**LAPORAN PRAKTIKUM**

**STRUKTUR DATA DAN ALGORITMA**

**Judul: Algoritma Prim dan Algoritma Kruskal**



**DISUSUN OLEH**

**Ilham Nur Romdoni M0520038**

**PROGRAM STUDI INFORMATIKA**

**FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM**

**UNIVERSITAS SEBELAS MARET**

**2021**

## **BAB I PENDAHULUAN**

1. **Tujuan praktikum**
2. Praktikan dapat mengetahui informasi terkait algoritma Prim dan algoritma Kruskal.
3. Praktikan dapat menganalisis cara kerja dari algoritma Prim dan algoritma Kruskal.
4. Praktikan dapat mengetahui perbedaan dari algoritma Prim dan algoritma Kruskal.
5. **Dasar teori**
6. **Algoritma Prim**

Algoritma Prim adalah sebuah algoritma dalam *graph* *theory* untuk mencari *minimum* *spanning* *tree* untuk sebuah *graph* berbobot yang saling terhubung. Ini berarti bahwa sebuah himpunan bagian dari *edge* yang membentuk suatu *tree* yang mengandung *node*, di mana bobot keseluruhan dari semua *edge* dalam *tree* diminimalisasikan. Bila graph tersebut tidak terhubung, maka *graph* itu hanya memiliki satu *minimum* *spanning* *tree* untuk satu dari komponen yang terhubung. Algoritma ini ditemukan pada 1930 oleh matematikawan Vojtěch Jarník dan kemudian secara terpisah oleh *computer* *scientist* Robert C. Prim pada 1957 dan ditemukan kembali oleh Dijkstra pada 1959. Karena itu algoritma ini sering dinamai algoritma DJP atau algoritma Jarnik.

Dengan struktur data *binary* *heap* sederhana, algoritma Prim dapat ditunjukkan berjalan dalam waktu O(Elog V), di mana E adalah jumlah cabang dan V adalah jumlah *node*. Dengan *Fibonacci* *heap*, hal ini dapat ditekan menjadi O(E + Vlog V), yang jauh lebih cepat bila grafiknya cukup padat sehingga E adalah Ω(Vlog V).

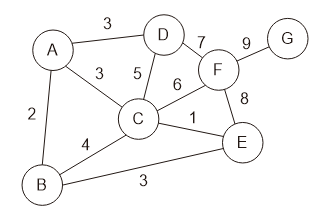
1. **Algoritma Kruskal**

Algoritma Kruskal adalah sebuah algoritma dalam teori *graph* yang mencari sebuah *minimum* *spanning* *tree* untuk sebuah *graph* berbobot yang terhubung. Ini berarti menemukan *subset* dari *edge* yang membentuk sebuah *tree* yang mencakup setiap titik, di mana berat total dari semua *edge* di atas *tree* diminimalkan. Jika *graph* tidak terhubung, maka menemukan *minimum spanning forest* (*minimum spanning tree* untuk setiap komponen terhubung). Algoritma Kruskal juga tergolong algoritma rakus dalam *graph theory* yang digunakan untuk mencari *minimum spanning tree*. Algoritma ini pertama kali muncul pada tahun 1956 dalam sebuah tulisan yang ditulis oleh Joseph Kruskal. Algoritma ini pertama kali muncul dalam *Prosiding* *American* *Mathematical* *Society*, tahun 1956. Dasar pembentukan algoritma Kruskal berasal dari analogi *growing forest.* *Growing* *forest* maksudnya adalah untuk membentuk *minimum* *spanning* *tree* T dari *graph* G adalah dengan cara mengambil satu per satu *edge* dari *graph* G dan memasukkannya ke dalam *tree* yang telah terbentuk sebelumnya. Seiring dengan berjalannya iterasi untuk setiap *edge*, maka *forest* akan memiliki *tree* yang semakin sedikit. Oleh sebab itu, maka analogi ini disebut dengan *growing* *forest*. Algoritma Kruskal akan terus menambahkan semua *edge* ke dalam *forest* yang sesuai hingga akhirnya tidak akan ada lagi *forest* dengan, melainkan hanyalah *minimum* *spanning* *tree*.

