Penerapan Algoritma UCS Untuk Menentukan Rute Terbaik Menuju ITB

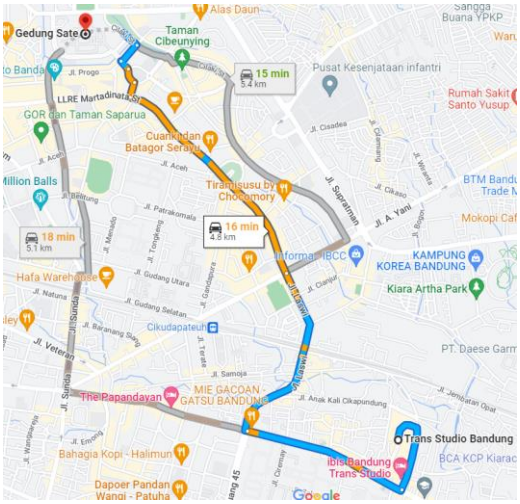
William Manuel Kurniawan - 13520020
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail : 13520020@std.stei.itb.ac.id

***Abstract*—Institut Teknologi Bandung atau dikenal sebagai ITB adalah sebuah universitas yang berada di Kota Bandung bagian utara. Untuk mencapai universitas, terdapat beberapa jalur berbeda yang dapat dipilih oleh mahasiswa. Untuk menghindari keterlambatan, khususnya bagi mahasiswa yang tinggal jauh dari ITB, harus ditentukan rute terbaik yang dapat digunakan. Salah satu cara menentukan rute terbaik tersebut adalah dengan menggunakan algoritma Uniform Cost Search atau dikenal sebagai UCS.**

Keywords—ITB, algoritma UCS, rute terbaik

I. PENDAHULUAN

Dalam daerah kota, sebuah daerah biasanya dihubungkan menggunakan jalan yang dapat diakses oleh masyarakat. Jalan yang dibuat menghubungkan berbagai macam tempat yang ada di kota tersebut sehingga salah satu konsekuensinya adalah adanya lebih dari 1 rute untuk mencapai suatu tempat. Hal ini dapat berfungsi untuk mengurangi kemacetan dengan memberikan lebih banyak akses menuju tempat yang ingin dituju. Untuk menentukan rute yang paling baik, dapat digunakan algoritma penentuan rute. Salah satu algoritma penentuan rute yang dapat digunakan adalah algoritma Uniform Cost Search (UCS). UCS merupakan sebuah algoritma uninformed search yang bekerja menggunakan data berisi simpul serta skor setiap rute antara simpul tersebut. Algoritma UCS akan memeriksa simpul awal yang diinginkan lalu menyimpan data skor dari rute menuju simpul tersebut. Skor bersifat kumulatif sehingga setiap simpul yang diperiksa akan menambahkan skor total yang disimpan oleh program. UCS akan memeriksa simpul berdasarkan skor kumulatif minimal sampai simpul yang diinginkan ditemukan. Contoh penentuan rute terbaik dalam aplikasi dapat ditemukan pada aplikasi peta seperti Google Maps atau dalam video game tertentu yang memiliki daerah permainan yang luas sehingga memerlukan peta.



Gambar 1.1. Gambar Google Maps menentukan rute dari Trans Studio Bandung menuju Gedung Sate

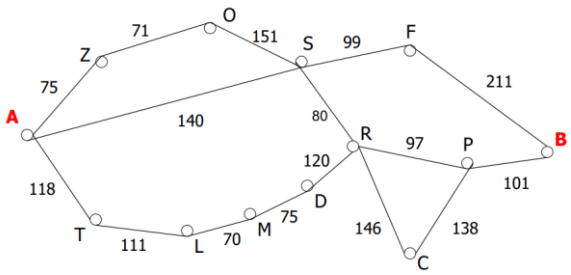
Sumber : Google Maps

Gambar di atas menggambarkan contoh rute-rute yang dapat diambil dari sebuah lokasi ke lokasi lain. Aplikasi Google Maps menggambarkan 3 jalur terbaik yang dapat digunakan dengan jalur terbaik diberi warna. Ketiga rute berawal dari jalan yang sama lalu mulai berubah ketika menemukan jalan lain yang dapat ditempuh.

