FAKULTAS ILMU KOMPUTER

CSGE602040 - Struktur Data dan Algoritma
Semester Ganjil - 2022/2023
TP 3
Deadline: Jumat, 9 Desember 2022, 21.00 WIB

Kudeta Negeri Dressrosa

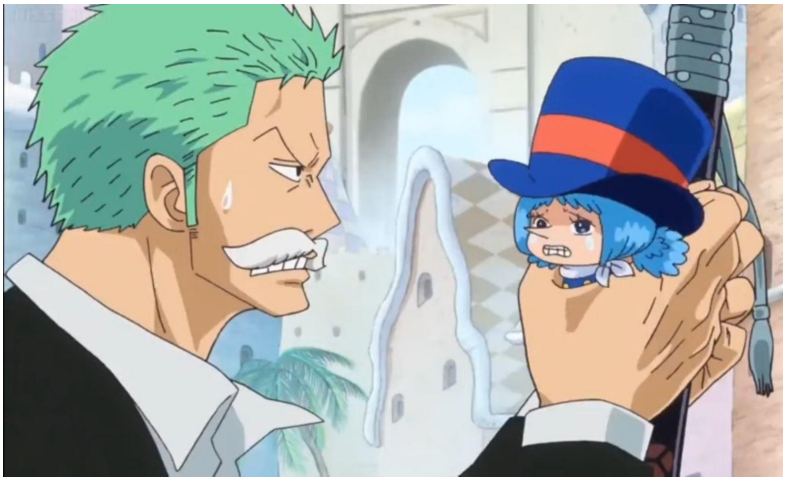
Revisi Dokumen

- Dokumen awal (2022-11-28 18.00)
- Revisi 1 (2022-11-30 07.15): Perubahan masukan *query* pada Contoh Masukan 3

Deskripsi

Suatu hari, negeri Dressrosa sedang mengalami sebuah bencana kelaparan. Hal tersebut terjadi karena raja negeri tersebut, Raja Julien, yang secara paksa mengambil setiap harta penduduknya sehingga penduduk pun menjadi kesusahan untuk membeli bahan-bahan makanan.

Mendengar berita tersebut, Luffy bersama rekan-rekannya berencana untuk membantu rakyat Negeri Dressrosa dengan cara melakukan kudeta secara sembunyi-sembunyi terhadap Raja Julien. Di tengah perjalanan menuju Dressrosa, Luffy & rekan-rekan secara tidak sengaja bertemu dengan sebuah manusia kecil. Ternyata manusia kecil tersebut adalah seorang kurcaci yang berasal dari Kerajaan Tontatta.



Kerajaan Tontatta adalah sebuah kerajaan yang terletak di bawah Negeri Dressrosa. Kurcaci-kurcaci tersebut selama ini hidup secara diam-diam di bawah tanah tanpa diketahui oleh penduduk di permukaan. Mereka menggali sebuah sistem terowongan bawah tanah agar mereka bisa berpindah-pindah tempat dan keberadaan mereka tetap tidak diketahui.

Mendengar kabar tersebut, Luffy meminta tolong kepada para kurcaci Tontatta untuk membantu mereka dalam menjalankan misi kudeta tersebut. Luffy meminta kepada para kurcaci Tontatta untuk membantu menyebarkan informasi dan logistik kepada rekan-rekannya melalui terowongan bawah tanah yang telah mereka buat saat misi dijalankan.

Diketahui bahwa sistem terowongan bawah tanah negeri Tontatta terdiri atas N pos peristirahatan yang dinomori dari 1 hingga N. Pada awalnya, setiap pos hanya dijaga oleh paling banyak satu kurcaci. Selain itu, di negeri Tontatta terdapat M terowongan. Setiap terowongan memiliki panjang L_i, berukuran S_i, dan merupakan terowongan **dua arah** yang menghubungkan dua pos peristirahatan. Dijamin bahwa **setiap pos dapat dicapai dari pos lain** dengan menggunakan terowongan-terowongan yang ada.

Para *programmer* kurcaci Tontatta diberikan sebuah tugas untuk membuat sebuah program komputer yang dapat mengimplementasikan *query-query* berikut.

- **KABUR** $F\ E$
  Seorang penyusup hendak membawa kabur logistik dari pos F ke pos E. Ia ingin membawa logistik sebanyak mungkin, namun jumlah logistik yang dapat dibawa melewati suatu terowongan **tidak dapat melebihi** ukuran terowongan tersebut.