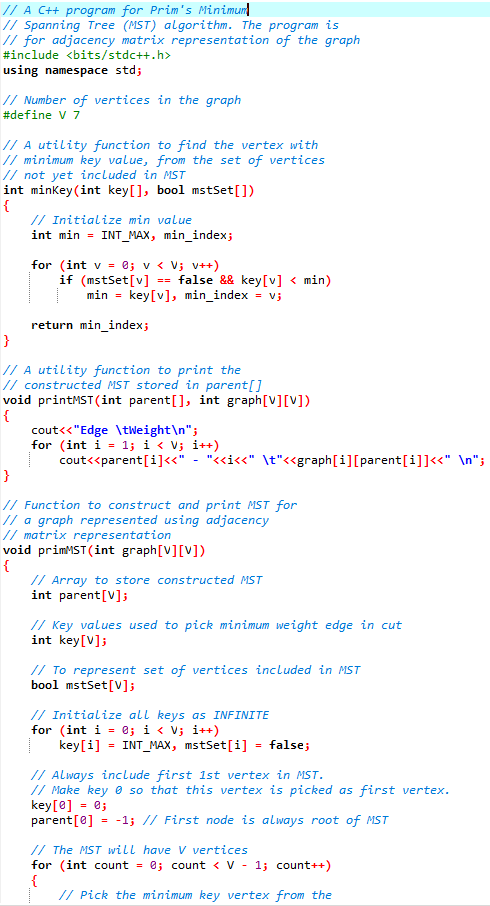
1. **Peralatan/perangkat yang digunakan**
2. Pc/ Laptop 1 Unit.
3. Microsoft Word.
4. CorelDraw.
5. Dev-C++.

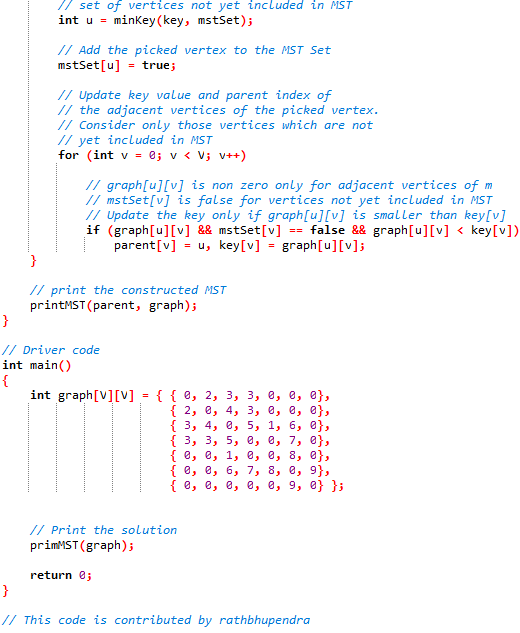
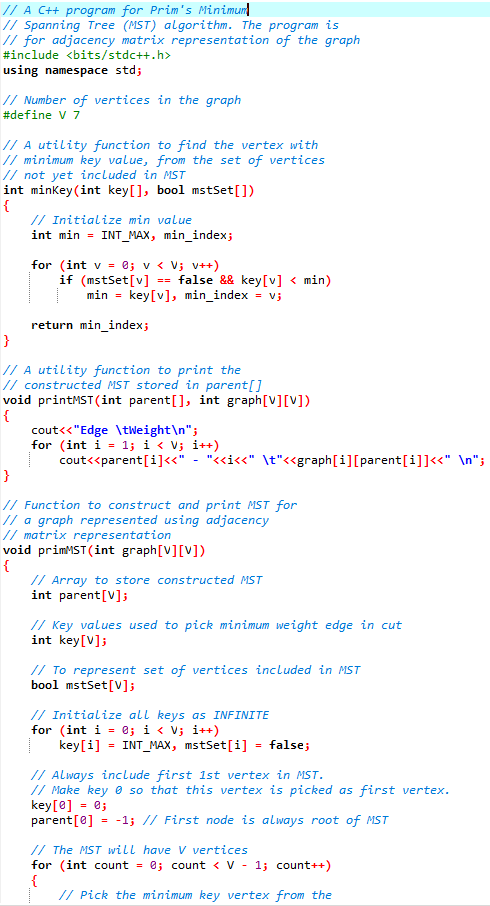
### **BAB II PEMBAHASAN**

Mencari *minimum spanning tree* dari *graph* di bawah ini menggunakan algoritma Prim dan algoritma Kruskal.

