# B. Menukar Kota

| Time limit | 2 s |
| --- | --- |
| Memory limit | 512 MB |

## Deskripsi

Terdapat $N$ buah kota di Indonesia, dinomori dari $0$ hingga $N-1$. Terdapat juga $M$ buah jalan dua arah, dinomori dari $0$ hingga $M-1$. Setiap jalan menghubungkan sepasang kota yang berbeda. Jalan ke-$i$ menghubungkan kota bernomor $U[i]$ dan kota bernomor $V[i]$, serta memerlukan $W[i]$ unit bensin untuk dilalui sebuah mobil dari salah satu ujung ke ujung yang lainnya. Kota-kota tersebut terhubung sedemikian rupa sehingga untuk setiap kota, terdapat minimal satu cara untuk pergi dari kota tersebut ke kota lainnya.

Selama $Q$ hari ke depan, terdapat tepat sepasang kota yang hendak membentuk hubungan politik. Secara formal, pada hari ke-$j$, kota bernomor $X[j]$ hendak membentuk hubungan politik dengan kota bernomor $Y[j]$. Untuk membentuk hubungan politik tersebut, kota bernomor $X[j]$ harus mengirim seorang perwakilan ke kota bernomor $Y[j]$ menggunakan mobil. Sebaliknya, kota bernomor $Y[j]$ juga harus mengirim perwakilan ke kota bernomor $X[j]$ menggunakan mobil.

Untuk menghindari kemacetan, mobil yang dikirim kedua kota tidak boleh bertemu di satu titik pada waktu yang sama. Lebih tepatnya, kedua mobil tidak boleh berada pada kota yang sama pada waktu yang sama, dan mereka tidak boleh melewati jalan yang sama dalam arah yang berlawanan pada waktu yang sama. Selain itu, ketika sebuah mobil memasuki sebuah jalan, mobil tersebut harus berjalan hingga mencapai kota tujuan (Dengan kata lain, mobil tidak boleh berputar balik). Namun, sebuah mobil diperbolehkan untuk mengunjungi sebuah kota lebih dari sekali, melewati sebuah jalan lebih dari sekali, dan menunggu kapanpun di kota apapun.

Karena mobil dengan kapasitas bensin yang tinggi cenderung lebih mahal, kedua kota memutuskan untuk memilih rute yang terbaik, sehingga kapasitas bensin maksimum yang diperlukan kedua mobil seminimum mungkin. Setiap mobil dapat mengisi bensin di kota manapun dengan jumlah berapapun, sehingga kapasitas bensin yang diperlukan sebuah mobil untuk melewati sebuah rute sama dengan konsumsi gas **maksimum** pada semua jalan di rute tersebut.

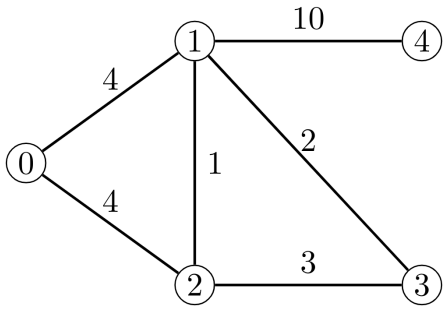
## Tugas

Anda diminta untuk mengimplementasikan fungsi `init` dan `getMinimumFuelCapacity`.

- `init(N, M, U, V, W)` Fungsi ini akan dipanggil *grader* tepat sekali sebelum pemanggilan fungsi `getMinimumFuelCapacity`.
  - $N$: Sebuah bilangan bulat yang menyatakan banyak kota.
  - $M$: Sebuah bilangan bulat yang menyatakan banyak jalan.
  - $U$: Sebuah *Array* yang berisi $M$ buah bilangan bulat, mendeskripsikan salah satu ujung dari setiap jalan.
  - $V$: Sebuah *Array* yang berisi $M$ buah bilangan bulat, mendeskripsikan ujung lainnya dari setiap jalan.
  - $W$: Sebuah *Array* yang berisi $M$ buah bilangan bulat, mendeskripsikan jumlah gas yang diperlukan untuk melewati sebuah jalan.
- `getMinimumFuelCapacity(X, Y)` - Fungsi ini akan dipanggil *grader*sebanyak $Q$ kali.
  - $X$: Kota pertama yang hendak membuat hubungan politik.
  - $Y$: Kota kedua yang hendak membuat hubungan politik.
  - Fungsi ini harus mengembalikan sebuah bilangan bulat: Kapasitas bensin minimum yang diperlukan kedua mobil sehingga seorang perwakilan dari kota $X$ dapat berkendara menuju kota $Y$ dan seorang perwakilan dari kota $Y$ dapat berkendara menuju kota $X$mengikuti peraturan yang telah dijelaskan sebelumnya, atau $-1$ jika tidak mungkin.

