Pencarian Rute Angkutan Kota Bandung Berdasarkan Jarak Terdekat Dengan Memanfaatkan Algoritma Dijkstra

Safiq Faray - 13519145
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail (gmail): 13519145@std.stei.itb.ac.id

***Abstrak*—Angkot merupakan salah satu bentuk kendaraan umum yang tersedia meluas di seluruh kota. Kurangnya kaum modern dalam memanfaatkan jenis transportasi ini dikarenakan kurangnya pengetahuan rute dari setiap jenis angkot. Untuk menyelesaikan masalah ini, dapat digunakan algoritma Dijkstra dalam menentukan rute angkot dengan jarak terpendek.**

Keywords—angkutan kota, kendaraan umum, algoritma greedy, algoritma Dijkstra.

I. PENDAHULUAN

Angkutan kota atau angkot adalah salah satu bentuk transportasi umum yang mudah dijumpai dan relatif murah untuk digunakan di hamper seluruh kota. Angkot biasanya beroperasi dengan trayek antar 2 destinasi bolak-balik dan memiliki rute yang telah ditentukan, sehingga saat melakukan perjalanan dengan angkot dapat turun di sebuah rute dan menunggu angkot lain yang melewati rute sama. Hal tersebut biasa disebut "oper angkot".

Akhir-akhir ini, telah melonjak popularitas dari penggunaan ojek online atau ojol. Ojek online yang popular di Indonesia contohnya adalah Go-jek dan Grab. Ojek online biasanya dipesan lewat aplikasi dan memiliki kemudahan dalam menentukan tujuan apapun dan penentuan jalur serta harganya. Kemudahan yang disediakan oleh ojek online tersebutlah yang membuat kaum zaman sekarang lebih memilih menggunakan layanan tersebut daripada angkot yang jauh lebih murah dan juga tersedia dimana-mana.

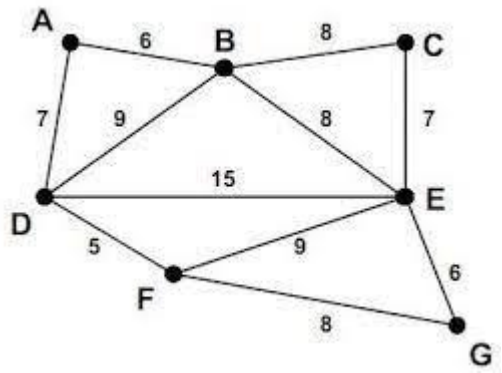
Terdapat banyak angkot dan telah diketahui jelas rute yang dilewatinya. Oleh karena itu, untuk memudahkan pencarian angkot yang akan dinaiki, dapat digunakan algoritma Dijkstra dalam penentuan rute paling dekat beserta angkot yang dinaiki.

II. LANDASAN TEORI

A. *Graf Berbobot*

Graf adalah sebuah data yang merepresentasikan objek-objek diskrit dan hubungan antara objek-objek tersebut [1][3]. Graf terdiri dari simpul, yang merupakan representasi objek, dan sisi, yang menghubungkan satu objek dengan objek-objek lainnya.

Graf berbobot adalah graf yang pada setiap sisinya memiliki bobot. Bobot ini berupa angka pada umumnya.