II. TEORI DASAR

A. *Algoritma Uniform Cost Search*

Algoritma Uniform Cost Search (UCS) merupakan sebuah algoritma uninformed search yang dapat digunakan untuk menentukan rute. Algoritma menerima data berisi simpul data serta rute antara simpul yang diberikan sebuah skor. Simpul-simpul dalam data beserta rutenya dapat digambarkan agar isi dari data lebih mudah dipahami.



Gambar 2.1. Contoh gambar simpul serta rute antara simpul untuk diproses dengan algoritma UCS

Sumber : https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Route-Planning-Bagian1-2021.pdf

Sebelum menggunakan algoritma UCS, pengguna akan menspesifikasikan simpul awal serta simpul yang dicari (pada gambar 2.1, ditandakan dengan simpul yang diberikan warna merah). Algoritma UCS akan memeriksa simpul yang memiliki rute dari simpul awal terlebih dahulu serta skor dari rute simpul tersebut. Untuk menentukan simpul yang akan dicek selanjutnya, dibuat priority queue berdasarkan skor terkecil yang ada. Simpul selanjutnya yang diperiksa diambil dari awal priority queue dan dihapus dari priority queue setelah diperiksa. Simpul tersebut akan diperiksa seperti simpul awal dengan skor yang didapatkan oleh simpul merupakan skor total dari simpul awal sampai simpul tersebut. Simpul-simpul dalam data yang diberikan akan terus ditelusuri berdasarkan priority queue yang dibuat sampai diperiksa simpul yang dicari. Ketika simpul yang dicari ditelusuri dalam oleh program, maka priority queue akan dikosongkan sehingga proses pencarian berhenti.

Simpul-E	Simpul Hidup
A	Z_{A-75}, T_{A-118}, S_{A-140}
Z_{A-75}	T_{A-118}, S_{A-140}, O_{AZ-146}
T_{A-118}	S_{A-140}, O_{AZ-146}, L_{AT-229}
S_{A-140}	O_{AZ-146}, R_{AS-220}, L_{AT-229}, F_{AS-239}, O_{AS-291}
O_{AZ-146}	R_{AS-220}, L_{AT-229}, F_{AS-239}, O_{AS-291}
R_{AS-220}	L_{AT-229}, F_{AS-239}, O_{AS-291}, $P_{ASR-317}$, $D_{ASR-340}$, $C_{ASR-366}$
L_{AT-229}	F_{AS-239}, O_{AS-291}, $M_{ATL-299}$, $P_{ASR-317}$, $D_{ASR-340}$, $C_{ASR-366}$
F_{AS-239}	O_{AS-291}, $M_{ATL-299}$, $P_{ASR-317}$, $D_{ASR-340}$, $C_{ASR-366}$, $B_{ASF-450}$
O_{AS-291}	$M_{ATL-299}$, $P_{ASR-317}$, $D_{ASR-340}$, $C_{ASR-366}$, $B_{ASF-450}$
$M_{ATL-299}$	$P_{ASR-317}$, $D_{ASR-340}$, $D_{ATLM-364}$, $C_{ASR-366}$, $B_{ASF-450}$
$P_{ASR-317}$	$D_{ASR-340}$, $D_{ATLM-364}$, $C_{ASR-366}$, $B_{ASRP-418}$, $C_{ASRP-455}$, $B_{ASF-450}$

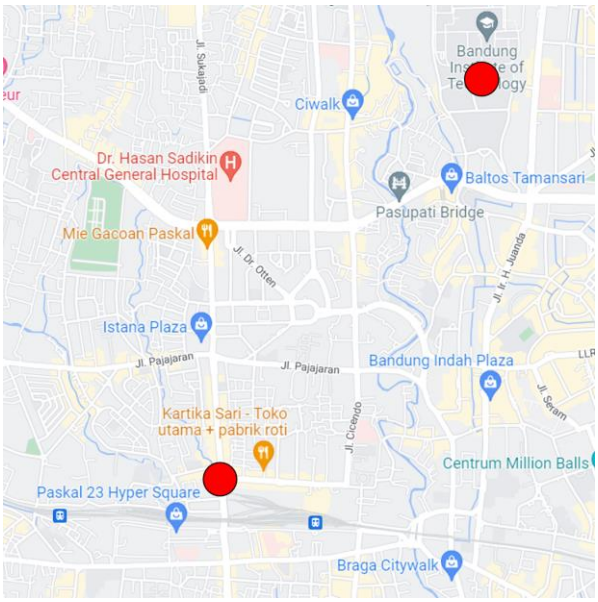
Gambar 2.2. Contoh proses data gambar 2.1 menggunakan algoritma UCS