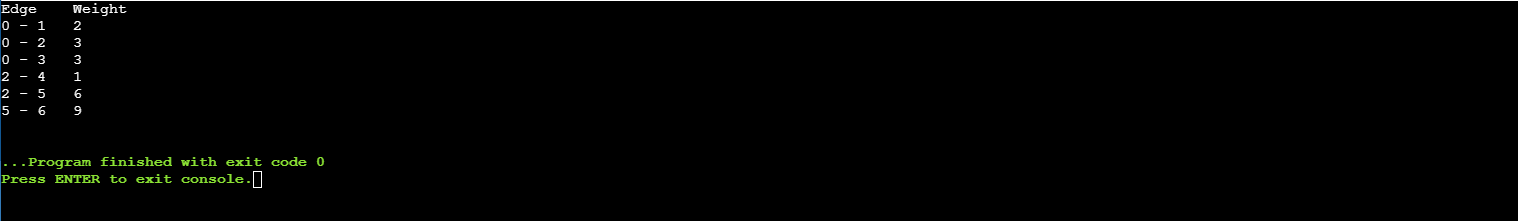


1. **Algoritma Prim**
2. **Source Code**



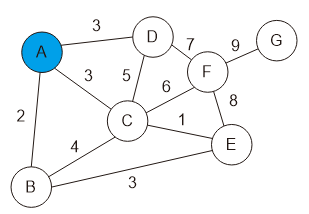


Saat program di atas dijalankan akan memunculkan hasil sebagai berikut.

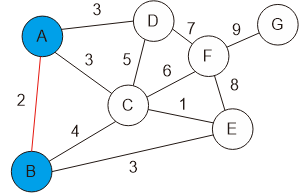


Pada program huruf A sampai G diganti menggunakan angka 0 sampai 6 di mana angka 0 berarti huruf A, angka 1 untuk huruf B, dan berurutan hingga angka 6 yang mewakili huruf G. Hasil dari program di atas menunjukkan bahwa *minimum spanning tree* dari *graph* adalah 2+3+3+1+6+9 =24 yang didapat dengan menjumlahkan semua bobot dari setiap *edge*. Urutan *edge* yang ditampilkan program sama dengan urutan langkah-langkah kerja algoritma Prim.

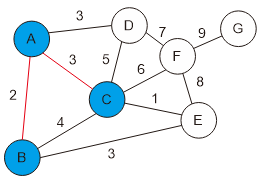
1. **Cara kerja**
2. Memilih *vertex* yang diketahui, *vertex* tersebut adalah A.



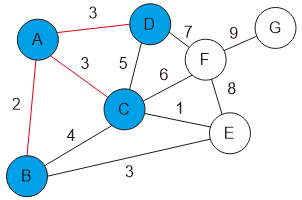
1. Menghubungkan *vertex* dengan lintasan yang paling kecil. Jika dilihat dari gambar di atas, dihubungkan ke B yang bernilai 2.



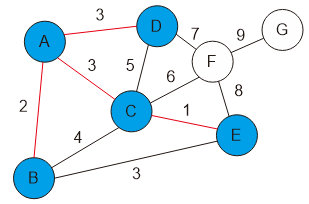
1. Menghubungkan *vertex* dengan lintasan yang paling kecil lainnya. Jika dilihat dari gambar, huruf C bernilai 3 Maka melewati lintasan C.



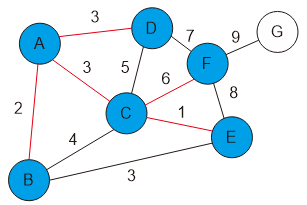
1. Menghubungkan *vertex* dengan lintasan yang paling kecil lainnya. Jika dilihat dari gambar, huruf D bernilai 3. Maka melewati lintasan D.



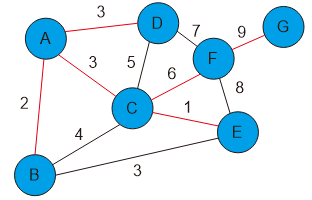
1. Karena *vertex* sudah terhubung dengan semua lintasan yang memungkinkan, langkah selanjutnya adalah menghubungkan lintasan dengan bobot paling kecil yang tidak menimbulkan *cycle*, yaitu lintasan C ke E yang bernilai 1.



