UTS KB

Nama Kelompok :

1. Alesandro Anggrek - C14190173
2. Christian Willson - C14190178
3. Richardo Jason - C14190181
4. Sutomo Suryanto - C14190184
5. Richson Sedjie - C14190185

Problem : GPS Mobil

Sebuah mobil Smart Car memiliki GPS Mobil. Penumpang menginputkan tujuan-tujuan yang ingin dikunjungi. Dimana GPS akan melakukan pencarian rute terpendek, teraman/tersepi dan tercepat melihat dari traffic yang ada dari posisi mereka sekarang ini. Kemudian nantinya GPS akan menampilkan rute atau jalur-jalur mana saja yang harus dilalui untuk mencapai tujuan-tujuan. Dimana semua tempat harus dikunjungi sekali.

Indikator pengukuran :

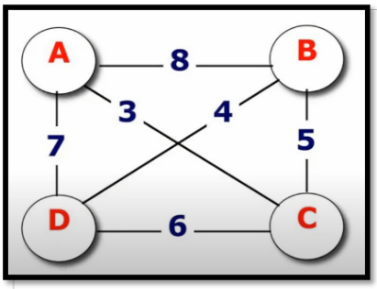
1. Jalan dengan jarak terkecil.

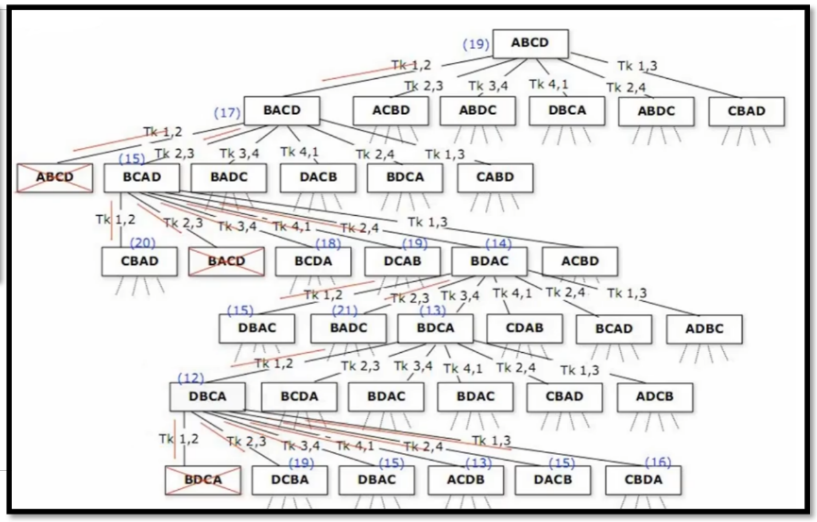
2. Apabila ada 2 jalan dengan tujuan yang sama, jalan yang satunya pendek dan jalan yang kedua cukup panjang dengan keadaan sepi. Maka Sistem akan memilih jalan yang lebih panjang tetapi dengan kondisi yang sepi. Apabila kedua jalanan ramai, maka system akan memilih jalur yang terpendek.