Sumber : https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Route-Planning-Bagian1-2021.pdf

Algoritma UCS memiliki kelebihan dalam mencari rute karena hasil rute yang dicari akan selalu menjadi hasil yang optimal karena algoritma mencari simpul dengan skor terkecil terlebih dahulu. Kekurangan dari UCS adalah penggunaan memori dan proses yang relatif besar dibandingkan dengan algoritma search yang lain, khususnya algoritma informed search yang dapat menghasilkan rute optimal seperti algoritma UCS dengan proses yang lebih sedikit.

III. Implementasi Algoritma UCS

Untuk menggunakan algoritma UCS, pertama ditentukan bagian jalan yang ingin dianalisis, lokasi yang ingin dituju, serta lokasi awal. Berhubungan dengan tema makalah ini, lokasi yang ingin dituju akan ditetapkan pada Institut Teknologi Bandung (ITB), lebih tepatnya pada bagian gerbang utama ITB. Lokasi awal akan dipilih pada pertigaan di dekat Paskal 23. Berikut adalah gambar peta yang akan digunakan dengan lingkaran merah menandakan lokasi awal dan lokasi tujuan.

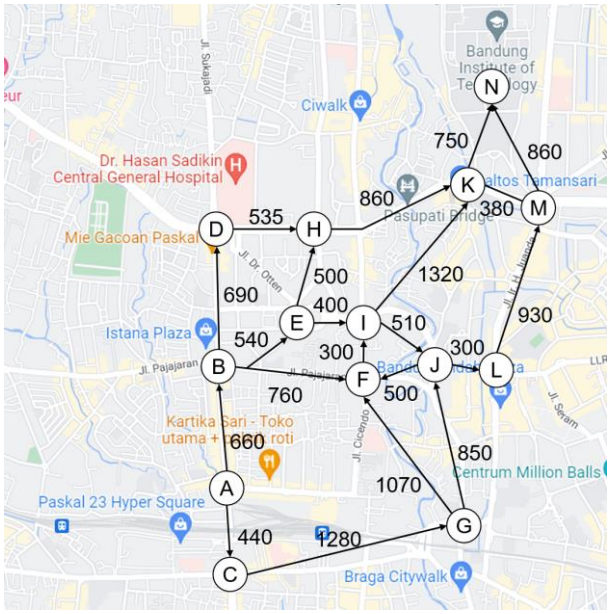


Gambar 3.1. Gambar peta dari Paskal 23 menuju ITB

Sumber : Google Maps

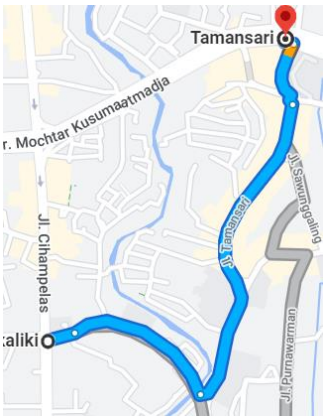
A. *Menentukan Simpul dan Rute Yang Digunakan*

Untuk menentukan simpul yang harus dibuat, dilihat jalan yang dapat ditempuh dalam peta untuk sampai ke ITB. Harus diperhatikan juga arah jalan tersebut agar kendaraan tidak melewati jalan dengan arah yang salah. Dari jalan yang ada, akan dibuat simpul untuk sebagian jalan yang bercabang (biasanya di perempatan atau jalan yang memiliki belokan). Simpul akan dibuat berdasarkan pertimbangan jika jalan yang dapat ditempuh pada simpul tersebut berguna untuk sampai ke simpul tujuan. Untuk menentukan skor bagi rute antara simpul, akan ditentukan jarak antara 2 simpul tersebut menggunakan fitur measure distance dalam aplikasi Google Maps. Dari pertimbangan di atas, maka dihasilkan gambar sebagai berikut.



