CSGE602040 - Struktur Data dan Algoritma
Semester Genap - 2020/2021
TP 2
Deadline: Minggu, 13-Juni-2021, 14.00 WIB

Distribusi Vaksin Negara Api

Sumber: https://truckmagz.com/fitur-baru-truk-hino-untuk-segmen-kargo/

Deskripsi

Demi melawan COVID-19 yang melanda Negara Api, pemerintah mengambil langkah tegas untuk memproduksi vaksin yang ampuh menghapus virus ini. Beberapa bulan lalu, pemerintah telah memproduksi, melakukan uji coba, dan mendistribusikan vaksin tersebut ke beberapa bank vaksin yang tersebar di seluruh Negara Api. Saat ini, pemerintah memerintahkan kota-kota tersebut untuk mendistribusikan vaksin-vaksin ke kota-kota lain yang belum memiliki vaksin.

Dalam mewujudkan upaya ini, pemerintah menunjuk dr. Watame sebagai koordinator. Anda merupakan seorang *Software Engineer* ternama yang dipercayai dr. Watame untuk membangun sebuah algoritma utama dalam sistem pendistribusian vaksin ini. Algoritma yang dibuat haruslah efektif, efisien, dan 100% optimal. Sebagai seorang *Software Engineer* yang baik dan profesional, bantulah dr. Watame dalam membangun sistemnya.

Negara Api akan memiliki konfigurasi jalan-jalan di negaranya. Konfigurasi ini berisikan tergantung tipe dari jalan tersebut. Ada 3 tipe jalan, yaitu tipe 0, tipe 1, dan tipe 2. Jalan tipe 0 dan tipe 1 tidak memiliki batasan arah sehingga dapat digunakan dua arah. Sedangkan jalan tipe 2 batasan arah. Jalan dengan tipe 0 memiliki konfigurasi ***A***, ***B***, dan ***W***. Artinya, jalan tersebut memiliki panjang ***W***, dan menghubungkan kota ***A*** dengan kota ***B*** (secara langsung, tidak harus melewati kota lain terlebih dahulu). Jalan dengan tipe 1 atau tipe 2, konfigurasinya sama-sama ***A*** dan ***B*** saja, dan sama-sama tidak memiliki panjang sehingga ketika digunakan panjangnya adalah 0. Konfigurasi-konfigurasi yang sudah dibuat dapat dihapus menggunakan **DELETE** sesuai dengan tipe dari jalan tersebut. Tidak semua kota terhubung dengan kota lainnya dengan artian ada kota-kota di Negara Api yang tidak terhubung satu sama lain (tidak ada jalur atau serangkaian jalan-jalan yang menghubungkan). Ringkasan dari tipe jalan yang telah disebutkan dapat dilihat pada Tabel 1.

Tabel 1. Ringkasan Tipe Jalan dan Propertinya

Tipe Jalan	Satu Arah / Dua Arah	Panjang	Konfigurasi Insert	Konfigurasi Delete
Tipe 0	Dua arah	Memiliki panjang	`INSERT 0 A B W`	`DELETE 0 A B`
Tipe 1	Dua arah	Tidak memiliki panjang (0)	`INSERT 1 A B`	`DELETE 1 A B`
Tipe 2	Satu arah	Tidak memiliki panjang (0)	`INSERT 2 A B`	`DELETE 2 A B`

Selain konfigurasi tersebut, pemerintah juga mengeluarkan perintah kepada kota-kota tersebut. Berikut adalah jenis-jenis perintah yang dikeluarkan oleh pemerintah.

- **`SHORTEST_PATH 0 A B`**
  Untuk memotong waktu, pemerintah mengeluarkan perintah **`SHORTEST_PATH 0`** untuk menemukan jalur distribusi terpendek antara kota ***A*** dengan ***B*** dengan hanya menggunakan jalan tipe 0. **Perintah ini akan menghasilkan jarak terpendek tersebut.** Apabila rute dari kota ***A*** dengan ***B*** tidak tersedia, maka perintah akan mengeluarkan "**−1**" (tanpa tanda kutip). ***A* dengan *B* dijamin adalah kota yang berbeda.**