## Contoh Kasus

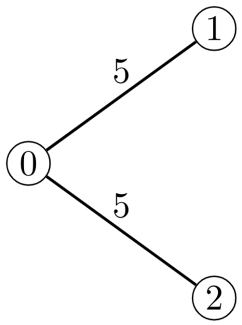
Pada contoh kasus pertama, $N = 5$, $M = 6$, $U = [0, 0, 1, 1, 1, 2]$, $V = [1, 2, 2, 3, 4, 3]$, $W = [4, 4, 1, 2, 10, 3]$, $Q = 3$, $X = [1, 2, 0]$, $Y = [2, 4, 1]$. Contoh tersebut digambarkan pada ilustrasi sebagai berikut:




*Grader* pada awalnya akan memanggil `init(5, 6, [0, 0, 1, 1, 1, 2], [1, 2, 2, 3, 4, 3], [4, 4, 1, 2, 10, 3])`. Setelah itu, *grader* akan memanggil fungsi-fungsi berikut:

- `getMinimumFuelCapacity(1, 2)`. Pertama, mobil dari kota pertama akan menuju kota ketiga. Lalu, mobil dari kota kedua akan menuju kota pertama. Terakhir, mobil yang sekarang berada di kota ketiga akan menuju kota kedua. Maka, kapasitas bensin maksimum yang diperlukan kedua mobil adalah $3$ unit bensin (diperlukan untuk berkendara dari kota ketiga ke kota kedua pada langkah terakhir). Tidak ada rute yang memerlukan kapasitas bensin yang lebih sedikit, sehingga fungsi harus mengembalikan angka $3$.
- `getMinimumFuelCapacity(2, 4)`. Semua mobil yang pergi dari atau menuju kota keempat pasti memerlukan kapasitas bensin sebanyak $10$unit, sehingga fungsi harus mengembalikan angka $10$.
- `getMinimumFuelCapacity(0, 1)`. Fungsi harus mengembalikan angka $4$.

Pada contoh kedua, $N = 3$, $M = 2$, $U = [0, 0]$, $V = [1, 2]$, $W = [5, 5]$, $Q = 1$, $X = [1]$, $Y = [2]$. Contoh tersebut digambarkan pada ilustrasi sebagai berikut:




*Grader* pada awalnya akan memanggil `init(3, 2, [0, 0], [1, 2], [5, 5])`. Berikutnya, *grader* akan memanggil fungsi-fungsi berikut:

- `getMinimumFuelCapacity(1, 2)`. Tidak ada rute yang memenuhi sehingga kedua mobil dapat pergi dari kota pertama ke kota kedua (dan sebaliknya) tanpa bertemu pada satu titik, sehingga fungsi harus mengembalikan angka $-1$.

## Batasan

- $2 \leq N \leq 100\,000$.
- $N - 1 \leq M \leq 200\,000$.
- $0 \leq U[i] < V[i] < N$.

- Terdapat paling banyak satu jalan untuk setiap pasang kota.
- Terdapat setidaknya satu buah rute yang menghubungkan setiap pasang kota.
- $1 \leq W[i] \leq 10^9$.
- $1 \leq Q \leq 200\,000$.
- $0 \leq X[j] < Y[j] < N$.

### Subsoal 1 (6 poin)

- Setiap kota memiliki paling banyak dua jalan yang terhubung dengannya.

### Subsoal 2 (7 poin)

- $M = N - 1$.
- $U[i] = 0$.

### Subsoal 3 (17 poin)

- $Q \leq 5$.
- $N \leq 1\,000$.
- $M \leq 2\,000$.

### Subsoal 4 (20 poin)

- $Q \leq 5$.

### Subsoal 5 (23 poin)

- $M = N - 1$.

### Subsoal 6 (27 poin)

- Tidak ada batasan tambahan.

## Contoh Grader

Contoh *grader* akan membaca masukan dengan format berikut ini:

```
N M
U[0] V[0] W[0]
U[1] V[1] W[1]
.
.
.
U[M-1] V[M-1] W[M-1]
Q
X[0] Y[0]
X[1] Y[1]
.
.
.
X[Q-1] Y[Q-1]
```

Untuk setiap pemanggilan `getMinimumFuelCapacity`, *grader* akan mencetak angka yang dikembalikan oleh fungsi.