Gambar 3.2. Gambar peta dari Paskal 23 menuju ITB dengan simpul dan rute yang akan digunakan

Dari gambar 3.2, simpul awal dinyatakan sebagai simpul A dan simpul yang ingin dicari dinyatakan sebagai simpul N. Simpul lain digunakan untuk membantu menggambarkan rute yang dapat ditempuh oleh kendaraan. Perlu dilihat arah panah pada rute antara simpul karena arah panah menunjukkan arah yang dapat digunakan kendaraan. Terdapat rute yang tidak memiliki panah pada rute antara simpul K dan simpul M menunjukkan bahwa kedua arah dapat diambil. Sebagian simpul mungkin tidak terhubung seperti yang diharapkan dengan contoh simpul L dengan simpul G karena arah kendaraan hanya bisa pergi dari simpul L ke simpul G dan arah tersebut dianggap tidak berguna untuk sampai ke simpul N. Simpul I berhubungan dengan simpul K melalui jalan yang kecil dan berbelok-belok tetapi digambarkan secara lurus pada gambar 3.2 agar lebih mudah dilihat. Penyederhanaan juga dilakukan antara simpul K dengan simpul N serta simpul M dengan simpul N. Berikut adalah gambar hubungan antara simpul I dengan simpul K di Google Maps.



Gambar 3.3. Gambar rute peta dari simpul I ke simpul K berdasarkan gambar 3.2

Sumber : Google Maps

Setelah simpul dan rute setiap simpul ditentukan dari bagian peta yang diperiksa, dapat ditentukan rute terbaik dengan algoritma UCS.

B. Proses Algoritma UCS

Ketika menggunakan algoritma UCS, proses algoritma akan ditampilkan seperti gambar 2.2 dengan sebuah table berisi jumlah iterasi, Simpul-E (simpul yang sedang diperiksa), dan simpul hidup. Berikut adalah hasil algoritma UCS dari gambar 3.2.

Iterasi	Simpul-E	Simpul Hidup
1	A	C_{A-440}, B_{A-660}
2	C_{A-440}	B_{A-660}, $G_{AC-1720}$
3	B_{A-660}	$E_{AB-1200}$, $D_{AB-1350}$, $F_{AB-1420}$, $G_{AC-1720}$
4	$E_{AB-1200}$	$D_{AB-1350}$, $F_{AB-1420}$, $I_{ABE-1600}$, $H_{ABE-1700}$, $G_{AC-1720}$
5	$D_{AB-1350}$	$F_{AB-1420}$, $I_{ABE-1600}$, $H_{ABE-1700}$, $G_{AC-1720}$, $H_{ABD-1885}$
6	$F_{AB-1420}$	$I_{ABE-1600}$, $H_{ABE-1700}$, $G_{AC-1720}$, $I_{ABF-1720}$, $H_{ABD-1885}$
7	$I_{ABE-1600}$	$H_{ABE-1700}$, $G_{AC-1720}$, $I_{ABF-1720}$, $H_{ABD-1885}$, $J_{ABEI-2110}$, $K_{ABEI-2920}$
8	$H_{ABE-1700}$	$G_{AC-1720}$, $I_{ABF-1720}$, $H_{ABD-1885}$, $J_{ABEI-2110}$, $K_{ABEH-2560}$, $K_{ABEI-2920}$
9	$G_{AC-1720}$	$I_{ABF-1720}$, $H_{ABD-1885}$, $J_{ABEI-2110}$, $K_{ABEH-2560}$, $J_{ACG-2570}$, $K_{ABEI-2920}$
10	$I_{ABF-1720}$	$H_{ABD-1885}$, $J_{ABEI-2110}$, $J_{ABFI-2230}$, $K_{ABEH-2560}$, $J_{ACG-2570}$, $K_{ABEI-2920}$, $K_{ABFI-3040}$
11	$H_{ABD-1885}$	$J_{ABEI-2110}$, $J_{ABFI-2230}$, $K_{ABEH-2560}$, $J_{ACG-2570}$, $K_{ABDH-2745}$, $K_{ABEI-2920}$, $K_{ABFI-3040}$
12	$J_{ABEI-2110}$	$J_{ABFI-2230}$, $L_{ABEIJ-2410}$, $K_{ABEH-2560}$, $J_{ACG-2570}$, $K_{ABDH-2745}$, $K_{ABEI-2920}$, $K_{ABFI-3040}$
13	$J_{ABFI-2230}$	$L_{ABEIJ-2410}$, $L_{ABFIJ-2530}$, $K_{ABEH-2560}$, $J_{ACG-2570}$, $K_{ABDH-2745}$, $K_{ABEI-2920}$, $K_{ABFI-3040}$
14	$L_{ABEIJ-2410}$	$L_{ABFIJ-2530}$, $K_{ABEH-2560}$, $J_{ACG-2570}$, $K_{ABDH-2745}$, $K_{ABEI-2920}$, $K_{ABFI-3040}$, $M_{ABEIJL-3340}$
15	$L_{ABFIJ-2530}$	$K_{ABEH-2560}$, $J_{ACG-2570}$, $K_{ABDH-2745}$, $K_{ABEI-2920}$, $K_{ABFI-3040}$, $M_{ABEIJL-3340}$, $M_{ABFIJL-3460}$
16	$K_{ABEH-2560}$	$J_{ACG-2570}$, $K_{ABDH-2745}$, $K_{ABEI-2920}$, $M_{ABEHK-2940}$, $K_{ABFI-3040}$, $N_{ABEHK-3310}$, $M_{ABEIJL-3340}$,