  Keluarkan **jumlah logistik terbanyak** yang dapat dibawa kabur penyusup tersebut dari pos F ke pos E dalam sekali perjalanan.

- **SIMULASI** K **[**$V_1\ V_2$ ... V_k**]**
  Jika seorang penyusup memasuki terowongan, maka sebuah sistem gas beracun akan diaktifkan dan semua kurcaci harus keluar dari sistem terowongan tersebut sebelum gas beracun diaktifkan. Diberikan sebuah daftar pos peristirahatan V_1, V_2, ... , V_k yang menyatakan nomor dari pos yang merupakan pos exit. Diketahui bahwa semua kurcaci bergerak dengan kelajuan 1 satuan jarak per detik.

  Keluarkan **jumlah detik terkecil** agar semua kurcaci dapat berhasil keluar dari sistem terowongan tersebut sebelum sistem gas beracun diaktifkan.

- **SUPER** $V_1\ V_2\ V_3$
  Seorang kurcaci super dapat menggunakan kekuatan super yang dapat membuatnya langsung berpindah ke sebuah pos tetangga dalam waktu nol detik. Akan tetapi, kekuatan ini hanya dapat dilakukan sekali saja selama ia berada di sistem terowongan tersebut.

  Keluarkan **jumlah detik terkecil** yang dapat ditempuh kurcaci super jika ia ingin berpindah dari pos V_1 ke pos V_2, kemudian dilanjutkan ke pos V_3.

Format Masukan

- Baris pertama berisi dua buah bilangan bulat N dan M yang dipisahkan dengan spasi yang menyatakan banyaknya pos dan terowongan.
- M baris selanjutnya terdiri atas empat bilangan bulat A_i, B_i, L_i, dan S_i yang dipisahkan dengan spasi yang menyatakan bahwa pos peristirahatan A_i dan B_i dihubungkan oleh terowongan yang memiliki panjang L_i dan berukuran S_i.

- Baris berikutnya berisi sebuah bilangan bulat P yang menyatakan banyaknya kurcaci yang ada di dalam keseluruhan sistem terowongan.
- P baris berikutnya adalah sebuah bilangan bulat R_i yaitu nomor dari pos yang berisi kurcaci.
- Baris selanjutnya berisi sebuah bilangan bulat Q, yaitu banyak *query* yang akan dijalankan.
- Q baris berikutnya berisi *query* sesuai dengan penjelasan sebelumnya.

Format Keluaran

Format keluaran dari setiap *query* telah dijelaskan pada penjabaran sebelumnya.

Batasan

$1 \leq N \leq 1\,000$
$1 \leq M \leq 2\,000$
$1 \leq A_i, B_i \leq N$
$1 \leq L_i, S_i \leq 1\,000\,000$

$1 \leq P \leq N$
$1 \leq R_i \leq N$
$R_i = R_j \leftrightarrow i = j$

$1 \leq Q \leq 500$

Untuk setiap *query* **KABUR** :

- $1 \leq F, E \leq N$
- $F \neq E$

Untuk setiap *query* **SIMULASI** :

- $1 \leq K \leq min(100, N)$
- $1 \leq V_i \leq N$
- $V_i = V_j \leftrightarrow i = j$

Untuk setiap *query* **SUPER** :

- $1 \leq V_i \leq N$

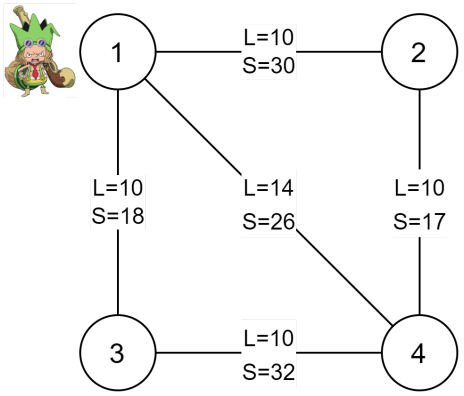
Contoh Masukan 1

```
4 5
1 2 10 30
1 3 10 18
1 4 14 26
2 4 10 17
3 4 10 32
1
1
3
KABUR 2 3
KABUR 3 1
KABUR 4 3
```

Contoh Keluaran 1

```
26
26
32
```

Penjelasan Contoh 1



KABUR 2 3

Banyak logistik terbanyak yang dapat dibawa oleh seorang penyusup dari pos-2 ke pos-3 adalah 26 karena ia masih dapat melalui *path* [2 - 1 - 4 - 3]. Jika banyak logistiknya lebih banyak dari itu, maka tidak mungkin ia akan bisa mencapai pos-3 dari pos-2.

