

STATION FINDER USING GRAPH

LAPORAN PROYEK AKHIR

MATA KULIAH COMP6362 – DATA STRUCTURES

KELAS BB20



Oleh :

2440092132 – Primus Nathan Orvala

2440069363 - Alief Kuku NK

2440085442 - Evan Raditya

SEMESTER GENAP 2020/2021
BINA NUSANTARA UNIVERSITY
MALANG

II. Persetujuan Dosen

LEMBAR PERSETUJUAN DOSEN

STATION FINDER USING GRAPH

MATA KULIAH COMP6362 – DATA STRUCTURES

KELAS BB20

Semester Genap 2020 / 2021

Laporan akhir proyek ini adalah benar karya kami :



Primus Nathan Orvala

2440092132



Evan Raditya

2440085442



Alief Kuku NK

2440069363

Malang, 17 Juni 2021

Chasandra Puspitasari

D6365

BAB 1

LATAR BELAKANG

MRT merupakan hal baru dalam dunia transportasi yang dapat memudahkan perjalanan dalam kota dengan menggunakan rute MRT yang ada dalam kota tanpa adanya gangguan seperti macet ataupun lampu merah. MRT berkembang menjadi hal yang mulai dibutuhkan semua orang bahkan semua orang menggunakan nya. Tetapi dikarenakan nya banyak nya stasiun dan jalan yang bermacam2 terkadang orang tidak mengetahui rute mana yang dapat membawa mereka lebih cepat tanpa berkeliling ke rute yang lebih jauh.

Dengan teknologi yang ada sekarang, teknologi ini dapat memudahkan inovasi dan ciptaan baru yang dapat membantu manusia memudahkan kehidupan sehari-hari nya. Data structure menjadi salah satu hal yang dapat membantu dalam hal meningkatkan dan memberikan kemudahan kepada orang untuk dapat beraktivitas.

Kelompok kami berencana membuat aplikasi untuk mencari rute terbaik dan terpendek dalam suatu rute kereta MRT yang bisa mudah dimengerti dan tidak terlalu kompleks.

Kami mengimplementasikan graph dengan tipe weighted graph dalam program kami dengan bantuan dijkstra algorithm. Graph ini sendiri dibuat agar dapat menentukan vertex (titik stasiun) yang ada dalam rute dan membentuk edges yang menyambungkan antara vertex (titik stasiun) satu dengan yang lain. Dan weighted graph berfungsi agar setiap edges yang ada memiliki ukuran nya masing-masing.

Jadi, dalam program ini kita dapat menginput stasiun posisi kita berada dan stasiun yang ingin kita tuju dan program akan mencari rute tercepat yang ada dari semua rute stasiun yang tersedia lalu program juga akan menghitung total pembayaran yang harus dilakukan jika akan melewati rute tersebut dari stasiun pemberangkatan. Program kami ini mempermudah orang-orang yang ingin mencari rute untuk lebih cepat tiba di lokasi yang dituju.

BAB 2

LITERATURE REVIEW

Data structure adalah suatu jenis dalam pemrograman dimana data structure sendiri dibuat agar memudahkan penyimpanan dan pengaturan dari suatu data. Dan algorithm yang bekerja sebagai cara untuk menyelesaikan suatu masalah dalam pemrograman. Data structure sendiri mempunyai banyak tipe-tipe nya sedangkan yang kami gunakan dalam program kami kali ini adalah graph dimana graph sendiri memiliki tipe weighted graph yang kami gunakan.

Graph adalah kumpulan dari nodes yang memiliki data dan terkoneksi antara satu nodes dengan yang lainnya. Dari setiap koleksi data ini terdapat nodes dan edges yang dapat diartikan nodes sebagai vertices (V) dan edges (E) yang menghubungkan satu nodes dengan yang lain.

Penggunaan graph sendiri biasa nya digunakan untuk pengkoleksian data di media social atau bisa juga digunakan sebagai pencarian rute di google map/ rute kereta maupun pesawat.

Cara graph bekerja adalah jika vertex dengan vertex lain berhubungan atau terhubung dengan edges maka dikatan sebagai adjacency vertex. Path adalah Sekuen dari suatu edges yang menghubungkan vertex satu ke lain menjadikan itu sebagai jalan/path. Sedangkan graph dimana edges nya (U,V) yang terdapat arah panah itu menunjukkan bahwa graph tersebut berjalan satu arah mengikuti arah panah yang ada.

Dijkstra algorithm adalah algoritma yang kita gunakan untuk program yang kita buat. Dijkstra algorithm sendiri bekerja dengan basis jika ada beberapa path/ subpath yang menghubungkan antara vertex A ke Vertex D dan fungsi dari algoritma ini adalah untuk menemukan jalan terpendek yang dapat digunakan untuk dapat menuju ke vertex yang diinginkan. Dengan cara algoritma ini akan mem-visit setiap nodes yang ada dalam path/ subpath untuk dapat menentukan mana jalur yang tercepat.jalur yang tercepat.

Binary search tree adalah tipe data structure yang dapat membuat/ memantain beberapa sorted list dari suatu angka, dimana setiap nodes value di dalam left subtree nya lebih kecil dari pada right subtree. BST sendiri memiliki 3 operation yang pertama adalah searching operation.

Searching operation digunakan untuk mencari value yang ada pada BST dimana jika value yang ditemukan lebih kecil dari pada root nya berarti akan berada di left subtree dan jika lebih besar maka akan berada di right subtree yang akan memudahkan pencarian lebih lanjut.