		$M_{ABFIJL-3460}$
17	$J_{ACG-2570}$	$K_{ABDH-2745}$, $K_{ABEI-2920}$, $M_{ABEHK-2940}$, $K_{ABFI-3040}$, $N_{ABEHK-3310}$, $M_{ABEIJL-3340}$, $M_{ABFIJL-3460}$
18	$K_{ABDH-2745}$	$K_{ABEI-2920}$, $M_{ABEHK-2940}$, $K_{ABFI-3040}$, $M_{ABDHK-3125}$, $N_{ABEHK-3310}$, $M_{ABEIJL-3340}$, $M_{ABFIJL-3460}$, $N_{ABDHK-3495}$
19	$K_{ABEI-2920}$	$M_{ABEHK-2940}$, $K_{ABFI-3040}$, $M_{ABDHK-3125}$, $M_{ABEIK-3300}$, $N_{ABEHK-3310}$, $M_{ABEIJL-3340}$, $M_{ABFIJL-3460}$, $N_{ABDHK-3495}$, $N_{ABEIK-3670}$
20	$M_{ABEHK-2940}$	$K_{ABFI-3040}$, $M_{ABDHK-3125}$, $M_{ABEIK-3300}$, $N_{ABEHK-3310}$, $M_{ABEIJL-3340}$, $M_{ABFIJL-3460}$, $N_{ABDHK-3495}$, $N_{ABEIK-3670}$, $N_{ABEHKM-3800}$
21	$K_{ABFI-3040}$	$M_{ABDHK-3125}$, $M_{ABEIK-3300}$, $N_{ABEHK-3310}$, $M_{ABEIJL-3340}$, $M_{ABFIJL-3460}$, $N_{ABDHK-3495}$, $N_{ABEIK-3670}$, $N_{ABFIK-3790}$, $N_{ABEHKM-3800}$
22	$M_{ABDHK-3125}$	$M_{ABEIK-3300}$, $N_{ABEHK-3310}$, $M_{ABEIJL-3340}$, $M_{ABFIJL-3460}$, $N_{ABDHK-3495}$, $N_{ABEIK-3670}$, $N_{ABFIK-3790}$, $N_{ABEHKM-3800}$, $N_{ABDHKM-3985}$
23	$M_{ABEIK-3300}$	$N_{ABEHK-3310}$, $M_{ABEIJL-3340}$, $M_{ABFIJL-3460}$, $N_{ABDHK-3495}$, $N_{ABEIK-3670}$, $N_{ABFIK-3790}$, $N_{ABEHKM-3800}$, $N_{ABDHKM-3985}$, $N_{ABEIKM-4160}$
24	$N_{ABEHK-3310}$	Solusi Ditemukan

Dari hasil algoritma UCS pada table di atas, didapatkan rute terbaik A→B→E→H→K→N dengan skor 3310.

