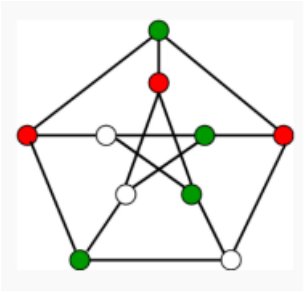
Nama : Maulana Dimas Syahputra

NIM : G.211.22.0104

Penjelasan Algoritma UAS Praktikum 8

1. M Coloring Problem



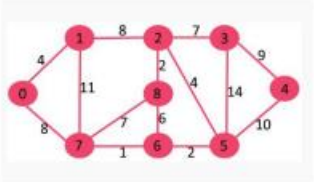
Menggunakan Algoritma Backtracking

Tetapkan warna satu per satu ke simpul yang berbeda, dimulai dari simpul 0 . Sebelum memberi warna, periksa apakah simpul-simpul yang berdekatan mempunyai warna yang sama atau tidak. Jika ada penetapan warna yang tidak melanggar ketentuan, tandai penetapan warna tersebut sebagai bagian dari solusi. Jika tidak ada penetapan warna yang memungkinkan, mundurlah dan kembalikan false.

Ikuti langkah-langkah yang diberikan untuk menyelesaikan masalah:

1. Buat fungsi rekursif yang mengambil grafik, indeks saat ini, jumlah simpul, dan susunan warna.
2. Jika indeks saat ini sama dengan jumlah simpul. Cetak konfigurasi warna dalam susunan warna.
3. Tetapkan warna pada suatu titik dari rentang (1 hingga m).
4. Untuk setiap warna yang ditetapkan, periksa apakah konfigurasinya aman, (yaitu periksa apakah simpul yang berdekatan tidak memiliki warna yang sama) dan panggil fungsi secara rekursif dengan indeks dan jumlah simpul berikutnya jika tidak, kembalikan salah
5. Jika ada fungsi rekursif yang mengembalikan nilai true , maka putuskan loop dan kembalikan nilai true
6. Jika tidak ada fungsi rekursif yang mengembalikan nilai benar, maka kembalikan salah

2. Dijkstra Shortest Path 1



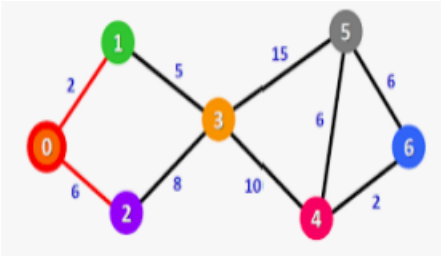
Algoritma Reverse Delete berkaitan erat dengan algoritma Kruskal . Dalam algoritma Kruskal yang kami lakukan adalah : Mengurutkan tepi berdasarkan peningkatan urutan bobotnya. Setelah menyortir, kami memilih tepi satu per satu dalam urutan yang meningkat. Kami menyertakan tepi yang dipilih saat ini jika dengan memasukkan ini ke dalam pohon merentang tidak membentuk siklus apa pun hingga terdapat tepi V-1 di pohon merentang, di mana V = jumlah simpul.

Dalam algoritma Reverse Delete, kami mengurutkan semua sisi berdasarkan urutan bobotnya. Setelah menyortir, kami memilih tepi satu per satu dalam urutan menurun. Kami menyertakan tepi yang dipilih saat ini jika mengecualikan tepi saat ini menyebabkan terputusnya koneksi pada grafik saat ini . Ide utamanya adalah menghapus tepi jika penghapusannya tidak menyebabkan terputusnya grafik.

Algoritma :

1. Urutkan semua tepi grafik dalam urutan bobot tepi yang tidak bertambah.
2. Inisialisasi MST sebagai grafik asli dan hapus tepi tambahan menggunakan langkah 3.
3. Pilih tepi dengan bobot tertinggi dari tepi yang tersisa dan periksa apakah menghapus tepi akan memutus hubungan grafik atau tidak .Jika terputus, maka kami tidak menghapus tepinya.Jika tidak, kami menghapus tepinya dan melanjutkan.

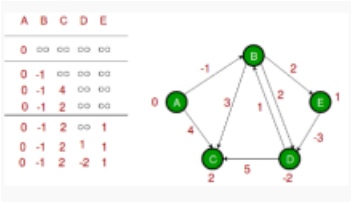
3. Algoritma Dijkstra Shortest Path.



Algoritma ini digunakan untuk menghitung dan mencari jalur terpendek antar node menggunakan bobot yang diberikan dalam grafik. (Dalam jaringan, bobot diberikan oleh paket link-state dan berisi informasi seperti kesehatan router, biaya lalu lintas, dll.).
Penjelasan
Ini dimulai dengan node sumber dan menemukan sisa jarak dari node sumber. Algoritme Dijkstra melacak jarak yang diketahui saat ini dari node sumber ke node lainnya dan secara dinamis memperbarui nilai-nilai ini jika ditemukan jalur yang lebih pendek. Sebuah node kemudian ditandai sebagai telah dikunjungi dan ditambahkan ke jalur jika jarak antara node tersebut dan node sumber adalah yang terpendek. Ini berlanjut sampai semua node telah ditambahkan ke jalur tersebut, dan akhirnya, kita mendapatkan jalur terpendek dari node sumber ke semua node lainnya, yang mana paket-paket dalam jaringan dapat mengikuti ke tujuannya.

Kita memerlukan bobot positif karena bobot tersebut harus ditambahkan ke dalam perhitungan untuk mencapai tujuan kita. Bobot negatif akan membuat algoritma tidak memberikan hasil yang diinginkan.

4. Algoritma Bellman -Ford



Masukan: Graf dan simpul sumber src

Keluaran: Jarak terpendek ke semua simpul dari src . Jika terdapat siklus bobot negatif, maka jarak terpendek tidak dihitung, siklus bobot negatif dilaporkan.

1. Langkah ini menginisialisasi jarak dari sumber ke semua simpul sebagai tak terhingga dan jarak ke sumber itu sendiri sebagai 0. Buatlah array dist[] berukuran |V| dengan semua nilai tak terbatas kecuali dist[src] di mana src adalah titik sumber.
2. Langkah ini menghitung jarak terpendek. Lakukan berikut |V|-1 kali dimana |V| adalah jumlah simpul pada graf tertentu.a) Lakukan hal berikut untuk setiap tepi uv .Jika dist[v] > dist[u] + bobot tepi uv, maka perbarui dist[v] dist[v] = dist[u] + bobot tepi uv
3. Langkah ini melaporkan jika terdapat siklus bobot negatif pada grafik. Lakukan hal berikut untuk setiap sisi uv .Jika dist[v] > dist[u] + bobot sisi uv, maka “Grafik mengandung siklus bobot negatif”
   Ide dari langkah 3 adalah, langkah 2 menjamin jarak terpendek jika grafik tidak mengandung siklus bobot negatif. Jika kita mengulangi semua sisi sekali lagi dan mendapatkan jalur yang lebih pendek untuk setiap titik, maka terdapat siklus bobot negatif.