Insertion operation digunakan untuk memasukkan data yang diinginkan dengan tetap memaintan aturan yang ada. Jika data yang ingin dimasukkan kurang dari root maka akan diletakkan di subtree kiri jika lebih dari root maka akan diletakkan di subtree sebelah kanani subtree sebelah kanan.

Deletion operation digunakan untuk menghapus suatu data dalam BST sendiri, contoh nya seperti jika ingin menghapus node ke 5 dalam suatu BST jika node ke 5 adalah child node maka penghapusan akan dilakukan secara normal.

BAB 3 PROGRAM DAN DEFINITION

Pseudocode

Start

BEGIN

```
Initialise pilih, berangkat, tujuan, nvertex = 11;
While pilih(≠ 2)
Print (" =====Station Finder===== ")
Print (" ----- ")
Print (" Stasiun: ")
Print (" 0.Utta station")
Print (" 1.Etogawa station")
Print (" 2.Shiojiri station")
Print (" 3.Otsuki station")
Print (" 4.Chiba station")
Print (" 5.Ueno station")
Print (" 6.Shinagawa station")
Print (" 7.Harajuku station")
Print (" 8.Meguro station")
Print (" 9.Shinjiku station")
Print (" 10.Nikko station")
Print ("Order: ")
    Print ("1.Mencari rute stasiun")
    Print ("2.Tambahkan data")
    Print ("3. Tampilkan data")
    Print ("4. Delete Data")
    Print ("5. Bayar")
    Print ("6. Keluar")
    Print (" Your input: ")
    accept pilih
    switch(pilih)
    begin
    case 1 : Print "1.Mencari rute stasiun"
            Input stasiun berangkat & stasiun tujuan
            Dijkstra()
            Display rute, jarak, dan harga
    break

    case 2 : Print "2.Tambahkan data"
            Input Id, nama, stasiun berangkat, & stasiun tujuan.
            Add_data()
            Display add data success.
    break
```

```
case 3 : Print "3. Tampilkan data "  
        Display_data()  
break
```

```
case 4 : Print "4. Delete data"  
        Input ID  
        delete_data()  
        Print "ID telah terhapus"  
break
```

```
case 5 : Print "5. Bayar"  
        Input ID  
        pay()  
        Print "ID telah membayar"  
break
```

```
case 6 : Print "6. Keluar"  
Exit program
```

```
Dijkstra(){  
for ( x < vertex; x++)  
for (y < vertex; y++)  
if (adjacency[x][y] == 0)  
temp[x][y] = 9999;  
  
else  
temp[x][y] = adjacency[x][y];  
  
for (x < vertex; x++)  
jarak[x] = temp[startnode][x];  
pred[x] = startnode;  
visited[x] = 0;  
  
jarak[startnode] = 0;  
visited[startnode] = 1;  
count = 1;  
while (count<vertex-1)  
jarakmin = 9999;  
for ( x < vertex; x++)  
if (jarak[x] < jarakmin && !visited[x])  
jarakmin = jarak[x];  
nextnode = x;  
visited[nextnode] = 1;  
for (x < vertex; x++)  
if (!visited[x])
```

```

if (jarakmin + temp[nextnode][x] < jarak[x]){
    jarak[x] = jarakmin + temp[nextnode][x];
    pred[x] = nextnode;