- **`SHORTEST_PATH 1 A B`**
  Untuk memotong waktu, pemerintah mengeluarkan perintah **`SHORTEST_PATH 1`** untuk menemukan jalur distribusi terpendek antara kota ***A*** dengan ***B***. Pemerintah memperbolehkan pengemudi truk untuk melewati jalan tipe 0 sebanyak apapun dan jalan tipe 1 sebanyak maksimal satu kali. **Perintah ini akan menghasilkan jarak terpendek tersebut.** Apabila rute dari kota ***A*** dengan ***B*** tidak tersedia, maka perintah akan mengeluarkan "**−1**" (tanpa tanda kutip). ***A* dengan *B* dijamin adalah kota yang berbeda.**

- **`MIN_PATH A B`**
  Pemerintah ingin mengetahui jumlah minimal jalan yang dilalui jika truk pengantar vaksin dari kota ***A*** ingin pergi ke kota ***B***. Perintah ini hanya menggunakan jalan tipe 0. **Perintah ini akan menghasilkan jumlah minimal jalan yang dilalui**. Apabila rute dari kota ***A*** dengan ***B*** tidak tersedia, maka perintah akan mengeluarkan "**−1**" (tanpa tanda kutip). ***A* dengan *B* dijamin adalah kota yang berbeda**.

- **`IS_CONNECTED A B`**
  Pemerintah ingin mengetahui apakah kota ***A*** memiliki jalan untuk mencapai kota ***B***. Perintah ini hanya menggunakan jalan tipe 0. **Perintah ini akan mengeluarkan "1" (tanpa kutip) jika kota *A* memiliki jalan untuk mencapai kota *B*. Jika tidak, perintah ini akan mengeluarkan "0" (tanpa kutip). *A* dengan *B* dijamin adalah kota yang berbeda.**

- **`COUNT_CITY A M`**
  Pemerintah ingin mengetahui berapa jumlah maksimal kota yang dapat dicapai oleh truk pengantar vaksin ketika bergerak dari kota ***A*** dengan jumlah jarak maksimal ***M***. Perintah ini

hanya menggunakan jalan tipe 0. **Perintah ini akan mengeluarkan jumlah maksimal kota yang dapat dicapai termasuk dengan kota *A* itu sendiri**.

- **`COUNT_CONNECTED`**
  Pemerintah ingin mengetahui ada berapa provinsi yang ada di Negara Api. Provinsi didefinisikan sebagai kumpulan kota yang terhubung maksimal dengan jalan tipe 0. **Perintah ini akan mengembalikan jumlah *connected graph* dari konfigurasi jalan-jalan Negara Api saat ini**. *Catatan: Lihat lagi materi di Matematika Diskrit 2 mengenai konsep komponen graf (subgraf terhubung maksimal)*.

- **`SIMULATE_WALK` *`A S`***
  Dimulai di kota ***A***, pemerintah memerintahkan truk untuk mengikuti jalan spesial sebanyak ***S*** jalan (dijamin untuk setiap kota, maksimal memiliki satu jalan spesial yang keluar dari kota tersebut dan masuk ke kota tersebut). Pemerintah ingin mengetahui jika truk berjalan sebanyak ***S*** langkah, truk tersebut akan berhenti di kota apa, **dengan hanya menggunakan jalan tipe 2 saja**. **Perintah ini akan mengembalikan kota tempat truk tersebut berhenti**.

Masukan

Baris pertama berisikan dua buah bilangan yaitu, ***N*** dan ***Q*** yang dipisahkan oleh sebuah tanda spasi.

N adalah jumlah kota yang akan dibuat. Sebagai contoh, jika ***N*** berisikan 7, maka kota yang terbuat adalah kota 0 sampai dengan kota 6.

Q buah baris berikutnya masing-masing berisikan satu konfigurasi jalan dari Negara Api atau perintah dari pemerintah.

Jika baris berisikan konfigurasi, maka akan diawali dengan **`INSERT/DELETE 0`** untuk jalan dengan tipe 0, **`INSERT/DELETE 1`** untuk jalan dengan tipe 1**,** atau **`INSERT/DELETE 2`** untuk jalan dengan tipe 2. Setiap parameter konfigurasi akan bergantung dengan parameter yang dibutuhkan masing-masing konfigurasi tersebut. Untuk detail dari parameter dapat merujuk kepada Tabel 1.

