Highway Tolls

Στην Ιαπωνία, οι πόλεις συνδέονται μεταξύ τους με ένα δίκτυο δρόμων. Το δίκτυο αποτελείται από N πόλεις και M δρόμους. Κάθε δρόμος ενώνει δύο διαφορετικές μεταξύ τους πόλεις. Για κάθε ζεύγος πόλεων υπάρχει το πολύ ένας δρόμος που τις ενώνει. Οι πόλεις είναι αριθμημένες από το 0 μέχρι και το $N - 1$ και οι δρόμοι από το 0 μέχρι και το $M - 1$. Σε κάθε δρόμο μπορείτε να οδηγήσετε και προς τις δύο κατευθύνσεις. Μπορείτε να πάτε από οποιαδήποτε πόλη σε οποιαδήποτε άλλη πόλη χρησιμοποιώντας τους δρόμους.

Σε κάθε δρόμο υπάρχουν διόδια. Το κόστος των διοδίων κάθε δρόμου εξαρτάται από την **κίνηση** που υπάρχει στον δρόμο αυτό. Η κίνηση μπορεί να είναι **αραιή** ή **πυκνή**. Όταν η κίνηση είναι αραιή, το κόστος είναι A ευρώ. Όταν η κίνηση είναι πυκνή, το κόστος είναι B ευρώ. Θεωρήστε δεδομένο ότι $A < B$. Προσέξτε ότι γνωρίζετε τις τιμές των A και B.

Έχετε μία μηχανή η οποία, αν της δοθούν οι συνθήκες κίνησης σε όλους τους δρόμους, υπολογίζει το ελάχιστο συνολικό κόστος που πρέπει κανείς να πληρώσει για να ταξιδέψει από την πόλη S στην πόλη T ($S \neq T$), κάτω από τις συγκεκριμένες συνθήκες κίνησης.

Η μηχανή, όμως, είναι μόνο ένα πρωτότυπο. Οι τιμές των S και T είναι σταθερές (δηλαδή hardcoded μέσα στην μηχανή) και δεν τις γνωρίζετε. Θέλετε να βρείτε τις τιμές των S και T. Για να το πετύχετε, σκοπεύετε να ορίσετε συγκεκριμένες συνθήκες κίνησης δρόμων στη μηχανή και να χρησιμοποιήσετε το κόστος διοδίων που επιστρέφει η μηχανή ώστε να συμπεράνετε τις τιμές των S και T. Όμως, η διαδικασία αυτή είναι δαπανηρή και δε θέλετε να χρησιμοποιήσετε τη μηχανή πολλές φορές.

Λεπτομέρειες υλοποίησης

Πρέπει να υλοποιήσετε την ακόλουθη συνάρτηση:

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

- `N`: το πλήθος των πόλεων.
- `U` και `V`: πίνακες μήκους M, όπου M το πλήθος των δρόμων που συνδέουν τις πόλεις. Για κάθε i ($0 \leq i \leq M - 1$), ο δρόμος i ενώνει τις πόλεις `U[i]` και `V[i]`.
- `A`: το κόστος διοδίων κάθε δρόμου όταν η κίνηση είναι αραιή.
- `B`: το κόστος διοδίων κάθε δρόμου όταν η κίνηση είναι πυκνή.

- Αυτή η συνάρτηση καλείται μόνο μία φορά για κάθε περίπτωση ελέγχου.
- Προσέξτε ότι η τιμή του M είναι το μήκος των πινάκων και μπορεί να βρεθεί όπως εξηγείται στο φυλλάδιο των Σημειώσεων Υλοποίησης.

Η συνάρτηση `find_pair` μπορεί να καλέσει την ακόλουθη συνάρτηση:

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

- Το μήκος του `w` πρέπει να είναι M. Ο πίνακας `w` περιγράφει τις συνθήκες κίνησης.
- Για κάθε i ($0 \leq i \leq M-1$), το `w[i]` ορίζει τις συνθήκες κίνησης στον δρόμο i. Η τιμή του `w[i]` θα είναι 0 ή 1.
  - `w[i] = 0` σημαίνει ότι η κίνηση στον δρόμο i είναι αραιή.
  - `w[i] = 1` σημαίνει ότι η κίνηση στον δρόμο i είναι πυκνή.
- Η συνάρτηση επιστρέφει το ελάχιστο συνολικό κόστος που απαιτείται για να ταξιδέψει κανείς από την πόλη S στην πόλη T, με τις συνθήκες κίνησης που περιγράφονται στον πίνακα `w`.
- Η συνάρτηση αυτή μπορεί να κληθεί το πολύ 100 φορές (για κάθε περίπτωση ελέγχου).