```

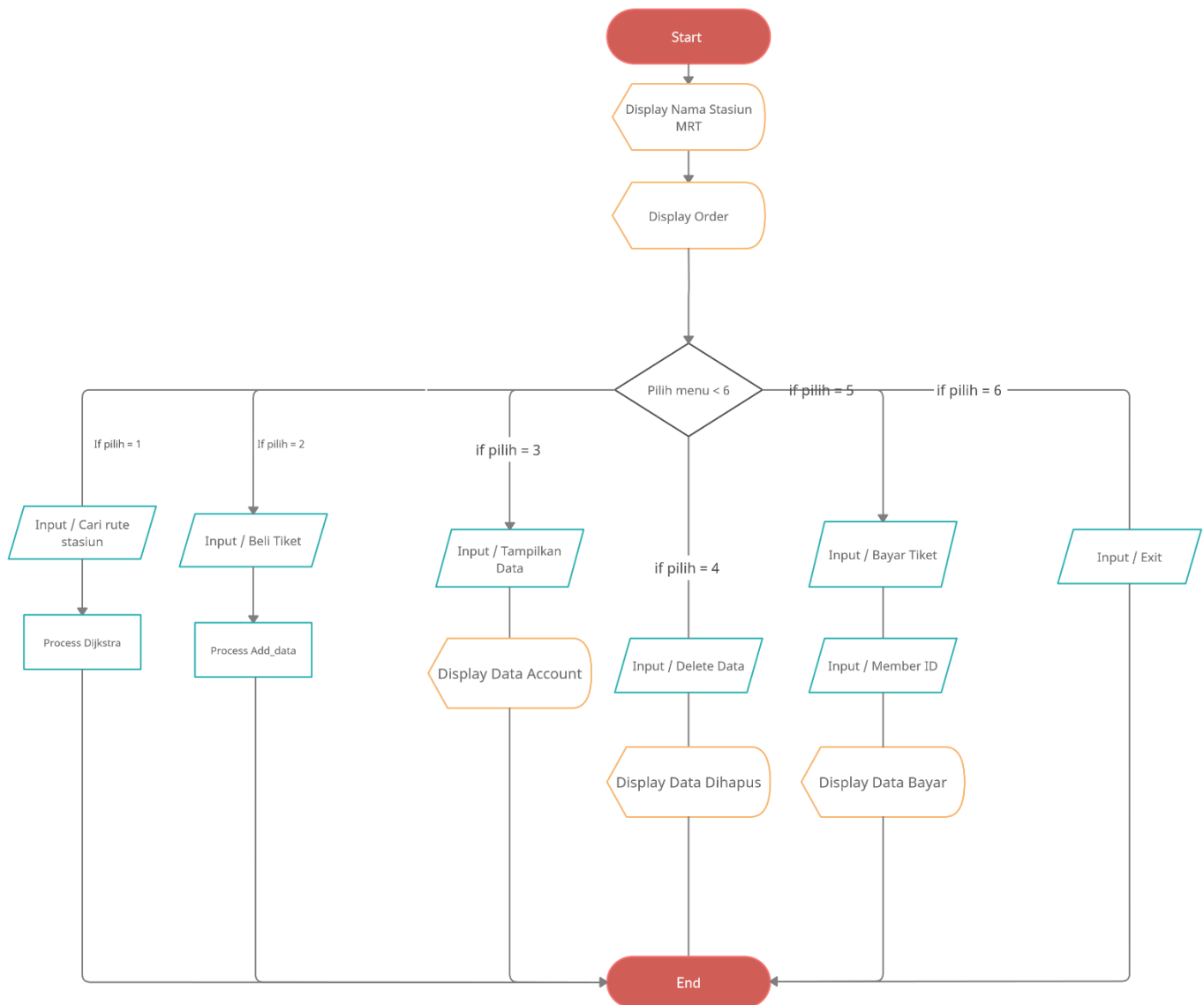
```

    count++;
}

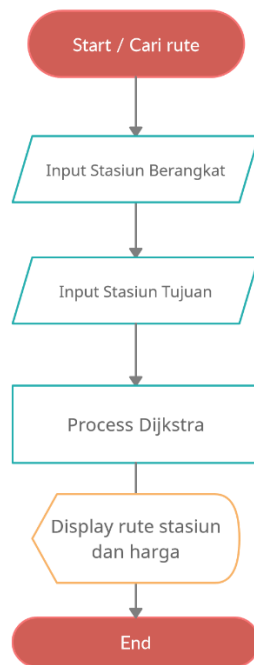
```