C. Pembuatan Algoritma UCS Dalam Komputer

Pembuatan algoritma UCS akan menggunakan bahasa pemrograman Java dan dibuat untuk menyelesaikan permasalahan gambar 3.2 untuk menghasilkan data seperti tabel pada bagian B. Hal pertama yang dilakukan untuk membuat algoritma UCS adalah untuk membuat suatu variabel atau objek yang dapat menampung data nama simpul, hubungan simpul tersebut dengan simpul lainnya, skor total dari penelusuran simpul, dan simpul-simpul yang sudah ditelusuri. Penyimpanan data tersebut akan diterapkan dengan membuat kelas baru pada Java. Berikut adalah contoh kode yang digunakan untuk membuat kelas yang menyimpan data yang disebutkan.

```
public class UCS{
    private String name;
    private ArrayList<Connect> connection;
    private int costFinal;
    private String path;
```

Gambar 3.4. Pembuatan kelas bernama UCS dalam Java

Gambar 3.4 merupakan potongan kode dari kelas UCS yang berisi atribut-atribut dalam kelas tersebut. Atribut *name* merupakan nama simpul yang akan dibuat, atribut *connection* merupakan sebuah arraylist berisi kelas *Connect* yang memuat informasi hubungan simpul tersebut dengan simpul lain, atribut *costFinal* merupakan skor yang disimpan oleh simpul saat penelusuran, dan atribut path merupakan simpul-simpul yang sudah dilewati sebelumnya. Untuk kelas *Connect*, kelas berisi nama simpul yang dituju dan skor ketika menelusuri simpul.

```
public class Connect {
    private String name;
    private int score;
```

Gambar 3.5. Pembuatan kelas bernama UCS dalam Java

Selanjutnya, simpul-simpul yang ada harus dibuat menjadi objek dari kelas UCS dan setiap hubungan yang ada antara simpul-simpul tersebut harus ditambahkan ke dalam objek yang dibuat. Untuk menyimpan data tersebut, dibuat arraylist baru yang menyimpan objek-objek dari kelas UCS.

```
ArrayList<UCS> arrUCS = new ArrayList<UCS>();
arrUCS.add(index: 0, new UCS(name: "A"));
arrUCS.add(index: 1, new UCS(name: "B"));
arrUCS.add(index: 2, new UCS(name: "C"));
arrUCS.add(index: 3, new UCS(name: "D"));
arrUCS.add(index: 4, new UCS(name: "E"));
arrUCS.add(index: 5, new UCS(name: "F"));
arrUCS.add(index: 6, new UCS(name: "G"));
arrUCS.add(index: 7, new UCS(name: "H"));
arrUCS.add(index: 8, new UCS(name: "I"));
arrUCS.add(index: 9, new UCS(name: "J"));
arrUCS.add(index: 10, new UCS(name: "K"));
arrUCS.add(index: 11, new UCS(name: "L"));
arrUCS.add(index: 12, new UCS(name: "M"));
arrUCS.add(index: 13, new UCS(name: "N"));
arrUCS.get(index: 0).setConnection(name: "B", cost: 660);
arrUCS.get(index: 0).setConnection(name: "C", cost: 440);
arrUCS.get(index: 1).setConnection(name: "D", cost: 690);
arrUCS.get(index: 1).setConnection(name: "E", cost: 540);
arrUCS.get(index: 1).setConnection(name: "F", cost: 760);
arrUCS.get(index: 2).setConnection(name: "G", cost: 1280);
arrUCS.get(index: 3).setConnection(name: "H", cost: 535);
arrUCS.get(index: 4).setConnection(name: "H", cost: 500);
arrUCS.get(index: 4).setConnection(name: "I", cost: 400);
arrUCS.get(index: 5).setConnection(name: "I", cost: 300);
arrUCS.get(index: 6).setConnection(name: "F", cost: 1070);
arrUCS.get(index: 6).setConnection(name: "J", cost: 850);
arrUCS.get(index: 7).setConnection(name: "K", cost: 860);
arrUCS.get(index: 8).setConnection(name: "J", cost: 510);
arrUCS.get(index: 8).setConnection(name: "K", cost: 1320);
arrUCS.get(index: 9).setConnection(name: "L", cost: 300);
arrUCS.get(index: 9).setConnection(name: "F", cost: 500);
arrUCS.get(index: 10).setConnection(name: "M", cost: 380);
arrUCS.get(index: 10).setConnection(name: "N", cost: 750);
arrUCS.get(index: 11).setConnection(name: "M", cost: 930);
arrUCS.get(index: 12).setConnection(name: "K", cost: 380);
arrUCS.get(index: 12).setConnection(name: "N", cost: 860);
```