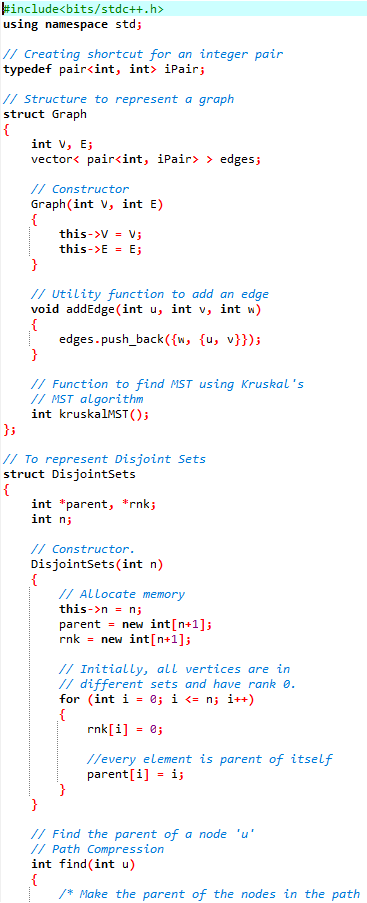
1. Menghubungkan dengan sisa huruf yang belum terhubung yaitu F dan G. Sebelum ke G harus terhubung terlebih dahulu dengan huruf F. Lintasan dengan bobot paling kecil yaitu C ke F dengan nilai 6.

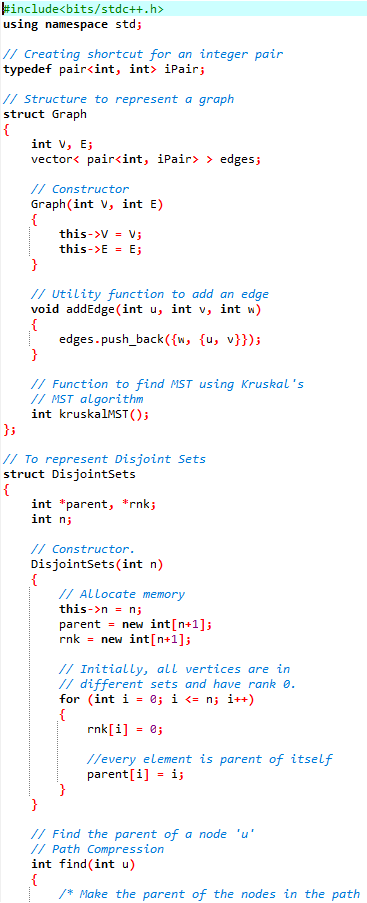


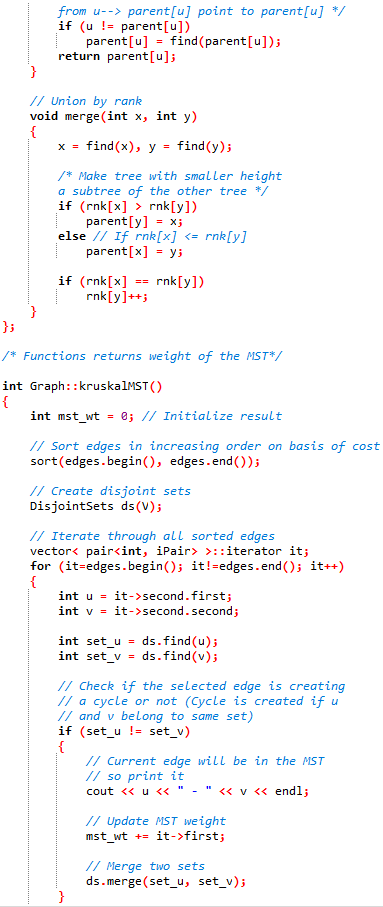
1. Menghubungkan dengan G yang bernilai 9. Pemilihan E yang bernilai 8 hanya akan menghasilkan *cycle*.