KABUR 3 1

Banyak logistik terbanyak yang dapat dibawa oleh seorang penyusup dari pos-3 ke pos-1 adalah 26 karena ia masih dapat melalui *path* [3 - 4 - 1]. Jika banyak logistiknya lebih banyak dari itu, maka tidak mungkin ia akan bisa mencapai pos-1 dari pos-3.

KABUR 4 3

Banyak logistik terbanyak yang dapat dibawa oleh seorang penyusup dari pos-4 ke pos-3 adalah 32 karena ia masih dapat melalui *path* [4 - 3]. Jika banyak logistiknya lebih banyak dari itu, maka tidak mungkin ia akan bisa mencapai pos-3 dari pos-4.

Contoh Masukan 2

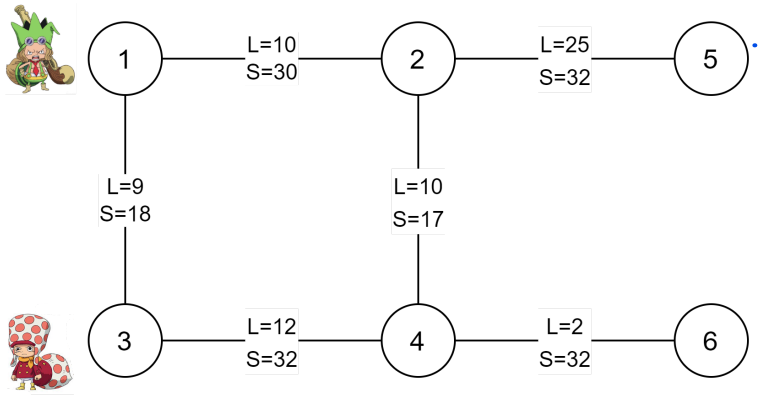
```
6 6
1 2 10 30
1 3 9 18
2 4 10 17
2 5 25 32
3 4 12 32
4 6 2 32
2
1
```

```
3
1
SIMULASI 2 5 6
```

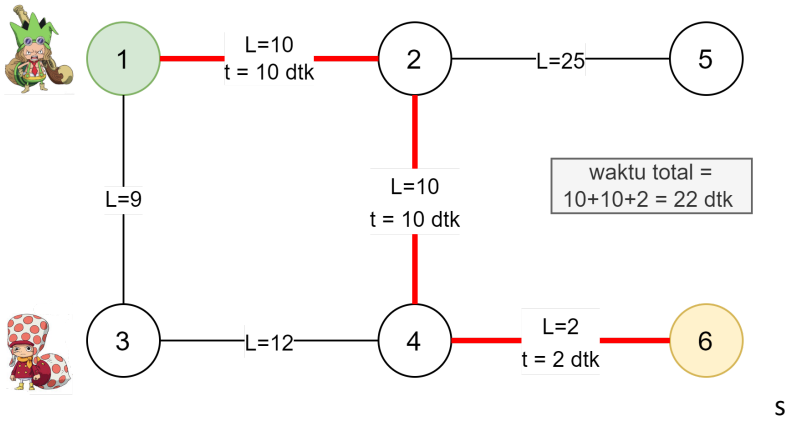
Contoh Keluaran 2

```
22
```

Penjelasan Contoh 2

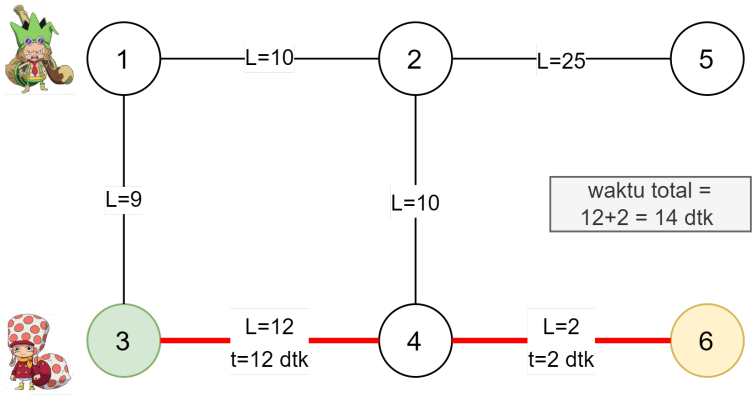


Pos exit terdekat dari kurcaci hijau adalah pos-6 yang dapat ia capai dalam waktu 22 detik melalui *path* [1 - 2 - 4 - 6].