Jika baris berisikan perintah-perintah dari pemerintah yang sudah dijelaskan di atas, maka akan diawali dengan satu string yang diikuti beberapa parameter sesuai dengan perintah yang dituliskan. String pertama menunjukan tipe *query* berdasarkan singkatannya. Sesuai dengan tipe *query* maka berikut adalah parameter setiap *query*nya,

- **`SHORTEST_PATH 0`** diikuti oleh dua buah bilangan bulat tidak negatif ***A*** dan ***B*** yang menunjukkan nama-nama kota.
- **`SHORTEST_PATH 1`** diikuti oleh dua buah bilangan bulat tidak negatif ***A*** dan ***B*** yang menunjukkan nama-nama kota.
- **`MIN_PATH`** diikuti oleh dua buah bilangan bulat tidak negatif ***A*** dan ***B*** yang menunjukkan nama-nama kota.
- **`IS_CONNECTED`** diikuti oleh dua buah bilangan bulat tidak negatif ***A*** dan ***B*** yang menunjukkan nama-nama kota.
- **`COUNT_CITY`** diikuti oleh sebuah bilangan bulat tidak negatif ***A*** yang menunjukkan kota yang menjadi titik awal dan sebuah bilangan bulat tidak negatif ***M*** yang menunjukkan jarak maksimal yang diberikan.

- **SIMULATE_WALK** diikuti oleh sebuah bilangan bulat ***A*** yang menunjukkan titik awal kota dan sebuah bilangan bulat tidak negatif ***S*** yang menunjukkan jarak maksimal yang diberikan.
- sebuah string **COUNT_CONNECTED** yang melakukan pemrosesan *query* menghitung banyaknya provinsi di negara Api.

Keluaran

Keluaran terdiri dari banyak baris perintah yang dibuat. Baris ke-i berisi sebuah angka yang merupakan jawaban dari *query* ke-i.

Batasan

- $1 \leq N \leq 10^4$
- $1 \leq Q \leq 10^4$
- $0 \leq A, B \leq N-1$
- $1 \leq W \leq 10^4$
- $1 \leq M \leq 10^9$
- $1 \leq S \leq 10^{18}$
- $A \neq B$

Semua panjang jalan tipe 0 dijamin bernilai bilangan bulat positif.

Contoh Masukan 1

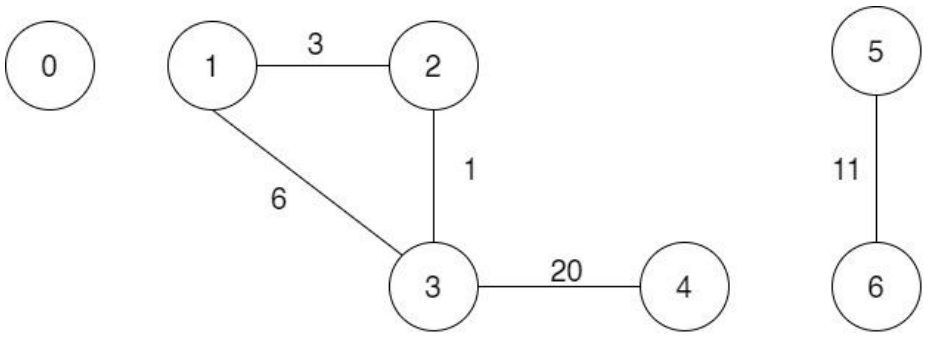
```
7 22
INSERT 0 1 2 3
INSERT 0 1 3 6
INSERT 0 2 3 1
INSERT 0 3 4 20
INSERT 0 5 6 11
SHORTEST_PATH 0 1 2
SHORTEST_PATH 0 1 3
SHORTEST_PATH 0 4 1
MIN_PATH 1 4
MIN_PATH 6 5
IS_CONNECTED 4 2
IS_CONNECTED 1 5
IS_CONNECTED 0 6
COUNT_CITY 2 10
COUNT_CITY 2 100
COUNT_CONNECTED
DELETE 0 1 2
SHORTEST_PATH 0 1 3
IS_CONNECTED 1 2
COUNT_CONNECTED
DELETE 0 1 3
COUNT_CONNECTED
```