End

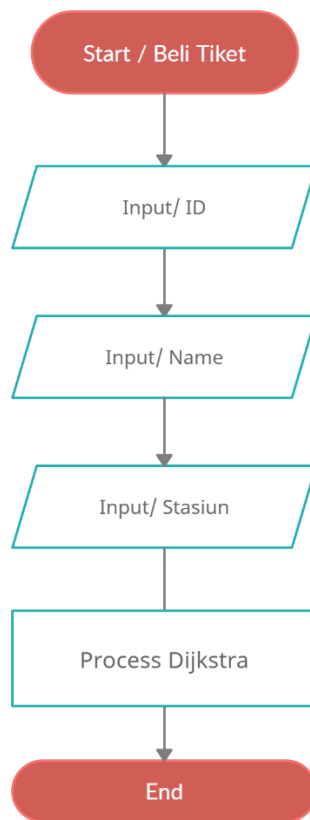
Flowchart



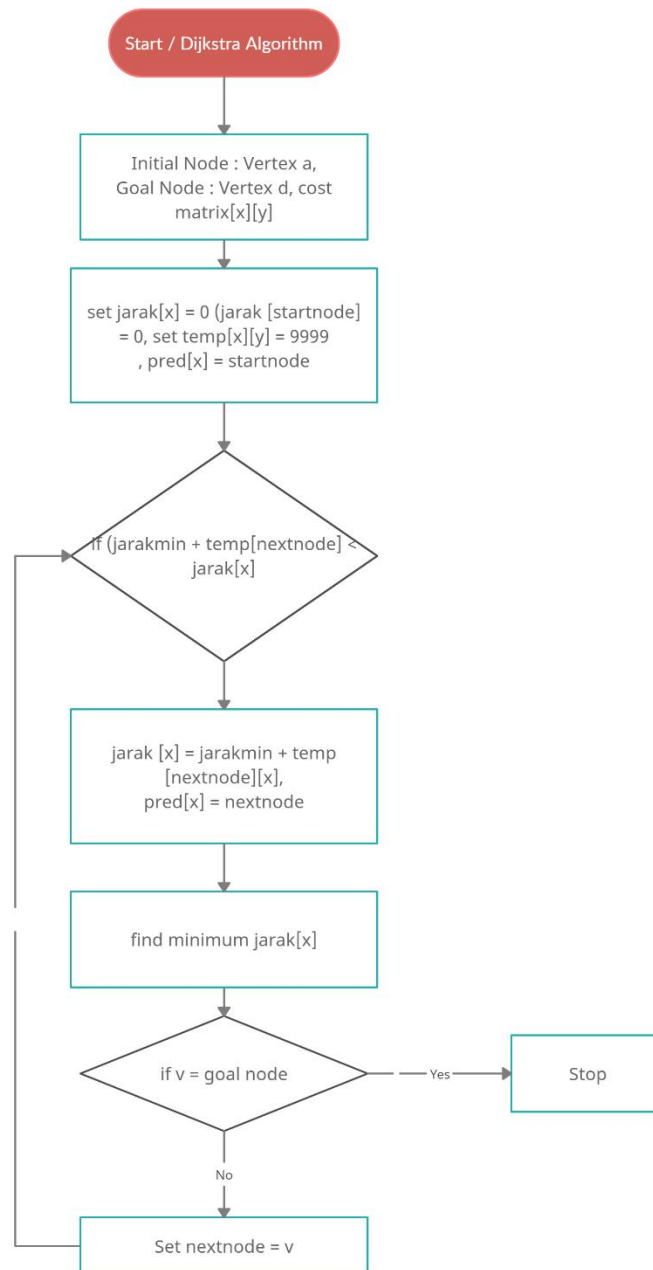
Gambar 3.1 Flowchart program



Gambar 3.2 Flowchart Cari rute

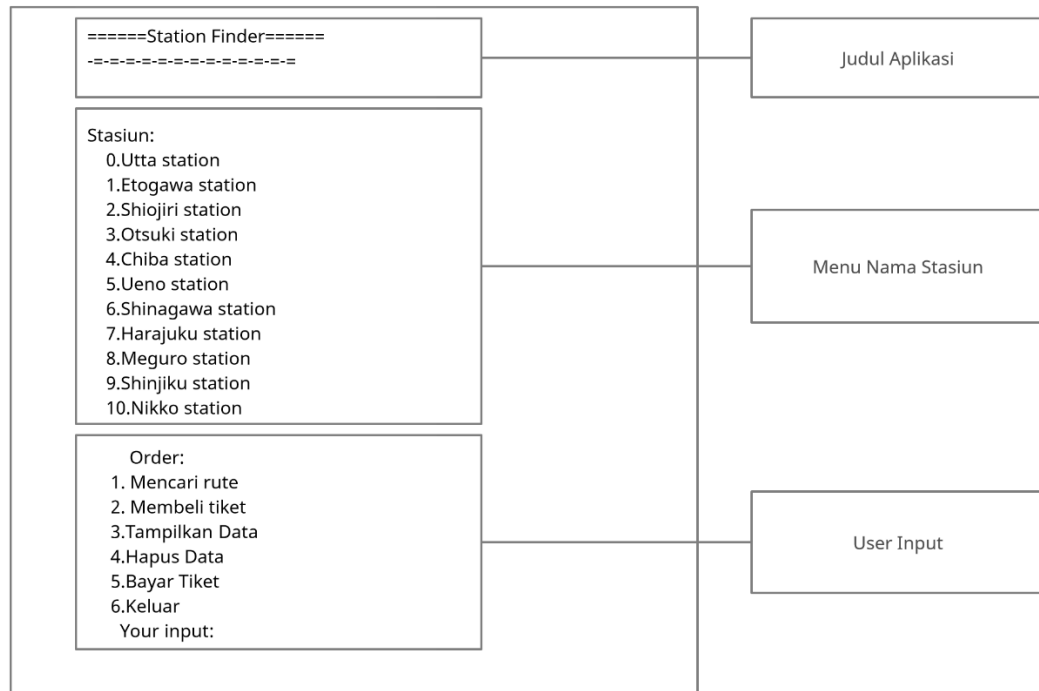


Gambar 3.3 Flowchart “Beli tiket”