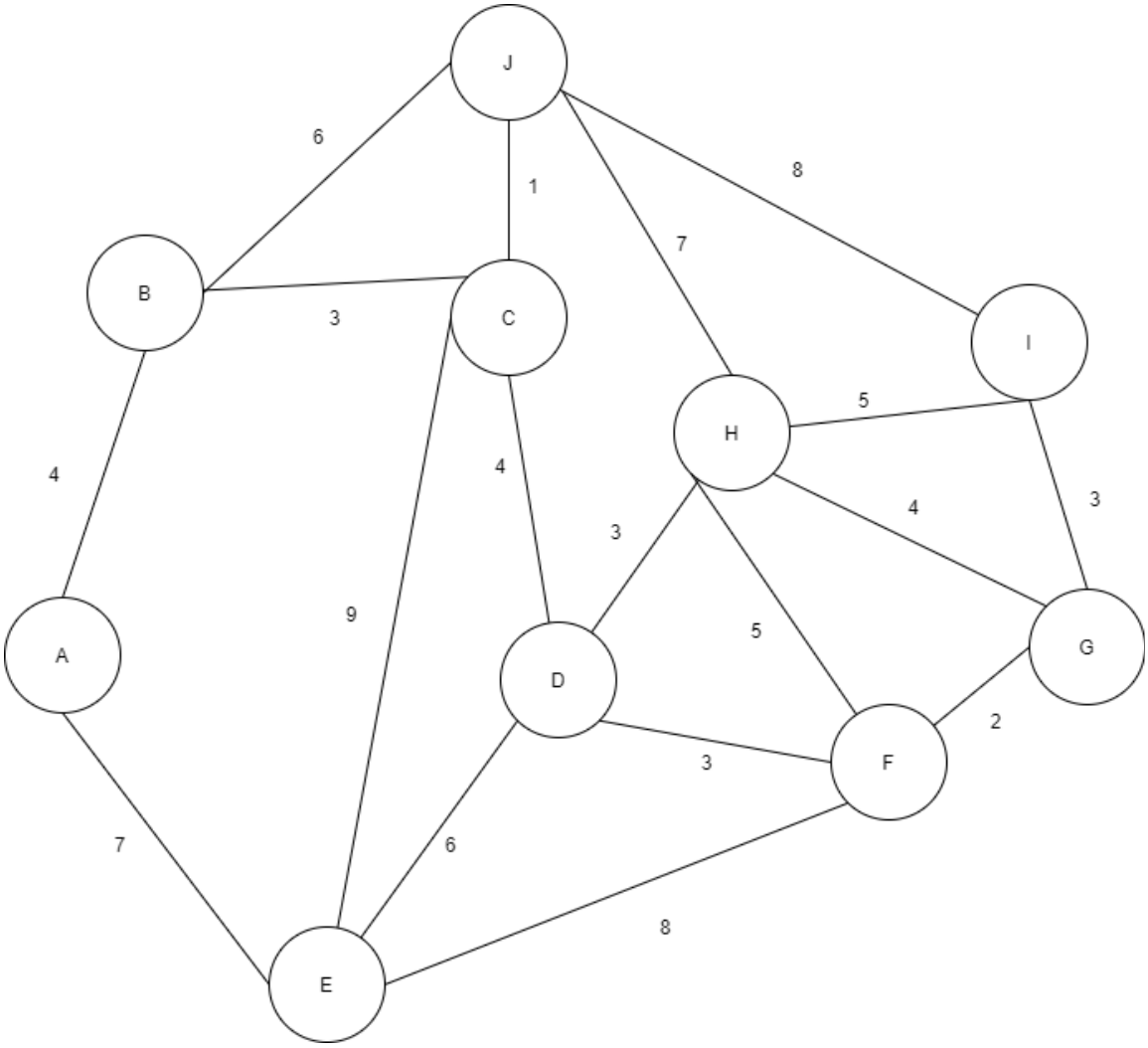
3. Apabila dalam kondisi random / tertentu, tiba-tiba jalannya mengalami perbaikan maka Sistem akan memilih jalan lain.

4. Sistem akan menampilkan rute terbaik dan alternatif yang bisa dipilih.

Contoh Kasus :