Contoh Keluaran

```
3
4
24
2
1
1
0
0
3
4
3
6
1
3
4
```

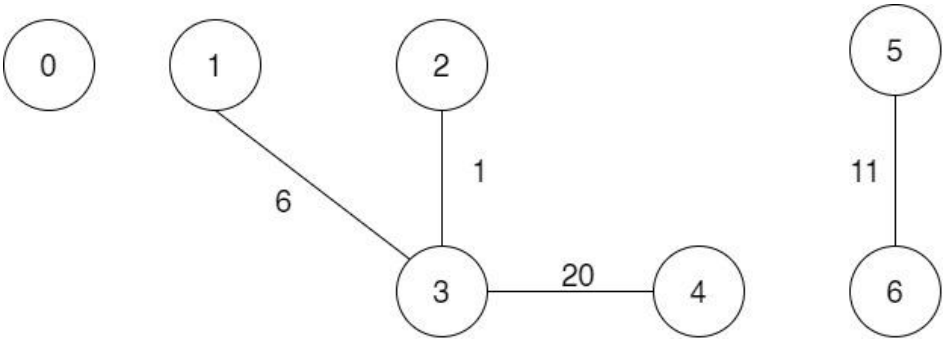
Penjelasan

Gambar graf saat setelah **`INSERT 0 5 6 11`**:



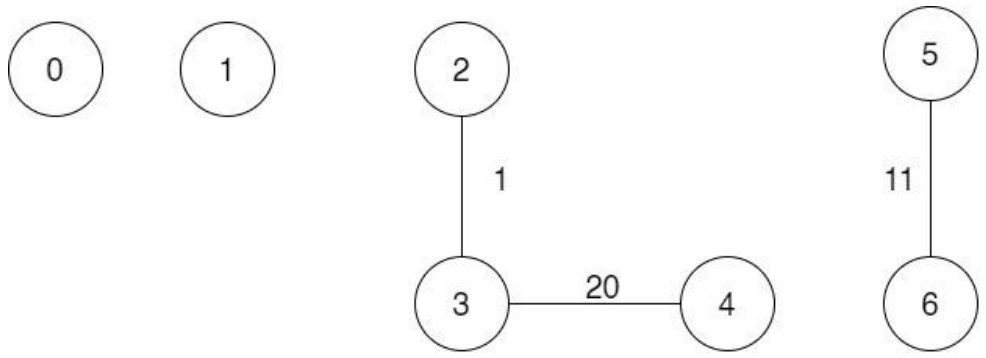
Graf 1.1

Gambar graf saat setelah **`DELETE 0 1 2`**:



Graf 1.2

Gambar graf saat setelah **`DELETE 0 1 3`**:



Graf 1.3

Masukan	Keluaran	Penjelasan
`SHORTEST_PATH 0 1 2`	3	Query ini meminta jarak terpendek dari kota 1 ke kota 2 dengan menggunakan jalan tipe 0 saja. Dapat dilihat pada graf 1.1 bahwa jarak terpendek adalah melalui *path* 1-2 yang memiliki total jarak sebesar 3
`SHORTEST_PATH 0 1 3`	4	Query ini meminta jarak terpendek dari kota 1 ke kota 3 dengan menggunakan jalan tipe 0 saja. Dapat dilihat pada graf 1.1 bahwa jarak terpendek adalah melalui *path* 1-2-3 yang memiliki total jarak sebesar 3+1 = 4
`SHORTEST_PATH 0 4 1`	24	Query ini meminta jarak terpendek dari kota 4 ke kota 1 dengan menggunakan jalan tipe 0 saja. Dapat dilihat pada graf 1.1 bahwa jarak terpendek adalah melalui *path* 4-3-2-1 yang memiliki total jarak sebesar 20+3+1 = 24
`MIN_PATH 1 4`	2	Query ini meminta path dengan banyak jalan tipe 0 minimal dari kota 1 ke kota 4. Dapat dilihat pada graf 1.1 bahwa *path* optimal adalah 1-3-4 dengan banyak jalan sebanyak 2.
`MIN_PATH 6 5`	1	Query ini meminta path dengan banyak jalan tipe 0 minimal dari kota 6 ke kota 5. Dapat dilihat pada graf 1.1 bahwa *path* optimal adalah 6-5 dengan banyak jalan sebanyak 1.
`IS_CONNECTED 4 2`	1	Query ini menanyakan keterhubungan antara kota 4 dengan kota 2. Dapat dilihat pada graf 1.1, karena terdapat *path* yang menggunakan jalan tipe 0 antara kedua kota tersebut, maka kita outputkan 1.
`IS_CONNECTED 1 5`	0	Query ini menanyakan keterhubungan antara kota 1 dengan kota 5. Dapat dilihat pada graf 1.1, karena tidak terdapat *path* yang menggunakan jalan tipe 0

