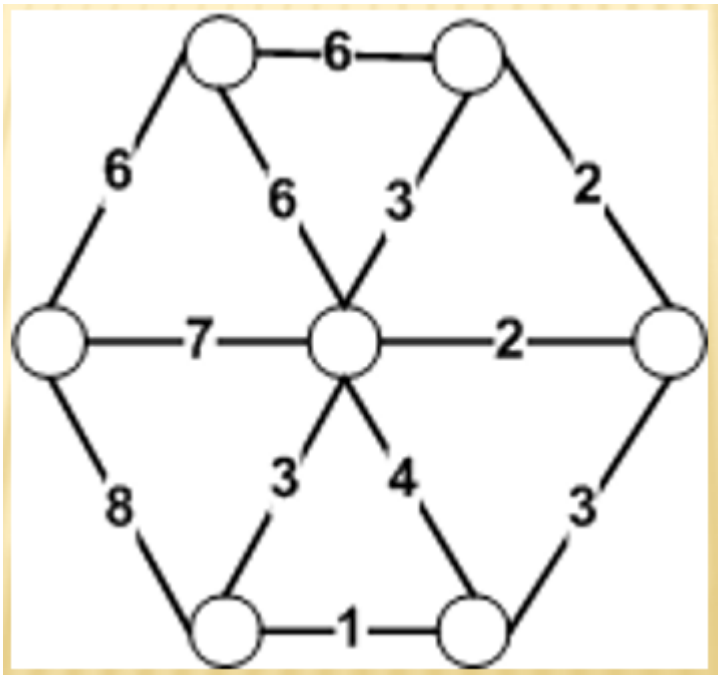
TUGAS MINGGUAN 2

Nama : Muhammad Daffa Rizky Sutrisno

NRP : 5025231207

Graph :



1. Temukan 3 algoritma yang disebutkan pada permasalahan “The Connector Problem” dan aplikasikan pada graph di bawah untuk menemukan spanning tree terkecilnya !

    a. **Prims**

    Algoritma Prim adalah metode untuk menemukan Minimum Spanning Tree (MST) dari sebuah graph berbobot. MST adalah pohon yang mencakup semua vertex dalam graph dengan total bobot edge yang minimum. Berikut adalah langkah-langkah dasar algoritma Prim:

    1. Pilih Vertex Awal: Pilih satu vertex sebagai titik awal MST. Vertex ini bisa dipilih secara acak atau berdasarkan kriteria tertentu.
    2. Inisialisasi: Buat himpunan kosong untuk menyimpan vertex yang sudah termasuk dalam MST dan himpunan untuk vertex yang belum termasuk.
    3. Eksekusi Algoritma:

Selama masih ada vertex yang belum termasuk dalam MST:

a. Pilih Edge Minimum: Pilih edge dengan bobot terkecil yang menghubungkan vertex dalam MST dengan vertex di luar MST.
b. Tambahkan ke MST: Masukkan vertex baru ke dalam MST dan tambahkan edge yang dipilih ke dalam MST.
c. Update Himpunan: Hapus vertex yang baru ditambahkan dari himpunan vertex yang belum termasuk MST.

4. Ulangi langkah 3 hingga semua vertex terhubung dan termasuk dalam MST.

Iterasi 1 :

V1	V2	V3	V4	V5	V6	V7
0	inf	inf	inf	inf	inf	inf

> Pilih V1 sebagai vertex awal

Iterasi 2 :

V1	V2	V3	V4	V5	V6	V7
0	6	8	7	inf	inf	inf

> Pilih (V1,V2) dengam minimum key 6

Iterasi 3 :

V1	V2	V3	V4	V5	V6	V7
0	6	8	6	6	inf	inf

> Pilih (V2,V4) dengam minimum key 6

Iterasi 4 :

V1	V2	V3	V4	V5	V6	V7
0	6	3	6	3	4	2

> Pilih (V4,V7) dengam minimum key 2

Iterasi 5 :

V1	V2	V3	V4	V5	V6	V7
0	6	3	6	2	3	2

> Pilih (V7,V5) dengam minimum key 2

Iterasi 6 :

V1	V2	V3	V4	V5	V6	V7
0	6	3	6	2	3	2

> Pilih (V4,V3) dengam minimum key 3

Iterasi 7 :

V1	V2	V3	V4	V5	V6	V7
0	6	3	6	2	1	2

> Pilih (V3,V6) dengam minimum key 1

Hasil :