Gambar 3.6. Kode untuk membuat simpul dan hubungan antara simpul tersebut

Objek UCS baru dengan nama yang berbeda ditambahkan ke dalam arraylist *arrUCS* lalu data UCS dalam *arrUCS* diambil dan digunakan perintah setConnection untuk menambah data dalam atribut connection.

Setelah membuat data simpul, dibuat sebuah arraylist sebagai priority queue untuk simpul hidup dan arraylist untuk menyimpan simpul E yang sudah diperiksa. Priority queue awal akan diisi oleh simpul awal (dalam kasus gambar 3.2, simpul awal adalah simpul A). Disiapkan juga variabel untuk menyimpan berapa iterasi yang dilakukan.

```
ArrayList<String> checked = new ArrayList<>();
ArrayList<UCS> queue = new ArrayList<>();

//menambahkan simpul A ke queue awal
queue.add(arrUCS.get(index: 0));
int iterasi = 0;
```

Gambar 3.7. Kode pembuatan variabel untuk algoritma UCS

Algoritma UCS akan memeriksa simpul dari priority queue sampai simpul yang diinginkan ditemukan. Ketika simpul tersebut ditemukan, maka program akan mengosongkan queue dan pemeriksaan simpul tidak dilakukan lagi.

```
while(queue.size() != 0){
    iterasi++;
    if (queue.get(index: 0).getName() == "N"){
        System.out.println(queue.get(index: 0).getName());
        System.out.println(queue.get(index: 0).getCostFinal());
        System.out.println(queue.get(index: 0).getPath());
        queue = new ArrayList<>();
    }
```

Gambar 3.8. Kode algoritma UCS memeriksa jika simpul yang diinginkan sudah ditemukan

Jika simpul belum ditemukan, maka program akan memeriksa setiap hubungan yang disimpan pada objek UCS dalam atribut connection. Jika simpul yang diperiksa dalam hubungan belum ada dalam arraylist simpul E yang sudah diperiksa, maka akan dibuat salinan dari simpul tersebut dengan skor diatur dari simpul E yang diperiksa dengan skor menuju simpul yang berhubungan lalu dimasukkan ke dalam priority queue

```
for (int i = 0; i<queue.get(index: 0).getConnection().size(); i++){
    Boolean isFound = false;
    for (int j = 0; j<checked.size(); j++){
        if (queue.get(index: 0).getConnection().get(i).getName() == checked.get(j)){
            isFound = true;
        }
    }
    if (isFound == false){
        ArrayList<UCS> tempArrUCS = new ArrayList<>();
        for (int k = 0; k<arrUCS.size(); k++){
            UCS tempUCS = new UCS(arrUCS.get(k));
            tempArrUCS.add(tempUCS);
        }

        for (int k = 0; k<arrUCS.size(); k++){
            String connectName = queue.get(index: 0).getConnection().get(i).getName();
            if (arrUCS.get(k).getName() == connectName){
                UCS tempUCS = tempArrUCS.get(k);
                int tempScore = queue.get(index: 0).getConnection().get(i).getScore()
                              + queue.get(index: 0).getCostFinal();

                tempUCS.setCostFinal(tempScore);
                tempUCS.setPath(queue.get(index: 0).getPath() + queue.get(index: 0).getName());
                queue.add(tempUCS);
                break;
            }
        }
    }
}
```