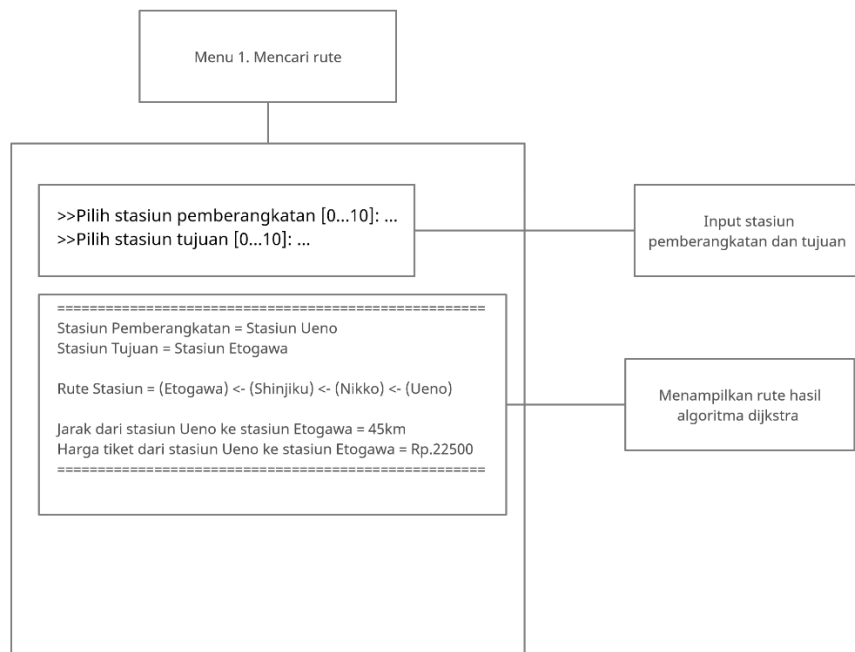


Gambar 3.4 Flowchart Algoritma Dijkstra

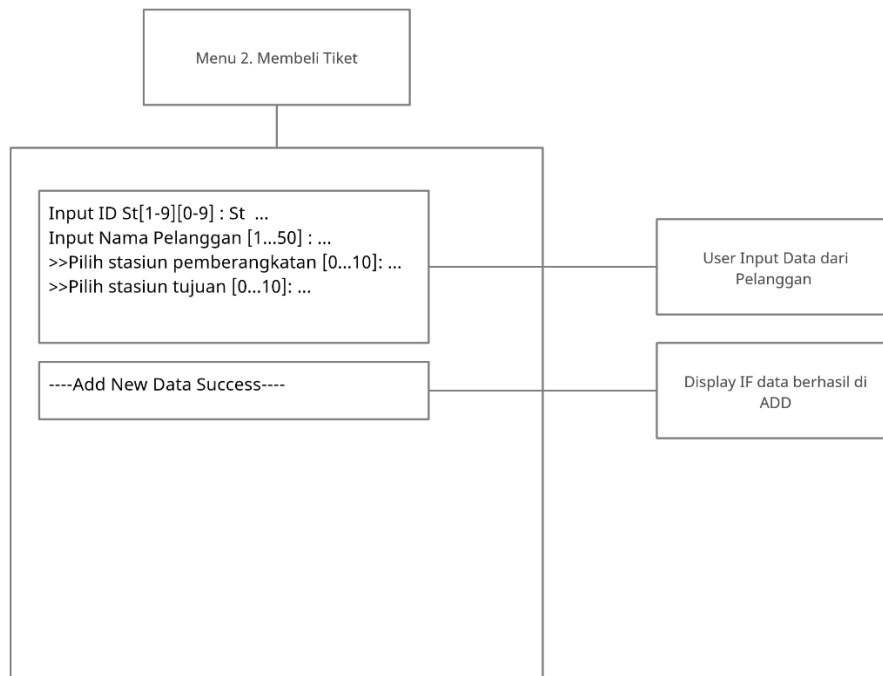
Layout design



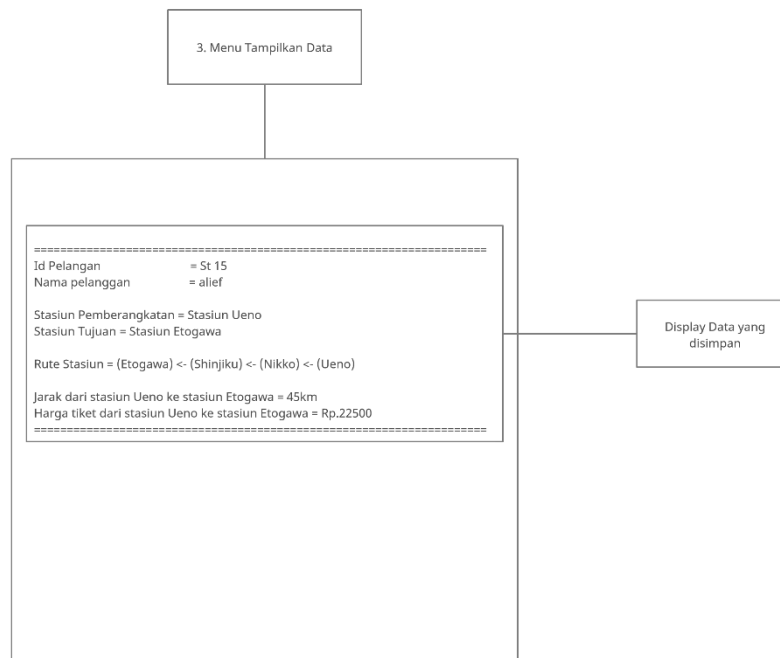
Gambar 3.5 Menu Utama



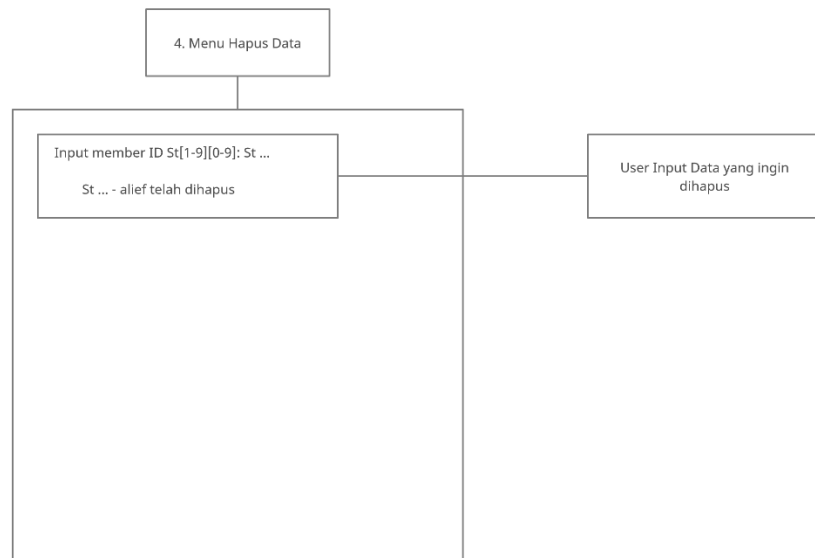
Gambar 3.6 Menu “Mencari rute”