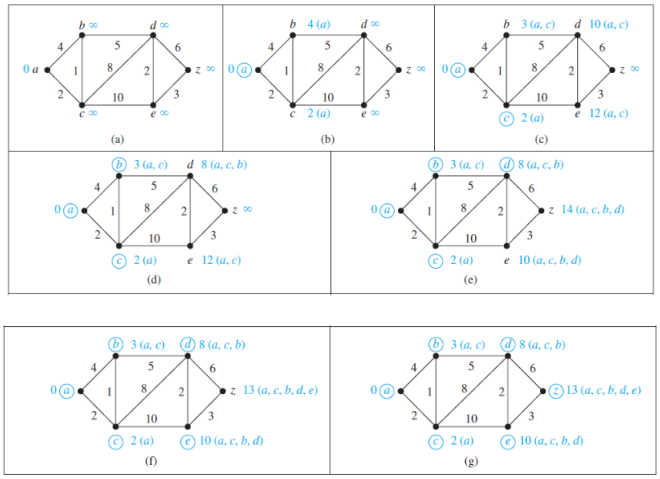
Gambar 2.1 Grapf Berbobot

Sumber : https://eprints.uny.ac.id/28787/2/c.BAB%20II.pdf

B. *Algoritma Dijkstra*

Algoritma Dijkstra merupakan bentuk dari algoritma greedy yang telah dioptimalkan. Algoritma ini merupakan algoritma yang optimal dalam menentukan lintasan terpendek. Lintasan terpendek dibangun langkah per langkah. Pada langkah pertama bangun lintasan terpendek pertama, pada langkah kedua bangun lintasan terpendek kedua, demikian seterusnya [2].

Pada setiap langkah, dipilih lintasan berbobot minimum yang menghubungkan simpul yang sudah terpilih dengan sebuah simpul lain yang belum terpilih. Lintasan dari simpul asal ke simpul yang baru haruslah merupakan lintasan yang terpendek diantara semua lintasannya ke simpul-simpul yang belum terpilih.



Gambar 2.2 Algoritma Dijkstra

Sumber :
https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Greedy-(2021)-Bag2.pdf

C. *Angkutan Kota Bandung*

Terdapat banyak sekali angkot di Kota Bandung dengan variasi trayek dan rute perjalanan. Pada kasus ini, penulis akan mendata data angkot di Kota Bandung yang terdiri dari 3 bagian : nama angkot, rute, dan jarak antar rute dalam kilometer. Pendataan ini dilakukan di dalam file *csv*.

```
File  Edit  Format  View  Help
Dago-Pasar Induk Caringin       Terminal Dago - Jl Cigadung Raya        2.2
Dago-Pasar Induk Caringin       Jl Cigadung Raya - Jl Cikutra Barat     2.5
Dago-Pasar Induk Caringin       Jl Cikutra Barat - Jl Pahlawan  1.3
```

III. Implementasi Algoritma

Dalam penerapan algoritma ini, penulis akan menjabarkan struktur data, algoritma, dan contoh aplikasi dari algoritma yang telah diimplementasikan. Data rute angkot diambil dari situs resmi Dinas Perhubungan Jawa Barat (http://dishub.jabarprov.go.id).

Dalam makalah ini, penulis akan mengambil contoh 2 angkot dengan rute yang berbeda dalam rangka mendemostrasikan kegunaan dari algoritma yang akan diimplementasikan.

A. *Struktur Data*

Struktur data yang akan digunakan dalam implementasi algoritma ini adalah graf berbobot. Implementasi graf berbobot ini adalah dengan pembuatan node dan setiap node menyimpan alamat yang menunjuk ke node tujuannya.

Node sendiri adalah tipe data layaknya merupakan *Linked List*, yang dimana setiap elemennya akan menunjuk ke alamat elemen setelahnya. Node ini juga menyimpan informasi nama jalan dan angkot-angkot yang melaluinya.

Angkot-angkot yang melalui sebuah jalan yang disimpan pada sebuah node akan disimpan dengan tipe data *List of Nodes*.