Η συνάρτηση `find_pair` πρέπει να καλέσει την ακόλουθη συνάρτηση για να δώσει την απάντηση:

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

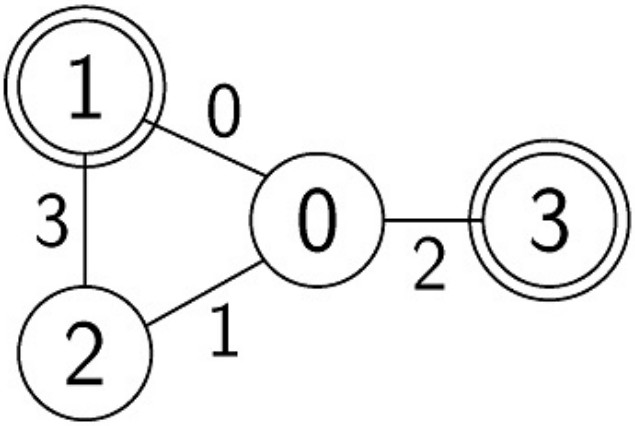
- `s` και `t` πρέπει να είναι το ζευγάρι S και T (η σειρά δεν έχει σημασία).
- Αυτή η συνάρτηση πρέπει να κληθεί ακριβώς μία φορά.

Αν κάποια από τις παραπάνω συνθήκες δεν ικανοποιείται, το πρόγραμμά σας θεωρείται ότι δίνει **`Wrong Answer`**. Διαφορετικά, το πρόγραμμά σας θεωρείται **`Accepted`** και το σκορ σας υπολογίζεται βάσει του πλήθους των κλήσεων στην `ask` που έκανε (βλ. Υποπροβλήματα).

Παράδειγμα

Έστω ότι $N = 4$, $M = 4$, $U = [0, 0, 0, 1]$, $V = [1, 2, 3, 2]$, $A = 1$, $B = 3$, $S = 1$ και $T = 3$.

Ο βαθμολογητής καλεί `find_pair(4, [0, 0, 0, 1], [1, 2, 3, 2], 1, 3)`.



Στην παραπάνω εικόνα, η ακμή με τιμή i αντιστοιχεί στον δρόμο i. Κάποιες πιθανές κλήσεις στην `ask` και οι αντίστοιχες επιστρεφόμενες τιμές τους εμφανίζονται πιο κάτω:

Κλήση συνάρτησης	Επιστρεφόμενη τιμή
`ask([0, 0, 0, 0])`	2
`ask([0, 1, 1, 0])`	4
`ask([1, 0, 1, 0])`	5
`ask([1, 1, 1, 1])`	6

Στην κλήση `ask([0, 0, 0, 0])`, η κίνηση σε κάθε δρόμο είναι αραιή και το κόστος διοδίων κάθε δρόμου είναι 1. Η φθηνότερη διαδρομή από την $S = 1$ στην $T = 3$ είναι $1 \to 0 \to 3$. Το συνολικό κόστος για αυτή τη διαδρομή είναι 2. Επομένως, η συνάρτηση επιστρέφει 2.

Για να θεωρηθεί σωστή η απάντηση, η συνάρτηση `find_pair` πρέπει να καλέσει `answer(1, 3)` ή `answer(3, 1)`.

Το αρχείο `sample-01-in.txt` στο συμπιεσμένο πακέτο αντιστοιχεί σε αυτό το παράδειγμα. Στο πακέτο θα βρείτε και επιπρόσθετα παραδείγματα.