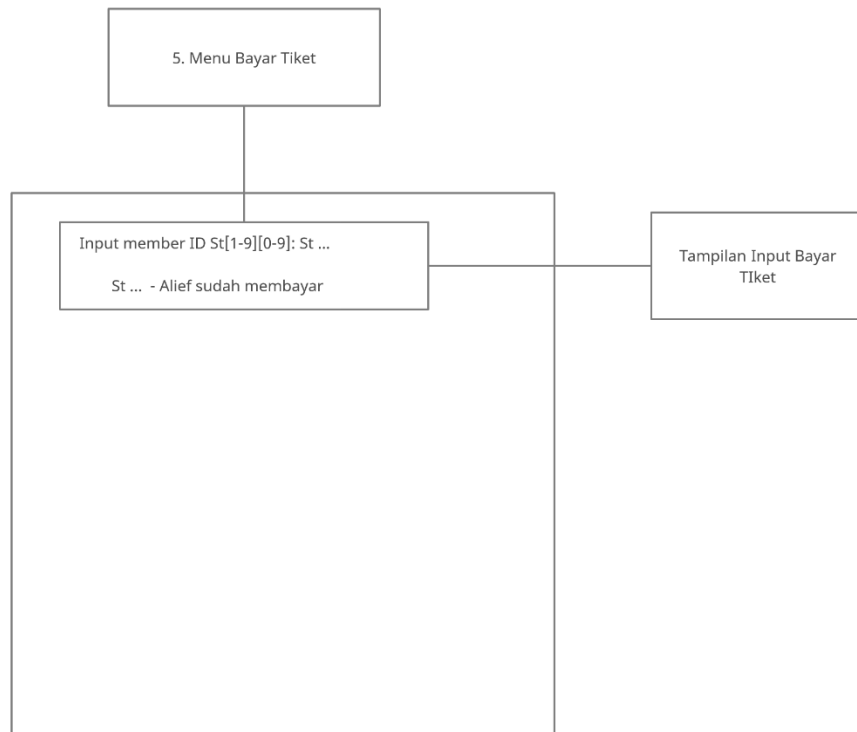
Gambar 3.7 Menu “Membeli Tiket”



Gambar 3.8 Menu “Tampilkan data”



Gambar 3.9 Menu “Hapus Data”



Gambar 3.10 Menu “Bayar Tiket”

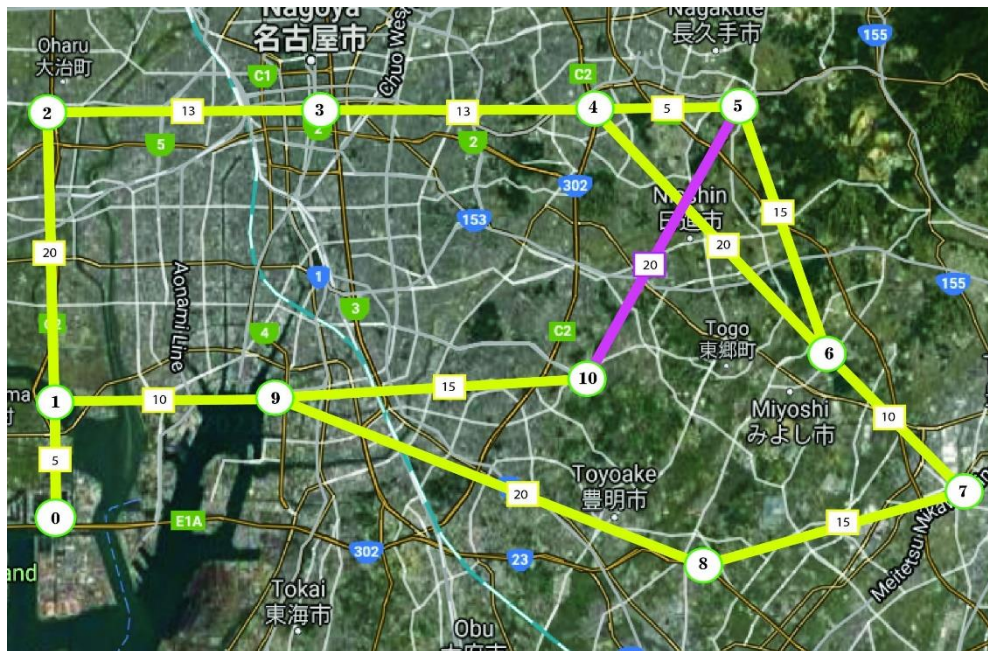
Program Features

Fitur pada program kami adalah mencari rute terdekat dalam rute MRT. User menginputkan stasiun pemberangkatan dan stasiun yang dituju. Setelah itu, program akan menampilkan rute terpendek dan jaraknya. Program juga akan menampilkan harga tiket yaitu Rp. 500 dikalikan dengan panjang rute kereta tersebut.

Program Details

Pada menu **Mencari Rute Stasiun**, kelompok kami mengimplementasikan algoritma Dijkstra untuk mencari rute terdekat. User menginputkan stasiun pemberangkatan atau source node dan stasiun yang dituju. Setelah menginputkan 2 variabel tersebut, program akan menjalankan function Dijkstra untuk mencari rute yang terdekat. Pada menu **Membeli Tiket**, kelompok kami mengimplementasikan Binnary search tree yang dimana nantinya user akan diminta untuk mengisi id dan nama user, lalu memilih stasiun pemberangkatan dan stasiun tujuan dan nantinya program akan mencatat apa yang sudah di isi oleh user. Pada menu **Tampilkan data**, kelompok kami mengimplementasikan pengurutan data secara In-order yang dimana nantinya program akan menampilkan akan mengurutkan berdasarkan ID mulai dari yang terkecil ke yang terbesar, dalam menu tampilan data berisikan ID, Nama pelanggan, Pilihan stasiun, rute, jarak dan harga tiket yang harus dibayar. Pada menu **Hapus Data**, kelompok kami mengimplementasikan Binnary search tree yang dimana user akan diminta untuk memasukan ID pelanggan yang akan dihapus, setelah memasukkan ID maka akan keluar tulisan ID dan nama user telah dihapus, lalu saat kita memilih menu tampilan data maka data yang telah dihapus tidak akan muncul di list tersebut. Pada menu **Bayar tiket**, kelompok kami mengimplementasikan Binnary search tree yang dimana nantinya user akan diminta untuk memasukan ID pelanggan yang sudah melakukan pembayaran, setelah memasukkan ID maka akan keluar tulisan ID dan nama user sudah membayar, lalu saat kita memilih menu tampilan data maka data yang user yang sudah membayar akan dihapus dari list, dan menu terakhir merupakan menu **Keluar** yang dimana nantinya saat user memilih menu tersebut maka program akan menghapus data lalu menutup program tersebut.