s

Sementara pos exit terdekat dari kurcaci merah adalah pos-6 yang dapat ia capai dalam waktu 14 detik melalui *path* [3 - 4 - 6].



Oleh karena itu, dibutuhkan waktu setidaknya 22 detik agar semua kurcaci dapat keluar dari sistem terowongan sebelum gas beracun diaktifkan.

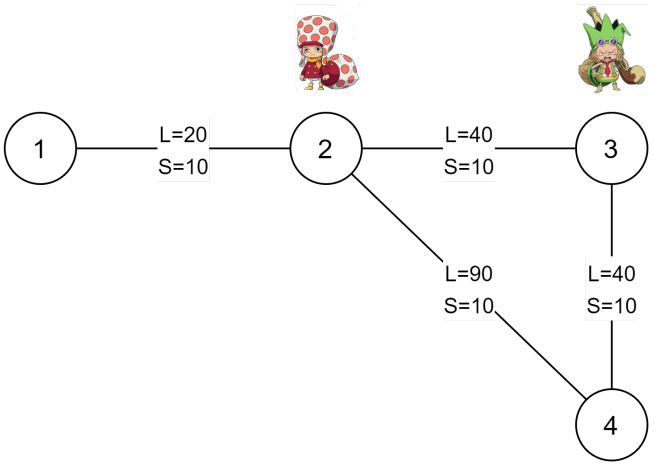
Contoh Masukan 3

```
4 4
1 2 20 10
2 3 40 10
2 4 90 10
3 4 40 10
2
2
3
2
SUPER 1 4 2
SUPER 4 3 1
```

Contoh Keluaran 3

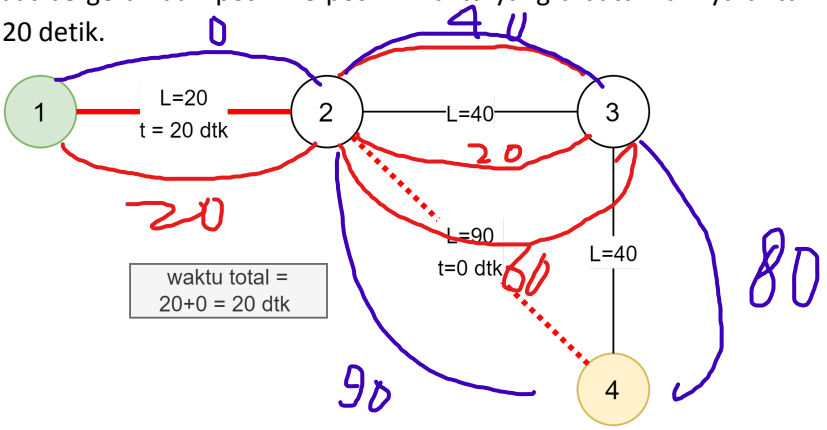
```
100
60
```

Penjelasan Contoh 3

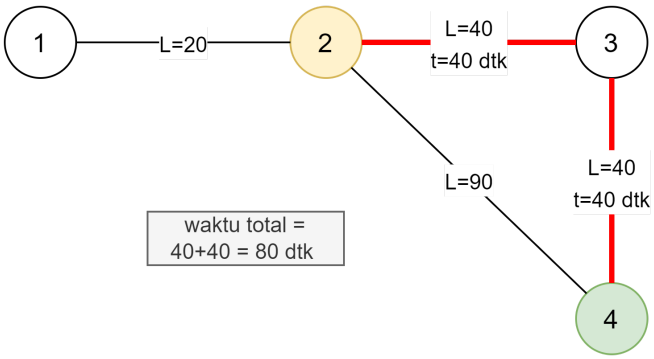


SUPER 1 4 2

Kurcaci super bergerak dari pos-1 ke pos-4 dengan menggunakan *path* [1 - 2 - 4] dan ia menggunakan kekuatan super pada saat bergerak dari pos-2 ke pos-4. Waktu yang dibutuhkannya untuk bergerak melalui path ini adalah 20 detik.