		antara kedua kota tersebut, maka kita outputkan 0.
IS_CONNECTED 0 6	0	Query ini menanyakan keterhubungan antara kota 0 dengan kota 6. Dapat dilihat pada graf 1.1, karena tidak terdapat *path* yang menggunakan jalan tipe 0 antara kedua kota tersebut, maka kita outputkan 0.
COUNT_CITY 2 10	3	Query ini menanyakan banyak kota yang bisa dijangkau oleh kota 2 menggunakan hanya jalan tipe 0 dengan jarak maksimal sebesar 10 (termasuk kota 2 sendiri). Dapat dilihat pada graf 1.1, jawabannya adalah kota 1, 2, dan 3. Output adalah banyaknya kota yaitu 3.
COUNT_CITY 2 100	4	Query ini menanyakan banyak kota yang bisa dijangkau oleh kota 2 menggunakan hanya jalan tipe 0 dengan jarak maksimal sebesar 100 (termasuk kota 2 sendiri). Dapat dilihat pada graf 1.1, jawabannya adalah kota 1, 2, 3, dan 4. Output adalah banyaknya kota yaitu 4.
COUNT_CONNECTED	3	Query ini meminta kita menghitung banyaknya provinsi yang ada pada negara api. Dapat dilihat pada graf 1.1 bahwa jawabannya adalah 3 yaitu {0}, {1, 2, 3, 4}, dan {5, 6}
SHORTEST_PATH 0 1 3	6	Query ini meminta jarak terpendek dari kota 1 ke kota 3 dengan menggunakan jalan tipe 0 saja. Dapat dilihat pada graf 1.2 bahwa jarak terpendek adalah melalui *path* 1-3 yang memiliki total jarak sebesar 6
IS_CONNECTED 1 2	1	Query ini menanyakan keterhubungan antara kota 1 dengan kota 2. Dapat dilihat pada graf 1.2, karena terdapat *path* yang menggunakan jalan tipe 0 antara kedua kota tersebut, maka kita outputkan 1.
COUNT_CONNECTED	3	Query ini meminta kita menghitung banyaknya provinsi yang ada pada negara api. Dapat dilihat pada graf 1.2 bahwa jawabannya adalah 3 yaitu {0}, {1, 2, 3, 4}, dan {5, 6}
COUNT_CONNECTED	4	Query ini meminta kita menghitung banyaknya provinsi yang ada pada negara api. Dapat dilihat pada graf 1.3 bahwa jawabannya adalah 4 yaitu {0}, {1}, {2, 3, 4}, dan {5, 6}

Contoh Masukan 2

```
7 18
INSERT 0 1 2 3
```

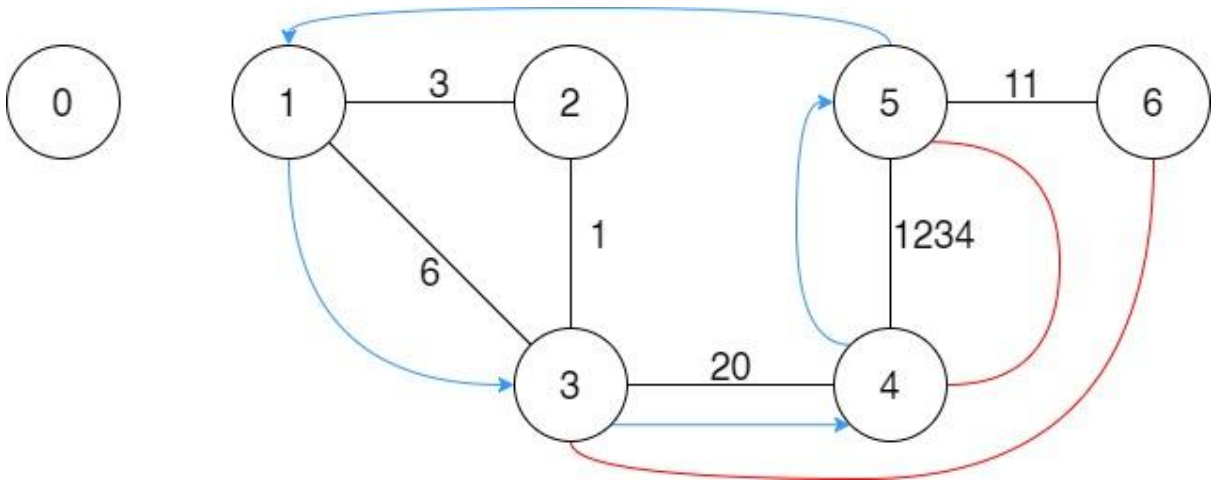
```
INSERT 0 1 3 6
INSERT 0 2 3 1
INSERT 0 3 4 20
INSERT 0 4 5 1234
INSERT 0 5 6 11
INSERT 1 4 5
INSERT 1 6 3
INSERT 2 1 3
INSERT 2 3 4
INSERT 2 4 5
INSERT 2 5 1
SHORTEST_PATH 0 1 5
SHORTEST_PATH 0 1 6
SHORTEST_PATH 1 1 5
SHORTEST_PATH 1 1 6
SIMULATE_WALK 1 6
SIMULATE_WALK 1 1000000000000
```