B. *Data Rute Angkutan Kota*

Data angkutan kota disimpan dalam file *comma separated values* (*csv*) yang terdiri dari nama angkot, rute, dan jarak. Jarak didapatkan dari *google maps*.

```
Nama    Rute    Jarak
Dago-Pasar Induk Caringin       Terminal Dago - Jl Cigadung Raya        2.2
Dago-Pasar Induk Caringin       Jl Cigadung Raya - Jl Cikutra Barat     2.5
Dago-Pasar Induk Caringin       Jl Cikutra Barat - Jl Pahlawan  1.3
Dago-Pasar Induk Caringin       Jl Pahlawan - Jl Surapati (Suci)        1
Dago-Pasar Induk Caringin       Jl Surapati (Suci) - Jl Cikapayang      1.7
Dago-Pasar Induk Caringin       Jl Cikapayang - Jl Tamansari    1.3
Dago-Pasar Induk Caringin       Jl Tamansari - Jl Sawunggaling  0.6
Dago-Pasar Induk Caringin       Jl Sawunggaling - Jl Rangga Gading      0.13
Dago-Pasar Induk Caringin       Jl Rangga Gading - UNISBA & UNPAS (Tamansari)   0.6
Dago-Pasar Induk Caringin       UNISBA & UNPAS (Tamansari) - Jl Wastukencana    1.5
Dago-Pasar Induk Caringin       Jl Wastukencana - Jl Purnawarman        0.45
Dago-Pasar Induk Caringin       Jl Purnawarman - Jl Pajajaran   3.1
Dago-Pasar Induk Caringin       Jl Pajajaran - Jl Cicendo       0.45
Dago-Pasar Induk Caringin       Jl Cicendo - Jl Rivai   1.2
Dago-Pasar Induk Caringin       Jl Rivai - Jl Pasir Kaliki      1.1
Dago-Pasar Induk Caringin       Jl Pasir Kaliki - Jl Pajajaran  2.1
Dago-Pasar Induk Caringin       Jl Pajajaran - Jl Arjuna        0.65
Dago-Pasar Induk Caringin       Jl Arjuna - Jl Supadio  0.65
Dago-Pasar Induk Caringin       Jl Supadio - Jl Ciroyom 1.3
Dago-Pasar Induk Caringin       Jl Ciroyom - Jl Rajawali Timur  1
Dago-Pasar Induk Caringin       Jl Rajawali Timur - Jl Kebon Jati       1.4
Dago-Pasar Induk Caringin       Jl Kebon Jati - Jl Waringin     2
Dago-Pasar Induk Caringin       Jl Waringin - Jl Sudirman       2.2
Dago-Pasar Induk Caringin       Jl Sudirman - Jl Jamika 2
Dago-Pasar Induk Caringin       Jl Jamika - Jl Terusan Jamika   0.7
Dago-Pasar Induk Caringin       Jl Terusan Jamika - Jl Sukamulya        1.9
Dago-Pasar Induk Caringin       Jl Sukamulya - Jl Sukarno Hatta 10
Dago-Pasar Induk Caringin       Jl Sukarno Hatta - Jl Babakan Ciparay   9.4
Dago-Pasar Induk Caringin       Jl Babakan Ciparay - Pasar Induk Caringin (Sukarno-Hatta)       1.5
```

Gambar 3.1 Rute angkot Dago-Pasar Induk Caringin

```
Sadang Serang-Caringin  Terminal Caringin -Jl Caringin  1.4
Sadang Serang-Caringin  Jl Caringin -Jl Holis   2.5
Sadang Serang-Caringin  Jl Holis -Jl Bojong Raya        0.85
Sadang Serang-Caringin  Jl Bojong Raya -Jl Sudirman     8.3
Sadang Serang-Caringin  Jl Sudirman -Jl Rajawali Barat  6.8
Sadang Serang-Caringin  Jl Rajawali Barat -Jl Garuda    0.55
Sadang Serang-Caringin  Jl Garuda -Jl Abdul Rahman Saleh        1
Sadang Serang-Caringin  Jl Abdul Rahman Saleh -Jl Pajajaran     1.1
Sadang Serang-Caringin  Jl Pajajaran -Jl Pandu  0.4
Sadang Serang-Caringin  Jl Pandu -Jl Radjiman   1.2
Sadang Serang-Caringin  Jl Radjiman -Jl Rivai   0.65
Sadang Serang-Caringin  Jl Rivai -Jl Wastukencana       0.75
Sadang Serang-Caringin  Jl Wastukencana -Jl Tamansari   1.4
Sadang Serang-Caringin  Jl Tamansari -Jl Ganesha        0.5
Sadang Serang-Caringin  Jl Ganesha -Jl Ir H Juanda (Dago)       1.8
Sadang Serang-Caringin  Jl Ir H Juanda (Dago) -Jl TB Ismail     0.75
Sadang Serang-Caringin  Jl TB Ismail -Jl Sadang Serang  0.95
Sadang Serang-Caringin  Jl Sadang Serang -Terminal Sadang Serang        0.55
```