Minimum Spawning Tree (MST)

(v1,v2) = 6

(v2,v4) = 6

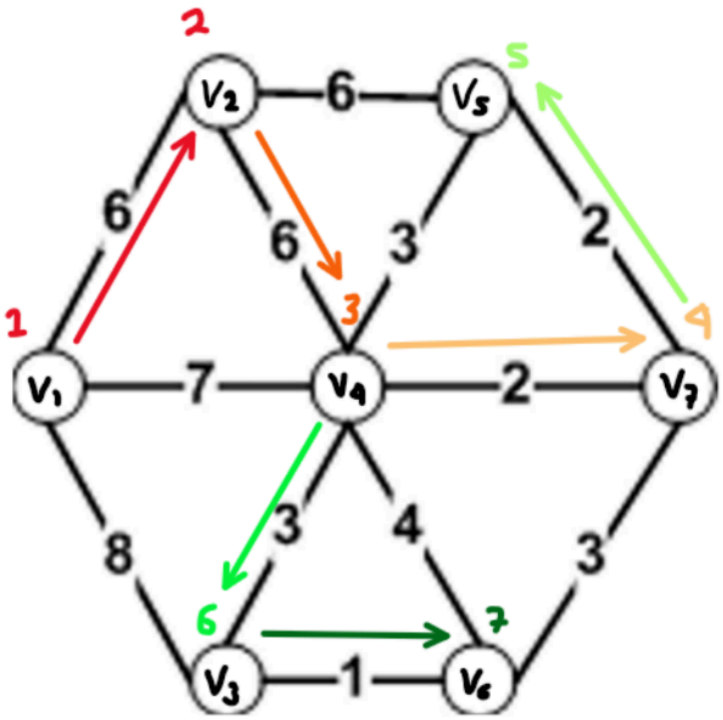
(v4,v7) = 2

(v7,v5) = 2

(v4,v3) = 3

(v3,v6) = 1

Ilustrasi :



b. Kruskall

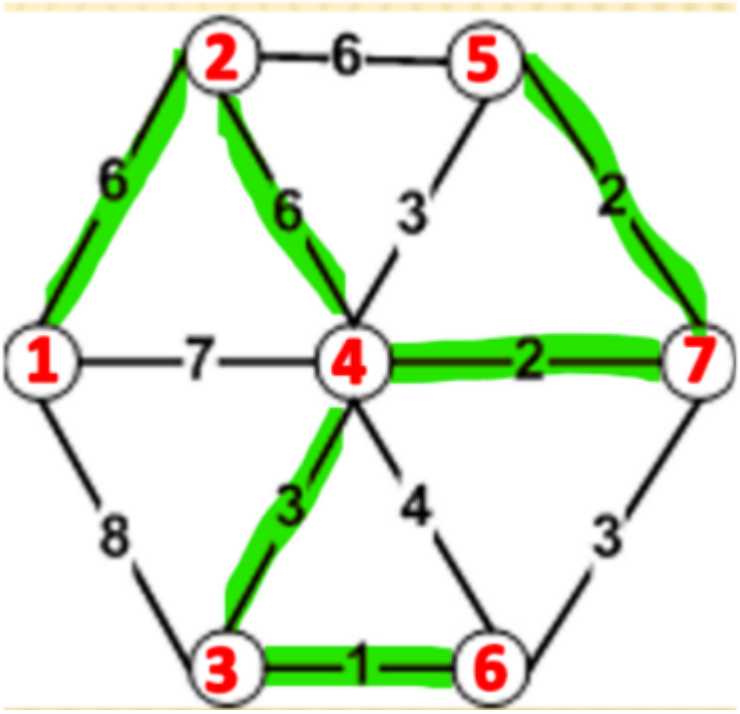
Algoritma Kruskal digunakan untuk menemukan Minimum Spanning Tree (MST) dalam sebuah graf berbobot. MST adalah subgraf yang menghubungkan semua simpul tanpa membentuk siklus, dengan total bobot terkecil. Algoritma Kruskal bekerja dengan memilih edge terkecil dan memastikan tidak ada siklus yang terbentuk. Berikut adalah langkah-langkah singkatnya:

1. Inisialisasi Graf:
    - Inisialisasi graf dengan sejumlah simpul (vertices) dan daftar edge yang terhubung beserta bobotnya.
    - Sort Edge: Urutkan semua edge berdasarkan bobotnya secara tidak menurun (ascending) menggunakan algoritma pengurutan, seperti sorted().
2. Inisialisasi Subset:
    - Setiap simpul awalnya menjadi subset terpisah. Subset ini digunakan untuk melacak apakah dua simpul berada dalam pohon yang sama.
3. Iterasi:
    - Lakukan iterasi melalui edge yang sudah diurutkan: a) Pilih edge dengan bobot terkecil yang belum digunakan. b) Periksa apakah edge ini membentuk siklus jika ditambahkan ke MST yang sedang dibangun. Gunakan fungsi find() untuk memeriksa apakah kedua simpul edge tersebut (u dan v) berada dalam subset yang sama. c) Jika edge tersebut tidak membentuk siklus, tambahkan edge ke MST dan gabungkan kedua subset simpul (u dan v) menggunakan fungsi union().
4. Terminasi:
    - Proses iterasi dilanjutkan hingga MST mencakup semua simpul dalam graf, atau hingga jumlah edge yang dipilih mencapai (jumlah simpul - 1), yang merupakan syarat terbentuknya MST.
5. Hasil:
    - Setelah semua iterasi selesai, hasilnya adalah MST dengan bobot total minimum.
    - Algoritma Kruskal memastikan pengambilan edge-edge dengan bobot terkecil tanpa membentuk siklus, sehingga memberikan MST dengan bobot minimum.

Output :

```
Edges in the constructed MST
3 -- 6 == 1
5 -- 7 == 2
4 -- 7 == 2
3 -- 4 == 3
1 -- 2 == 6
2 -- 4 == 6
Minimum Spanning Tree 20
```

Ilustrasi :



c. Boruvka's Algorithm

Algoritma Boruvka digunakan untuk menemukan Minimum Spanning Tree (MST) dari sebuah graf berbobot. Algoritma ini bekerja dengan cara menggabungkan komponen-komponen (subgraf) secara bertahap hingga seluruh vertex terhubung dalam satu komponen besar. Boruvka sangat efisien untuk graf besar dan dapat diimplementasikan dalam lingkungan komputasi paralel, menjadikannya cocok untuk graf dengan banyak vertex dan edge.

Langkah-langkah Algoritma Boruvka:

1. Inisialisasi:
    - Mulai dengan MST kosong dan setiap vertex dianggap sebagai komponen terpisah. Setiap vertex di graf dianggap sebagai satu komponen sendiri-sendiri.

2. Iterasi:

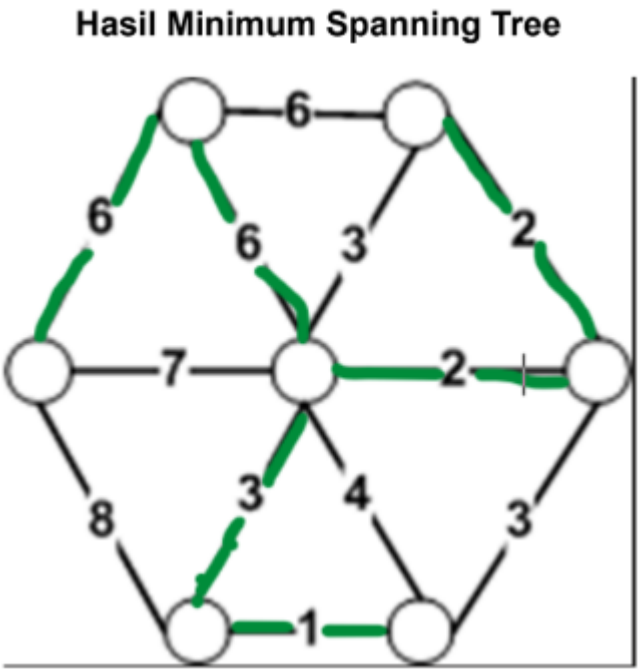
   Lakukan iterasi selama masih ada lebih dari satu komponen dalam graf:

   a. Pilih Edge Terkecil:
      - Untuk setiap komponen, pilih edge terkecil yang menghubungkan komponen tersebut dengan komponen lainnya.
      - Ini dapat dilakukan secara paralel karena setiap komponen diproses secara independen.
   b. Gabungkan Komponen:
      - Tambahkan edge terkecil yang dipilih ke MST, dan gunakan edge tersebut untuk menggabungkan dua komponen menjadi satu komponen yang lebih besar.
   c. Perbarui MST:
      - Edge-edge yang digunakan untuk menggabungkan komponen ditambahkan ke dalam MST.
   d. Perbarui Set Komponen:
      - Setelah penggabungan, komponen-komponen diperbarui. Komponen-komponen baru terbentuk dari hasil penggabungan vertex-vertex yang sebelumnya terpisah.
3. Terminasi:
   - Proses iterasi berlanjut hingga hanya tersisa satu komponen dalam graf, yang berarti semua vertex sudah terhubung.
   - Pada titik ini, MST sudah terbentuk dan algoritma selesai.

Output :

```
Edge 1-2 with weight 6 included in MST
Edge 3-6 with weight 1 included in MST
Edge 4-7 with weight 2 included in MST
Edge 5-7 with weight 2 included in MST
Edge 2-4 with weight 6 included in MST
Edge 3-4 with weight 3 included in MST
Weight of MST is 20
```

Ilustrasi :



2. Carilah sebuah permasalahan riil yang dapat dimodelkan dengan menggunakan tree !

a. **Contoh Permsalahan :** Menentukan rute terpendek dalam peta kota sederhana.
b. **Deskripsi Permasalahan :** Dalam permasalahan menentukan rute terpendek dalam peta kota sederhana, tujuan utamanya adalah mencari jalur paling efisien dari satu lokasi ke lokasi lainnya. Misalnya, Anda ingin mengetahui rute terpendek dari rumah Anda ke toko di kota.
c. **Model Graph :** Pada model graf ini, setiap lokasi pada peta kota, seperti rumah dan toko, akan menjadi vertex dalam graf. Setiap jalan yang menghubungkan lokasi-lokasi ini akan menjadi edge. Misalnya, jika ada satu jalan yang menghubungkan rumah dengan toko, maka graf akan memiliki dua vertex (Rumah dan Toko) dan satu edge (Jalan1) yang menghubungkan keduanya.
d. **Vertex (Simpul) :** Vertex atau simpul dalam graf ini mewakili lokasi-lokasi di peta kota. Dalam contoh ini, setiap lokasi yang penting untuk perjalanan adalah sebuah vertex. Misalnya, jika kita mempertimbangkan peta kota yang hanya memiliki dua lokasi utama, yaitu rumah dan toko, maka vertex 1 = rumah dan vertex 2 = toko. Dalam graf, vertex ini akan menjadi titik-titik di mana rute-rute atau jalan-jalan akan terhubung.
e. **Edge (Sisi) :** Edge atau sisi dalam graf ini mewakili jalan atau rute yang menghubungkan lokasi-lokasi (vertex) satu sama lain. Setiap edge menunjukkan adanya koneksi langsung antara dua vertex. Dalam contoh peta kota sederhana, jika ada satu jalan yang menghubungkan rumah dengan toko, maka edge 1 = Jalan1, yang menghubungkan vertex Rumah dengan vertex Toko.

Dengan model graf ini, kita bisa menerapkan algoritma untuk mencari rute terpendek, seperti **Breadth-First Search (BFS)** jika graf tanpa bobot, atau **Dijkstra's**

Algorithm jika ada bobot yang mewakili jarak atau waktu tempuh antara lokasi-lokasi yang dihubungkan oleh edges. Model graf ini memungkinkan kita untuk memvisualisasikan dan memecahkan masalah rute terpendek secara efisien.

3. Sebuah lahan persawahan memiliki 6 petak sawah yang kesemuanya penuh terisi air. Berapa jumlah minimal pematang yang harus dilubangi agar air dapat terkuras habis dari keenam petak sawah tsb?

Jumlah cabang (branch) dalam sebuah spanning tree adalah v − 1, di mana v adalah jumlah vertex (simpul) dalam graf. Dalam kasus ini, terdapat 10 vertex, sehingga jumlah cabang dalam spanning tree adalah 9. Oleh karena itu, jumlah chord dalam graf tersebut dapat dihitung dengan mengurangkan jumlah cabang dari total edge yang ada. Jika terdapat 15 edge dalam graf secara keseluruhan, maka jumlah chordnya adalah 15 − 9 = 6.

Berikut adalaha visualisasi hasil dari spanning pada graph tersebut menggunakan algoritma prim :