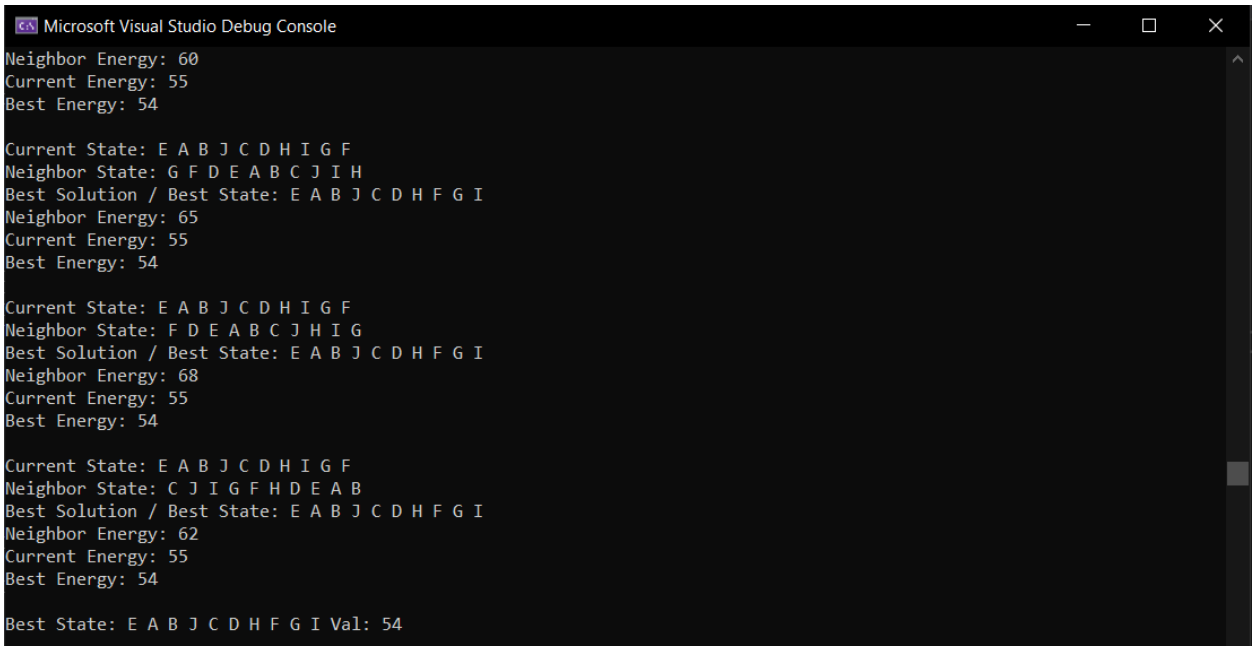




Kendala :

1. Menentukan struktur data yang tepat untuk menyimpan panjang jalan dan tingkat kemacetan. Untuk kendala ini, kelompok kami memilih solusi yaitu menggunakan array of pair. Hal ini dikarenakan data panjang jalan berpasangan dengan data tingkat kemacetan. Sehingga kelompok kami memutuskan bahwa array of pair paling cocok dan mudah untuk diakses.
   Contoh pemanggilan :
   Arr[i][j].first = …
   Arr[i][j].second = ….
2. Alasan kelompok kami menggunakan array untuk menyimpan graph dalam program c++ adalah karena data jalan dan tingkat kemacetan per node itu relatif tetap dalam satu proses.
3. Menggunakan vector karena lebih mudah untuk memasukkan data ataupun menghapus data yang sudah selesai digunakan.
4. Masalah yang kami kerjakan merupakan hasil modifikasi dari kelompok kami sendiri sehingga untuk menguji kebenaran output dari program harus dilakukan berulang kami dan juga menggunakan problem lain yang ada di internet.
5. Simulated annealing walaupun secara konsep mungkin mudah dipahami oleh beberapa orang, tetapi dalam hal pengimplementasian tidak semudah yang dibayangkan. Dikarenakan dalam pengimplementasian, kita harus memilih struktur penyimpanan & teknik memodifikasi *hill climbing* yang tepat untuk mengolah data tersebut.
6. *Simulated annealing* cukup sulit untuk di debug karena adanya variabel random.
7. Kelemahan menggunakan metode *Simulated Annealing* dibandingkan dengan *Hill Climbing* biasa terdapat pada running time. *Simulated Annealing* membutuhkan waktu yang lebih lama. Semakin besar temperatur, maka semakin banyak pula waktu yang dibutuhkan. Tetapi *Simulated Annealing* dapat menghasilkan output yang lebih optimal.
8. Kesulitannya bukan hanya mendapatkan cara random yang tepat agar mendapatkan hasil yang optimal tetapi juga harus melakukan pengecekan jalan/path.

Contoh Output :



Untuk Contoh problem diatas, didapatkan Jalur Terbaik yaitu :
E A B C J H D F G I dengan value 54.

Kesimpulan

Dari percobaan-percobaan yang telah kita lakukan, kelompok kami mendapat kesimpulan bahwa *Hill Climbing* lebih cocok digunakan untuk kasus yang sederhana(dalam kasus kami rute yang simetris dan sedikit). Sedangkan kasus yang lebih kompleks, *Hill Climbing* memiliki peluang yang lebih besar untuk terjebak di local maxima. Problem yang lebih kompleks bisa diselesaikan dengan *Simulated Annealing* dengan menggunakan temperatur yang tinggi dan cooling rate yang rendah. Namun running time dari simulated annealing lebih besar dibandingkan dengan *Hill Climbing*. *Simulated Annealing* melakukan trade-off dari segi waktu untuk mendapatkan hasil yang lebih baik. *Simulated Annealing* membutuhkan lebih sedikit penyimpanan dalam bentuk vector.