $U[i] = i$, $V[i] = i + 1$ $(0 \leq i \leq M - 1)$
4. (33 points) $M = N - 1$
5. (18 points) $A = 1$, $B = 2$
6. (31 points) Aucune contrainte additionnelle

Considérons que votre programme obtient le jugement **`Accepted`**, et fait X appels à la fonction `ask`. Votre score P pour ce test, en fonction de la sous-tâche, est calculé de la façon suivante :

- Sous-tâche 1. $P = 5$.
- Sous-tâche 2. Si $X \leq 60$, $P = 7$. Sinon $P = 0$.
- Sous-tâche 3. Si $X \leq 60$, $P = 6$. Sinon $P = 0$.
- Sous-tâche 4. Si $X \leq 60$, $P = 33$. Sinon $P = 0$.
- Sous-tâche 5. Si $X \leq 52$, $P = 18$. Sinon $P = 0$.
- Sous-tâche 6.
  - Si $X \leq 50$, $P = 31$.
  - Si $51 \leq X \leq 52$, $P = 21$.
  - Si $53 \leq X$, $P = 0$.

Notez que votre score sur chaque sous-tâche est le score minimum parmi les scores obtenus sur les tests de cette sous-tâche.

Évaluateur d'exemple (Sample grader)

L'évaluateur d'exemple lit l'entrée dans le format suivant :

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

Si votre solution obtient le jugement **`Accepted`**, l'évaluateur d'exemple affiche `Accepted: q`, avec `q` le nombre d'appels à la fonction `ask`.

Si votre solution obtient le jugement **`Wrong Answer`**, il affiche `Wrong Answer: MSG`, avec `MSG` l'un des messages ci-dessous :

- `answered not exactly once` : La procédure `answer` n'a pas été appelée exactement une fois.
- `w is invalid` : La longueur du tableau `w` fourni à la fonction `ask` est différente de M, ou `w` contient une valeur autre que 0 ou 1.
- `more than 100 calls to ask` : La fonction `ask` est appelée plus de 100 fois.
- `{s, t} is wrong`: La procédure `answer` est appelée avec une paire `s` et `t` incorrecte.