Contoh Keluaran

```
1258
1269
15
4
4
1
```

Penjelasan

Gambar graf setelah **`INSERT 2 5 1`**:



Graf 2.1

Catatan:

- Jalan warna HITAM adalah jalan tipe 0
- Jalan warna MERAH adalah jalan tipe 1
- Jalan warna BIRU adalah jalan tipe 2

Masukan	Keluaran	Penjelasan
SHORTEST_PATH 0 1 5	1258	Query ini meminta jarak terpendek dari kota 1 ke kota 5 dengan menggunakan jalan tipe 0 saja. Dapat dilihat pada graf 2.1 bahwa jarak terpendek adalah melalui *path* 1-2-3-4-5 yang memiliki total jarak sebesar 3+1+20+1234 = 1258
SHORTEST_PATH 0 1 6	1269	Query ini meminta jarak terpendek dari kota 1 ke kota 6 dengan menggunakan jalan tipe 0 saja. Dapat dilihat pada graf 2.1 bahwa jarak terpendek adalah melalui *path* 1-2-3-4-5-6 yang memiliki total jarak sebesar 3+1+20+1234+11 = 1269
SHORTEST_PATH 1 1 5	15	Query ini meminta jarak terpendek dari kota 1 ke kota 5 dengan menggunakan jalan tipe 0 sebanyak apapun dan jalan tipe 1 sebanyak maksimal 1 kali. Dapat dilihat pada graf 2.1, *Path* optimalnya adalah 1-2-3<>6-5 (<> merupakan jalan tipe 1) yang memiliki total jarak sebesar 3+1+11 = 15
SHORTEST_PATH 1 1 6	4	Query ini meminta jarak terpendek dari kota 1 ke kota 6 dengan menggunakan jalan tipe 0 sebanyak apapun dan jalan tipe 1 sebanyak maksimal 1 kali. Dapat dilihat pada graf 2.1, *Path* optimalnya adalah 1-2-3<>6 (<> merupakan jalan tipe 1) yang memiliki total jarak sebesar 3+1 = 4
SIMULATE_WALK 1 6	4	Query ini meminta kita melakukan simulasi perjalanan truk yang dimulai dari kota 1 menggunakan jalan tipe 2 saja sebanyak 6 langkah. *Path* yang dihasilkan adalah 1-3-4-5-1-3-4. Output merupakan kota dimana truk berhenti yaitu 4
SIMULATE_WALK 1 1000000000000	1	Query ini meminta kita melakukan simulasi perjalanan truk yang dimulai dari kota 1 menggunakan jalan tipe 2 saja sebanyak 1000.000.000.000 langkah. Output merupakan kota dimana truk berhenti yaitu 1