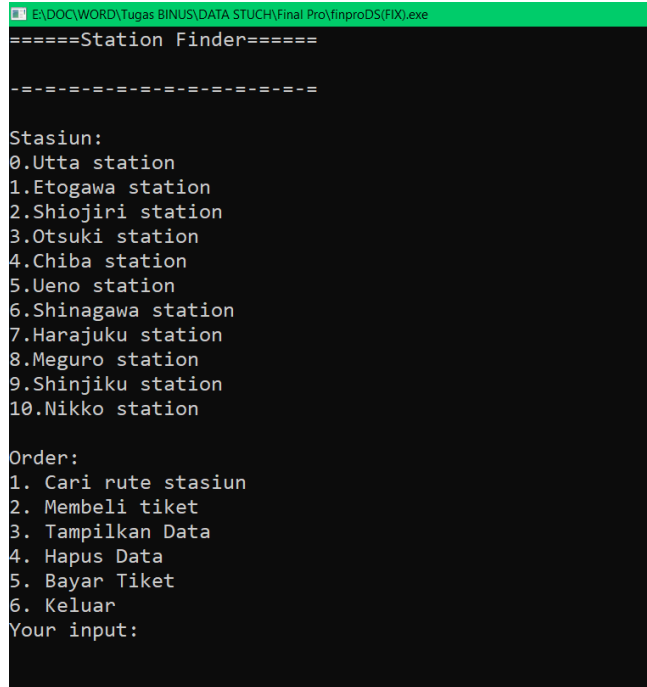
Berikut adalah gambar graph yang kita gunakan:



Gambar 3.11 Graph Rute MRT

BAB 4 RESULT

4.1 Program Screenshot



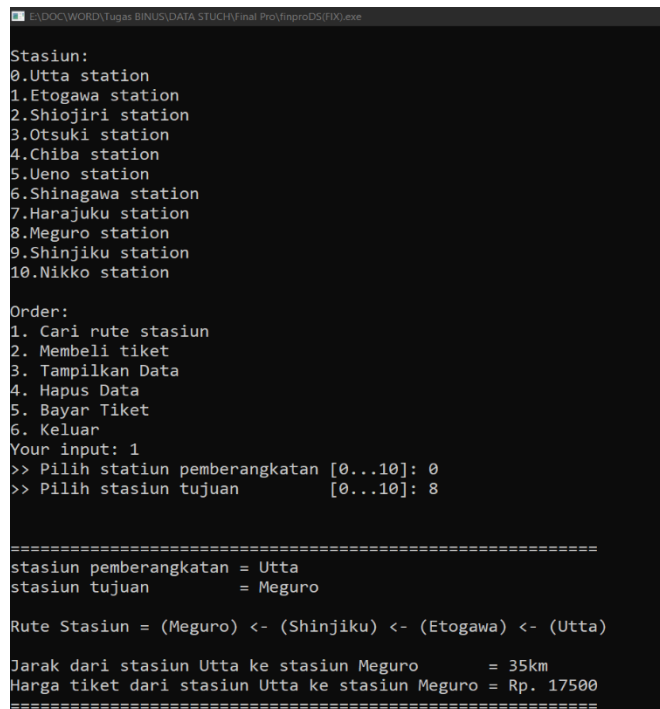
```
E:\DOC\WORD\Tugas BINUS\DATA STUCH\Final Pro\finproDS(FIX).exe
====Station Finder====

-----

Stasiun:
0.Utta station
1.Etogawa station
2.Shiojiri station
3.Otsuki station
4.Chiba station
5.Ueno station
6.Shinagawa station
7.Harajuku station
8.Meguro station
9.Shinjiku station
10.Nikko station

Order:
1. Cari rute stasiun
2. Membeli tiket
3. Tampilkan Data
4. Hapus Data
5. Bayar Tiket
6. Keluar
Your input:
```

Gambar 4.1 Tampilan menu utama



```
E:\DOC\WORD\Tugas BINUS\DATA STUCH\Final Pro\finproDS(FIX).exe

Stasiun:
0.Utta station
1.Etogawa station
2.Shiojiri station
3.Otsuki station
4.Chiba station
5.Ueno station
6.Shinagawa station
7.Harajuku station
8.Meguro station
9.Shinjiku station
10.Nikko station

Order:
1. Cari rute stasiun
2. Membeli tiket
3. Tampilkan Data
4. Hapus Data
5. Bayar Tiket
6. Keluar
Your input: 1
>> Pilih stasiun pemberangkatan [0...10]: 0
>> Pilih stasiun tujuan          [0...10]: 8

=====
stasiun pemberangkatan = Utta
stasiun tujuan         = Meguro

Rute Stasiun = (Meguro) <- (Shinjiku) <- (Etogawa) <- (Utta)

Jarak dari stasiun Utta ke stasiun Meguro      = 35km
Harga tiket dari stasiun Utta ke stasiun Meguro = Rp. 17500
=====
```