Gambar 3.2 Rute angkot Sadang Serang-Caringin

C. *Algoritma Dijkstra*

Pseudocode dari algoritma Dijkstra adalah sebagai berikut.

```
procedure Dijkstra (input G: weighted_graph, input a: intial_vertex, output L : array [1..n] of real)
{ Mencari lintasan terpendek dari simpul a ke semua simpul lain di dalam graf berbobot G.
Masukan: graf-berbobot yang terhubung, G = (V, E) dengan |V| = n
Luaran: L[1..n], L[i] berisi panjang terpendek dari simpul a ke simpul v_i }

Deklarasi:
   i : integer
   u, v : vertex
   S : set of vertex    { himpunan solusi untuk mencatat simpul-simpul yang sudah dipilih di dalam tur }
Algoritma
   for i←1 to n
      L(v_i) ← ∞
   endfor
   L(a) ← 0    { jarak dari a ke a adalah 0 }
   S ← { }
   for k ← 1 to n  do
      u ← pilih simpul yang belum terdapat di dalam S dan memiliki L(u) minimum
      S ← S ∪ {u}    { masukkan u ke dalam S }
      for semua simpul v yang tidak terdapat di dalam S
          { update jarak yang baru dari a ke v }
          if L(u) + G(u, v) < L(v) then    { jarak dari a ke u ditambah bobot sisi dari u ke v lebih kecil dari jarak a ke v }
             L(v) ← L(u) + G(u, v)    { jarak dari a ke v yang baru diganti dengan L(u) + G(u, v) }
          endif
      endfor
   enfor
```

Gambar 3.3 Pseudocode Algoritma Dijsktra

Identify applicable sponsor/s here. If no sponsors, delete this text box (*sponsors).*

Sumber :
https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Greedy-(2021)-Bag2.pdf

Penulis menerapkan algoritma tersebut ke dalam Bahasa Java. Implementasi dari graf dan node adalah sebagai berukut.

```java
public class Node {
    public String name;
    public Map<Node, Float> next;
    public List<String> daftarAngkot;

    public Node(){
        this.name="";
        this.next = new HashMap<>();
        this.daftarAngkot = new ArrayList<>();
    }
    public Node(String name,String angkot){
        this.name = name;
        this.next = new HashMap<>();
        this.daftarAngkot = new ArrayList<>();
        this.daftarAngkot.add(angkot);
    }

    public void tambahAngkot(String angkot){
        if (!this.daftarAngkot.contains(angkot))
            this.daftarAngkot.add(angkot);
    }

    public void tambahNext(Node next, float jarak){
        this.next.put(next,jarak);
    }
```

Gambar 3.4. Tipe data Node

```java
public class Graph {
    public List<Node> nodes = new ArrayList<>();

    public Graph(String filename) throws IOException {
        String line;
        boolean firstLine = true;
        boolean skipFirstLine = true;
        File angkot = new File(filename);
        FileReader fileReader = new FileReader(angkot);
        BufferedReader br = new BufferedReader(fileReader);
        ArrayList<String[]> list = new ArrayList<String[]>();

        while ((line = br.readLine()) != null) {
            if (!firstLine || !skipFirstLine) {
                String[] row = line.split( regex: "\t");
                // This code will ignore double quotes, if present as first and last character
                for (int i = 0; i < row.length; i++) {
                    String val = row[i];
                    if (val.length() >= 2 && val.charAt(0) == '"' && val.charAt(val.length() - 1) == '"') {
                        row[i] = val.substring(1, val.length() - 1);
                    }
                }
                list.add(row);
            }
            firstLine = false;
```