Περιορισμοί

- $2 \leq N \leq 90\,000$
- $1 \leq M \leq 130\,000$
- $1 \leq A < B \leq 1\,000\,000\,000$
- Για κάθε $0 \leq i \leq M - 1$
  - $0 \leq U[i] \leq N - 1$
  - $0 \leq V[i] \leq N - 1$
  - $U[i] \neq V[i]$

- $(U[i], V[i]) \neq (U[j], V[j])$ και $(U[i], V[i]) \neq (V[j], U[j])$ $(0 \leq i < j \leq M - 1)$
- Μπορείτε να ταξιδέψετε από οποιαδήποτε πόλη σε οποιαδήποτε άλλη πόλη χρησιμοποιώντας τους δρόμους.
- $0 \leq S \leq N - 1$
- $0 \leq T \leq N - 1$
- $S \neq T$

Σε αυτό το πρόβλημα, ο βαθμολογητής ΔΕΝ είναι προσαρμοστικός. Αυτό σημαίνει ότι οι τιμές των S και T είναι σταθερές και γνωστές από την αρχή της εκτέλεσης του βαθμολογητή και δεν εξαρτώνται από τα ερωτήματα που θέτει η λύση σας.

Υποπροβλήματα

1. (5 βαθμοί) ένα από τα S ή T είναι 0, $N \leq 100$, $M = N - 1$
2. (7 βαθμοί) ένα από τα S ή T είναι 0, $M = N - 1$
3. (6 βαθμοί) $M = N - 1$, $U[i] = i$, $V[i] = i + 1$ $(0 \leq i \leq M - 1)$
4. (33 βαθμοί) $M = N - 1$
5. (18 βαθμοί) $A = 1$, $B = 2$
6. (31 βαθμοί) Κανένας επιπρόσθετος περιορισμός

Υποθέστε ότι το πρόγραμμά σας θεωρήθηκε **Accepted** και έκανε X κλήσεις στην `ask`. Το σκορ σας P για την περίπτωση ελέγχου, αναλόγως με τον αριθμό του υποπροβλήματος, υπολογίζεται ως εξής:

- Υποπρόβλημα 1. $P = 5$.
- Υποπρόβλημα 2. Αν $X \leq 60$, τότε $P = 7$. Διαφορετικά $P = 0$.
- Υποπρόβλημα 3. Αν $X \leq 60$, τότε $P = 6$. Διαφορετικά $P = 0$.
- Υποπρόβλημα 4. Αν $X \leq 60$, τότε $P = 33$. Διαφορετικά $P = 0$.
- Υποπρόβλημα 5. Αν $X \leq 52$, τότε $P = 18$. Διαφορετικά $P = 0$.
- Υποπρόβλημα 6.
  - Αν $X \leq 50$, τότε $P = 31$.
  - Αν $51 \leq X \leq 52$, τότε $P = 21$.
  - Αν $53 \leq X$, τότε $P = 0$.

Προσέξτε ότι το σκορ κάθε υποπροβλήματος είναι το ελάχιστο των σκορ όλων των περιπτώσεων ελέγχου του υποπροβλήματος.

Υποδειγματικός βαθμολογητής

Ο υποδειγματικός βαθμολογητής διαβάζει την είσοδο ως εξής:

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

Αν το πρόγραμμά σας θεωρηθεί **Accepted**, ο υποδειγματικός βαθμολογητής εκτυπώνει

`Accepted: q`, όπου `q` το πλήθος των κλήσεων στην `ask`.

Αν το πρόγραμμά σας θεωρηθεί ότι δίνει **`Wrong Answer`**, εκτυπώνει `Wrong Answer: MSG`. Η σημασία του `MSG` είναι η εξής:

- `answered not exactly once`: Η συνάρτηση `answer` δεν κλήθηκε ακριβώς μία φορά.
- `w is invalid`: Το μήκος του `w` που δίνεται στην `ask` δεν είναι M ή η τιμή `w[i]` δεν είναι ούτε 0 ούτε 1 για κάποιο i ($0 \leq i \leq M-1$).
- `more than 100 calls to ask`: Η συνάρτηση `ask` κλήθηκε πάνω από 100 φορές.
- `{s, t} is wrong`: Η συνάρτηση `answer` κλήθηκε με λανθασμένες τιμές των `s` και `t`.