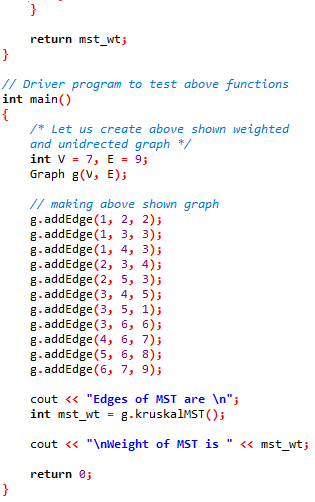
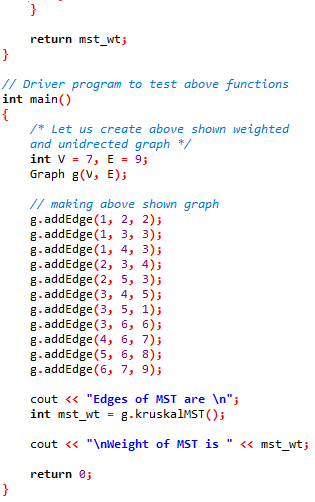
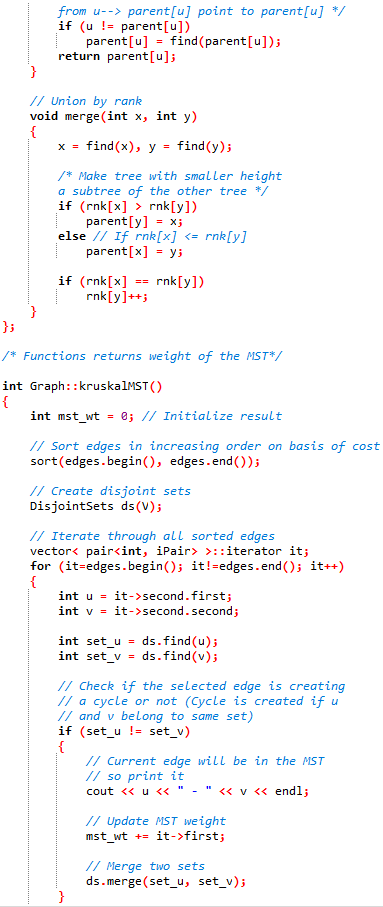


1. Maka dapat diketahui bahwa, nilai dari *minimum* *spanning* *tree* dengan menghitung semua bobot yaitu, 2 + 3 + 3 + 1 + 6 + 9 = 24.
2. **Algoritma Kruskal**
3. **Source Code**

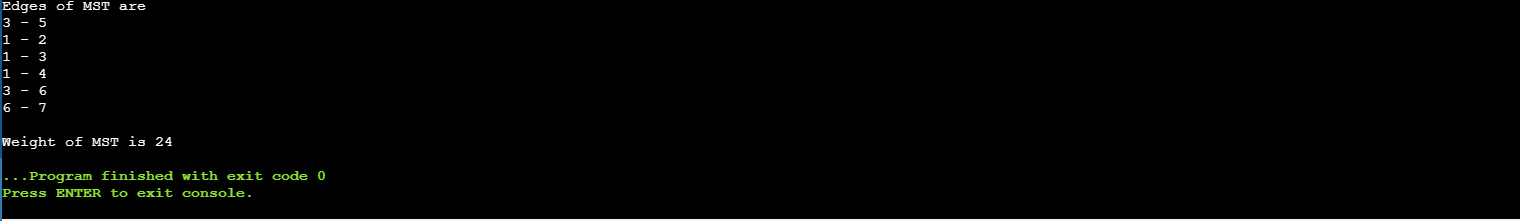






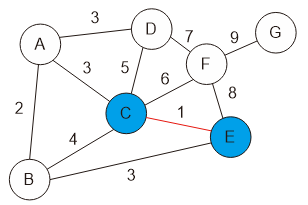


Saat program di atas dijalankan akan memunculkan hasil sebagai berikut.

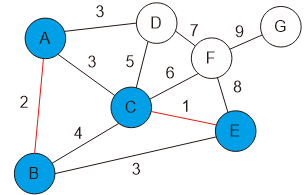


Pada program huruf A sampai G diganti menggunakan angka 1 sampai 7 di mana angka 1 berarti huruf A, angka 2 untuk huruf B, dan berurutan hingga angka 7 yang mewakili huruf G. Hasil dari program di atas menunjukkan bahwa *minimum* *spanning* *tree* dari *graph* adalah 24 yang didapat dengan menjumlahkan semua bobot dari setiap *edge*. Secara berurutan bobot dari *edge* adalah 1, 2, 3, 3, 6, dan 9. Urutan *edge* yang ditampilkan program sama dengan urutan langkah-langkah kerja algoritma Kruskal.