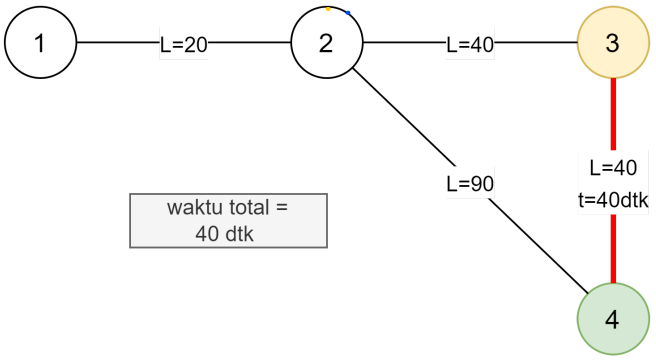
Kemudian untuk bergerak dari pos-4 ke pos-2, kurcaci super bergerak menggunakan *path* [4 - 3 - 2]. Waktu yang dibutuhkannya untuk bergerak melalui path ini adalah 80 detik.



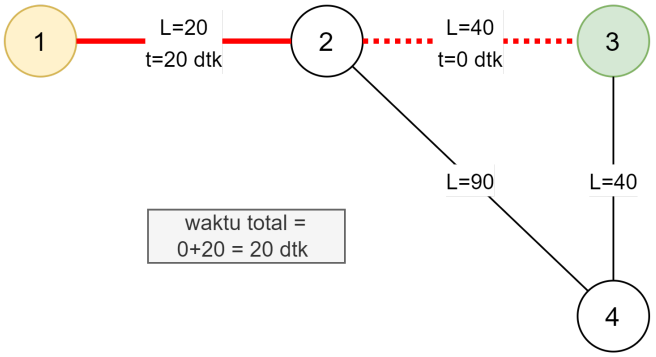
Total waktu yang ia butuhkan untuk menjalankan seluruh proses tersebut adalah 20 + 80 = 100 detik.

SUPER 4 3 1

Kurcaci super bergerak menggunakan *path* [4 - 3]. Waktu yang dibutuhkannya untuk bergerak melalui path ini adalah 40 detik.



Kemudian untuk bergerak dari pos-3 ke pos-1, kurcaci super bergerak menggunakan *path* [3 - 2 - 1] dan ia menggunakan kekuatan super pada saat bergerak dari pos-3 ke pos-2. Waktu yang dibutuhkannya untuk bergerak melalui path ini adalah 20 detik.



Total waktu yang ia butuhkan untuk menjalankan seluruh proses tersebut adalah 40 + 20 = 60 detik.

Keterangan Tambahan

- Struktur data bawaan java yang diperbolehkan pada TP ini hanya **primitive array, ArrayList, Stack, Queue, LinkedList, dan ArrayDeque**.
- Anda bebas untuk memakai seluruh algoritma graf yang sudah atau akan dipelajari di SDA, seperti Dijkstra, Kruskal (dan struktur data union-find), Prim dan DFS/BFS.
- Jika solusi Anda memerlukan *sorting*, Anda wajib mengimplementasikan algoritma *sorting* sendiri.

Penalti 20% akan diberikan jika solusi Anda

- Menggunakan struktur data **HashMap, HashSet, TreeMap, TreeSet, dan PriorityQueue**
- Mengimplementasikan BST sendiri, seperti AVL, Treap, RB-tree
- Menggunakan algoritma atau struktur data **tingkat lanjut** yang belum atau tidak diajarkan pada SDA, seperti Hash Table, Dinic Algorithm, Hungarian Algorithm, dan Ear Decomposition

Penalti 15% akan diberikan jika solusi Anda

- Menggunakan fungsionalitas *sorting* bawaan java, seperti **Collections.Sort**.

Informasi Tambahan *Test Case*

Test Case	*Query*
1-13	KABUR
14-33	SIMULASI
34-55	SUPER
56-65	KABUR, SIMULASI
66-75	KABUR, SUPER
76-85	SIMULASI, SUPER
86-100	KABUR, SIMULASI, SUPER