Contoh Masukan 3

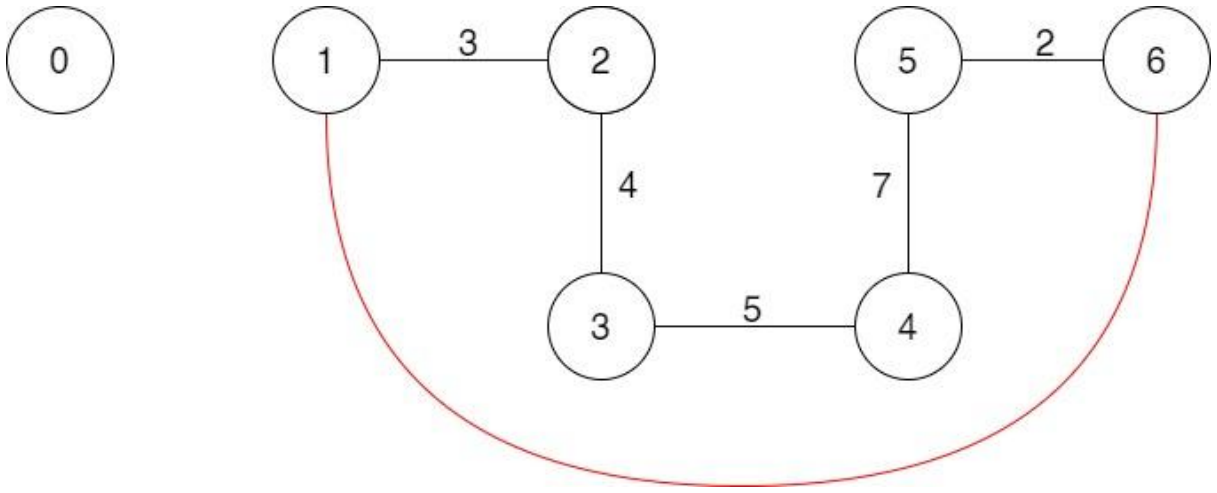
```
7 10
INSERT 0 1 2 3
INSERT 0 2 3 4
INSERT 0 3 4 5
INSERT 0 4 5 7
INSERT 0 5 6 2
INSERT 1 1 6
SHORTEST_PATH 0 3 4
SHORTEST_PATH 1 3 4
SHORTEST_PATH 0 1 6
SHORTEST_PATH 1 1 6
```

Contoh Keluaran

```
5
5
21
0
```

Penjelasan

Gambar graf saat setelah **`INSERT 1 1 6`**:



Graf 3.1

Catatan:

- Jalan warna HITAM adalah jalan tipe 0
- Jalan warna MERAH adalah jalan tipe 1

Masukan	Keluaran	Penjelasan
`SHORTEST_PATH 0 3 4`	5	Query ini meminta jarak terpendek dari kota 3 ke kota 4 dengan menggunakan jalan tipe 0 saja.

		Dapat dilihat pada graf 3.1 bahwa jarak terpendek adalah melalui *path* 3-4 yang memiliki total jarak sebesar 5
`SHORTEST_PATH 1 3 4`	`5`	Query ini meminta jarak terpendek dari kota 3 ke kota 4 dengan menggunakan jalan tipe 0 sebanyak apapun dan jalan tipe 1 sebanyak maksimal 1 kali. Dapat dilihat pada graf 3.1, *Path* optimalnya adalah 3-4 yang memiliki total jarak sebesar 5
`SHORTEST_PATH 0 1 6`	`21`	Query ini meminta jarak terpendek dari kota 1 ke kota 6 dengan menggunakan jalan tipe 0 saja. Dapat dilihat pada graf 3.1 bahwa jarak terpendek adalah melalui *path* 1-2-3-4-5-6 yang memiliki total jarak sebesar 3+4+5+7+2 = 21
`SHORTEST_PATH 1 1 6`	`0`	Query ini meminta jarak terpendek dari kota 1 ke kota 6 dengan menggunakan jalan tipe 0 sebanyak apapun dan jalan tipe 1 sebanyak maksimal 1 kali. Dapat dilihat pada graf 3.1, *Path* optimalnya adalah 1<>6 (<> merupakan jalan tipe 1) yang memiliki total jarak sebesar 0

Pembagian Test Case

Query	Test Case
`SHORTEST_PATH 0`	1 - 6
`SHORTEST_PATH 1`	7 - 12
`MIN_PATH`	13 - 18
`IS_CONNECTED`	19 - 24
`COUNT_CITY`	25 - 30
`SIMULATE_WALK`	31 - 36
`COUNT_CONNECTED`	**37 - 40**
Mixed Queries	**41 - 50**