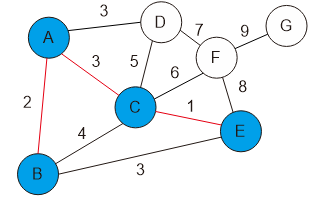
1. **Cara kerja**
2. Memilih *edge* yang memiliki bobot paling kecil yakni C-E yang bernilai 1.



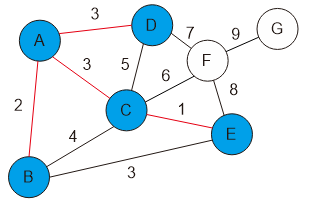
1. Memilih *edge* yang memiliki bobot paling kecil berikutnya, yakni A-B yang bernilai 2.



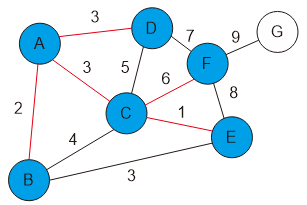
1. Memilih *edge* yang memiliki bobot paling kecil berikutnya yang menghubungkan dua lintasan, yakni jalur A-C dan B-E yang nilainya sama-sama 3. Secara arbiter jalur yang akan dipilih adalah A-C.



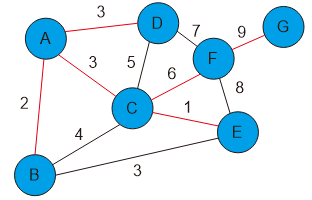
1. Memilih *edge* yang memiliki bobot paling kecil berikutnya, yakni A-D yang bernilai 3.



1. Memilih *edge* yang memiliki bobot paling kecil selain jalur A-C dan B-E, Satu-satunya jalur dengan bobot paling kecil tanpa menimbulkan *cycle* adalah jalur C-F yang bernilai 6.



1. Menghubungkan jalur F-G yang bernilai 9.



1. Maka dapat diketahui bahwa, nilai dari *minimum* *spanning* *tree* dengan menghitung semua bobot yaitu, 1 + 2 + 3 + 3 + 6 + 9 = 24.

# **BAB III PENUTUP**

1. **Kesimpulan**

Dari penghitungan *minimum* *spanning* *tree* di atas dapat diketahui perbedaan dari algoritma Prim dan algoritma Kruskal. Perbedaan tersebut ditampilkan pada tabel di bawah ini.

|  |  |  |
| --- | --- | --- |
| No | Algoritma Prim | Algoritma Kruskal |
| 1 | Penyelesaian masalah dimulai dari *node* | Penyelesaian masalah dimulai dari *edge* |
| 2 | Terbentang dari *node* satu ke *node* lainnya | Memilih posisi *edge* dengan tidak berdasar pada langkah terakhir |
| 3 | *Graph* penyelesaian harus berhubungan | *Graph* penyelesaian dapat terputus |
| 4 | Kompleksitas waktu = O (V2) | Kompleksitas waktu = O (log V) |

1. **Referensi**

Wikipedia. 2021. “Algoritma Prim”, [https://id.wikipedia.org/wiki/Algoritma\_Prim](https://id.wikipedia.org/wiki/Algoritme_Prim), diakses pada 19 Juni 2021 pukul 20.55.

Purbadinata, Rizal Aji. 2021. “Week 11 – Graph 2”, [https://classroom.google.com/ u/1/c/MzExMjIzMTIxODA4/m/MzYxMjYwMDY1MDU0/details](https://classroom.google.com/%09u/1/c/MzExMjIzMTIxODA4/m/MzYxMjYwMDY1MDU0/details), diakses pada 15 Juni 2021 pukul 16.50.

Mustofa, Zaenal. 2017. “Makalah Algoritma Kruskal”,www.slideshare.net/ zaenalmustofa54943/makalah-algoritma-kruskal, diakses pada 19 Juni 2021 pukul 22.36.