Gambar 3.5 Tipe data Graph

```java
public PathResult getShortestPath(Node src){
    Map<Node, Float> distance = new HashMap<>();
    Map<Node, Boolean> sptSet = new HashMap<>();
    Map<Node, List<Node>> path = new HashMap<>();
    for (Node n : this.graph.nodes){
        distance.put(n, Float.MAX_VALUE);
        sptSet.put(n, false);
        path.put(n, new ArrayList<>(Arrays.asList(src)));
    }
    distance.put(src, (float) 0);

    for (Node n : this.graph.nodes){
        Node u = minDistance(distance, sptSet);
        sptSet.put(u, true);
        for (Node n2 : this.graph.nodes){
            float v = u.next.get(n2) == null? 0 : u.next.get(n2);
            if (!sptSet.get(n2) && v!=0 && Float.compare(distance.get(u), Float.MAX_VALUE)!=0
                    && distance.get(u) + v < distance.get(n2)){
                distance.put(n2, distance.get(u) + u.next.get(n2));
                path.get(n2).add(u);
            }
        }
    }

    return new PathResult(distance, path);
```

Gambar 3.6 Penerapan algoritma Dijkstra

Pada program yang ditulis, belum dibuat sebuah input agar hasil yang dikeluarkan fleksibel. Program yang dibuat hanya untuk demonstrasi algoritma semata.

IV. PEMBAHASAN

A. *Pengujian*

Pengujian program ini dilakukan dengan skenario pengguna sedang berada di Terminal Dago dan hendak menuju Jl. Ganesha. Program akan menghitung jarak terdekat dan mencari rute berdasarkan jarak yang telah ditentukan. Dari terminal Dago ke Jl. Ganesha—karena pada awalnya memang tidak searah—jarak yang akan ditempuh dengan angkot adalah sekitar 10.5 km, dan rute yang akan diambil akan berputar-putar, yaitu mulai Terminal Dago hingga Kembali ke Taman Sari. Dari Taman Sari, oper ke angkot Sadang Serang-Caringin untuk menuju ke Jl. Ganesha.

```
[Terminal Dago , Jl Tamansari ]
10.5
Tujuan : Jl. Ganesha
Pemberhentian : Terminal Dago
Pemberhentian : Jl Tamansari
Oper angkot : [Sadang Serang-Caringin]

Process finished with exit code 0
```

Gambar 4.1. Hasil Program

Untuk lebih jelasnya, berikut adalah kode program yang ditulis untuk pengujian.

```java
public class Main {
    public static void main(String[] args) throws IOException {
        Graph g = new Graph( filename: "./src/angkot.txt");

        Djikstra djikstra = new Djikstra(g);
        PathResult res = djikstra.getShortestPath(g.getNode( name: "Terminal Dago "));
        System.out.println(res.path.get(g.getNode( name: "Jl Ganesha ")));
        System.out.println(res.distance.get(g.getNode( name: "Jl Ganesha ")));
        //Contoh : tujuan Jl. Taman Sari
        System.out.println("Tujuan : Jl. Ganesha");
        for (Node n : res.path.get(g.getNode( name: "Jl Ganesha "))){
            System.out.println("Pemberhentian : "+ n.name);
            if (n.next.containsKey(g.getNode( name: "Jl Ganesha "))){
                System.out.println("Oper angkot : " + g.getNode( name: "Jl Ganesha ").daftarAngkot);
            }
        }
```

Gambar 4.2 Kode pengujian

B. Pembahasan

Program yang dibuat akan membaca semua data rute angkot di file “angkot.txt” pada objek “Graph”. Kemudian, dari pembacaan tersebut akan dilakukan pembentukan objek “Node” berdasarkan pemberhentian angkot.

```java
        if (searchNode(rute[1])){
            //kalau 2-2 nya ada, maka update node yg ada aja
            getNode(rute[0]).tambahAngkot(row[0]);
            getNode(rute[1]).tambahAngkot(row[0]);
            getNode(rute[0]).tambahNext(getNode(rute[1]),Float.parseFloat(row[2]));
        }
        else{
            //kalau n2 belum ada, maka buat baru
            n2 = new Node(rute[1], row[0]);
            n2.tambahAngkot(row[0]);
            getNode(rute[0]).tambahNext(n2,Float.parseFloat(row[2]));
            nodes.add(n2);
        }
    }
    else if (searchNode(rute[1])){
        //kalau cuma n2 yg ada, maka kebalikannya
        n1 = new Node(rute[0], row[0]);
        n1.tambahAngkot(row[0]);
        getNode(rute[1]).tambahNext(n1,Float.parseFloat(row[2]));
        nodes.add(n1);
    }
    else{
        //gaada dua dua nya
        n1 = new Node(rute[0], row[0]);
        n2 = new Node(rute[1], row[0]);
        n1.tambahNext(n2, Float.parseFloat(row[2]));
        n1.tambahAngkot(row[0]);
        n2.tambahAngkot(row[0]);
```