Gambar 3.9. Kode algoritma UCS memeriksa simpul E

Program akan menghasilkan data iterasi yang dilakukan, simpul E yang diperiksa, dan simpul hidup yang sudah diurutkan dalam priority queue sampai simpul yang diinginkan ditemukan seperti pada table di bagian B.

```
iterasi 1
Simpul E: A
Simpul hidup:
C A-440
B A-660
```

Gambar 3.10. Contoh hasil iterasi algoritma UCS dalam Java

Jika simpul yang diinginkan ditemukan, akan mengeluarkan data nama simpul akhir, skor akhir, dan simpul-simpul yang dilewati.

```
iterasi 24
Simpul akhir: N
Skor akhir: 3310
Path: ABEHK
```

Gambar 3.11. Hasil akhir algoritma UCS dalam Java

Dari gambar 3.2, skor yang digunakan dalam rute antara simpul adalah jarak dari 1 simpul ke simpul yang berhubungan. Hasil algoritma UCS dari gambar tersebut merupakan jalur terbaik yang dapat digunakan berdasarkan jarak antara simpul awal dan simpul akhir. Permasalahan datang ketika ingin menentukan jalur terbaik berdasarkan kategori lain seperti jalur tercepat antara simpul karena waktu tempuh dari 1 simpul ke simpul lain dapat berubah. Contoh perubahan waktu terjadi ketika adanya kemacetan pada jalur tertentu sehingga jarak antara simpul bukan satu-satunya penentu skor pada rute yang ada dalam gambar. Untuk mengubah skor dalam rute pada gambar 3.2 sehingga menghasilkan jalur tercepat, harus ditentukan jarak rute dan kecepatan rata-rata kendaraan saat itu pada rute tersebut sehingga bisa diketahui waktu tempuhnya.

IV. KESIMPULAN

Algoritma Uniform Cost Search merupakan algoritma searching bertipe uninformed search yang digunakan untuk menentukan hasil optimal suatu pencarian. Pencarian dilakukan dengan mengecek simpul data beserta dengan skor yang diperlukan ke simpul lain berdasarkan urutan skor akumulatif terkecil sampai simpul yang diinginkan ditemukan. Algoritma UCS dapat digunakan dalam permasalahan di dunia nyata yaitu untuk menentukan jalur terbaik dalam peta dari 1 tempat ke tempat lainnya.

Implementasi algoritma UCS dilakukan dengan menentukan simpul-simpul yang ada antara 2 tempat serta hubungan antara simpul untuk melihat kemungkinan jalur yang dapat diambil oleh pengguna. Setelah itu, ditentukan skor untuk hubungan simpul tersebut berdasarkan informasi yang tersedia seperti jarak dan kecepatan rata-rata lalu disesuaikan dengan kebutuhan pengguna. Terakhir, algoritma UCS dijalankan untuk melihat jalur yang dihasilkan serta skor yang diberikan untuk jalur tersebut.

PRANALA

Kode lengkap yang dibahas pada makalah dapat ditemukan di https://github.com/wmk567/Algoritma-UCS.git

Link video Youtube dapat ditemukan di https://youtu.be/EGWl3EZLrFs

UCAPAN TERIMA KASIH

Penulis mengucapkan syukur kepada Tuhan Yang Maha Esa atas rahmat-Nya dalam pembuatan makalah "Penerapan Algoritma UCS Untuk Menentukan Rute Terbaik Menuju ITB" sehingga dapat diselesaikan dengan baik. Penulis juga mengucapkan terima kasih kepada tim pengajar mata kuliah IF2211 Strategi Algoritma atas bimbingannya selama kuliah. Terakhir, penulis mengucapkan terima kasih kepada orang tua dan teman penulis yang memberi semangat selama kuliah ini.

REFERENCES

[1] Maulidevi, Nur Ulfa. 2021. "Penentuan Rute Bagian 1". https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Route-Planning-Bagian1-2021.pdf. Diakses pada 19 Mei 2022

[2] Munir, Rinaldi, dan Nur Ulfa Maulidevi. 2021. "Penentuan Rute Bagian 2". https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Route-Planning-Bagian1-2021.pdf. Diakses pada 19 Mei 2022

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 20 Mei 2022

William Manuel Kurniawan
13520020