Gambar 4.3 Pembentukan objek Node

Node-node yang terbuat akan disimpan pada objek Graph dalam bentuk List. Setelah itu, akan diterapkan algoritma Dijkstra pada node-node yang telah dibuat

Pada penerapan algoritma dijkstra, juga memanfaatkan tipe data *HashMap*, hal ini mempermudah dalam pengelolahan data jarak antar sebuah Node dengan Source. Data jarak tersebut akan disimpan pada *HashMap* bernama “distance”. Setiap pencarian jarak terdekat dari node-node yang belum dikunjungi, juga akan dicatat Node mana saja yang telah dikunjungi sambil memperbarui nilai distance pada node tersebut. Node yang dikunjungi ini akan disimpan pada *HashMap* path, yang mengandung Node sebagai key dan List of Nodes sebagai value-nya.

```java
Map<Node, Float> distance = new HashMap<>();
Map<Node, Boolean> sptSet = new HashMap<>();
Map<Node, List<Node>> path = new HashMap<>();
```

Gambar 4.4 Pemanfaatan *HashMap*

V. Kemungkinan Perbaikan

Penyusunan strategi algoritma beserta implementasi yang dibuat oleh penulis jauh dari sempurna. Hal yang dapat diperbaiki dari implementasi yang dibuat adalah dengan mencatat rute angkot yang jauh lebih detail, karena implementasi yang sekarang hanyalah mencatat rute sampai dengan pemberhentian dan/atau pergantian jenis angkot.

Selain itu, bisa juga diperbaiki dengan cara pencatatan nama jalan beserta angkot yang menuju ke jalan tersebut. Implementasi yang sekarang sudah cukup dalam konteks menyimpan data saja, tetapi tidak cukup bagus untuk diolah datanya.

VI. Kesimpulan

Graf memiliki banyak kegunaan dan salah satu kegunaan pentingnya adalah representasi terhadap sesuatu. Representasi suatu data dalam bentuk graf dapat memudahkan kita dalam memahami hubungan satu data dengan data lainnya. Graf yang berbobot juga bermanfaat dalam merepresentasikan—salah satunya adalah—hubungan antara lokasi yang dimana bobotnya berupa jarak.

Hal yang telah dibuat oleh penulis juga membuktikan bahwa algoritma-algoritma yang telah dipelajari dapat mempermudah orang-orang dalam berbagai hal, salah satunya adalah penggunaan angkutan kota yang makin lama makin dilupakan.

VII. . Ucapan Terima kasih

Pertama-tama, puji syukur saya panjatkan kepada Allah SWT, yang senantiasa memberikan saya kesehatan untuk menuntut ilmu dan juga memberikan kesempatan untuk menyelesaikan makalah ini. Selain itu, saya berterima kasih kepada dosen strategi algoritma kelas 01, Ibu Masayu Leylia Khodra, yang senantiasa mengajar kami dengan sabar sehingga kami semua dapat mencapai titik ini, Saya juga ingin berterima kasih kepada keluarga dan teman-teman seperjuangan karena telah mendorong dan menyemangati satu sama lain.

Video Link at Youtube

Tidak ada

References

[1] https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2020-2021/Graf-2020-Bagian1.pdf. Diakses pada tanggal 14 Desember 2021.

[2] https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Greedy-(2021)-Bag2.pdf

[3] Faray, Safiq. (2021). Penerapan Konsep Graphf Dalam Desain Area dan Eksplorasi Pada Gim Hollow Knight. https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2021-2022/Makalah2021/Makalah-Matdis-2021%20(5).pdf. Diakses pada 23 Maret 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

Safiq Faray 13519145