Gambar 4.2 Tampilan menu “Mencari rute stasiun”


```
E:\DOC\WORD\Tugas BINUS\DATA STUCH\Final Pro\finproDS(FDX).exe

=====Station Finder=====

-----

Stasiun:
0.Utta station
1.Etogawa station
2.Shiojiri station
3.Otsuki station
4.Chiba station
5.Ueno station
6.Shinagawa station
7.Harajuku station
8.Meguro station
9.Shinjiku station
10.Nikko station

Order:
1. Cari rute stasiun
2. Membeli tiket
3. Tampilkan Data
4. Hapus Data
5. Bayar Tiket
6. Keluar
Your input: 2
Input ID St[1-9][0-9]          : St 10
Input Nama Pelanggan [1..50]   : Evan Raditya
>> Pilih stasiun pemberangkatan [0...10]: 9
>> Pilih stasiun tujuan        [0...10]: 3

---Add New Data Success---

=====Station Finder=====
```

Gambar 4.3 Tampilan menu “Membeli Tiket”

```
E:\DOC\WORD\Tugas BINUS\DATA STUCH\Final Pro\finproDS(FDX).exe

4. Hapus Data
5. Bayar Tiket
6. Keluar
Your input: 3

=====
Id Pelangan           = St 10
Nama pelanggan        = Evan Raditya
Stasiun Pemberangkatan = Stasiun Shinjiku
Stasiun Tujuan        = Stasiun Otsuki

Rute Stasiun = (Otsuki) <- (Shiojiri) <- (Etogawa) <- (Shinjiku)

Jarak dari stasiun Shinjiku ke stasiun Otsuki = 43km
Harga tiket dari stasiun Shinjiku ke stasiun Otsuki = Rp.21500
=====

=====
Id Pelangan           = St 42
Nama pelanggan        = Alief Kuku
Stasiun Pemberangkatan = Stasiun Ueno
Stasiun Tujuan        = Stasiun Nikko

Rute Stasiun = (Nikko) <- (Ueno)

Jarak dari stasiun Ueno ke stasiun Nikko = 20km
Harga tiket dari stasiun Ueno ke stasiun Nikko = Rp.10000
=====

=====
Id Pelangan           = St 80
Nama pelanggan        = Primus Nathan
Stasiun Pemberangkatan = Stasiun Shinagawa
Stasiun Tujuan        = Stasiun Shiojiri

Rute Stasiun = (Shiojiri) <- (Otsuki) <- (Chiba) <- (Shinagawa)

Jarak dari stasiun Shinagawa ke stasiun Shiojiri = 46km
Harga tiket dari stasiun Shinagawa ke stasiun Shiojiri = Rp.23000
=====
```

Gambar 4.4 Tampilan menu ”Tampilkan Data”

```

Order:
1. Cari rute stasiun
2. Membeli tiket
3. Tampilkan Data
4. Hapus Data
5. Bayar Tiket
6. Keluar
Your input: 4
Input member ID St[1-9][0-9]: St10

      St10 - Evan Raditya telah dihapus

```

```

Order:
1. Cari rute stasiun
2. Membeli tiket
3. Tampilkan Data
4. Hapus Data
5. Bayar Tiket
6. Keluar
Your input: 3

=====
Id Pelangan           = St 42
Nama pelanggan        = Alief Kuku
Stasiun Pemberangkatan = Stasiun Ueno
Stasiun Tujuan        = Stasiun Nikko

Rute Stasiun = (Nikko) <- (Ueno)

Jarak dari stasiun Ueno ke stasiun Nikko      = 20km
Harga tiket dari stasiun Ueno ke stasiun Nikko = Rp.10000
=====

=====
Id Pelangan           = St 80
Nama pelanggan        = Primus Nathan
Stasiun Pemberangkatan = Stasiun Shinagawa
Stasiun Tujuan        = Stasiun Shiojiri

Rute Stasiun = (Shiojiri) <- (Otsuki) <- (Chiba) <- (Shinagawa)

Jarak dari stasiun Shinagawa ke stasiun Shiojiri = 46km
Harga tiket dari stasiun Shinagawa ke stasiun Shiojiri = Rp.23000
=====

```

Gambar 4.5 Tampilan menu “Hapus Data”

```

Order:
1. Cari rute stasiun
2. Membeli tiket
3. Tampilkan Data
4. Hapus Data
5. Bayar Tiket
6. Keluar
Your input: 5
Input member ID St[1-9][0-9]: St80

      St80 - Primus Nathan sudah membayar

```

```

Order:
1. Cari rute stasiun
2. Membeli tiket
3. Tampilkan Data
4. Hapus Data
5. Bayar Tiket
6. Keluar
Your input: 3

=====
Id Pelangan           = St 42
Nama pelanggan        = Alief Kuku
Stasiun Pemberangkatan = Stasiun Ueno
Stasiun Tujuan        = Stasiun Nikko

Rute Stasiun = (Nikko) <- (Ueno)

Jarak dari stasiun Ueno ke stasiun Nikko      = 20km
Harga tiket dari stasiun Ueno ke stasiun Nikko = Rp.10000
=====

```

Gambar 4.6 Tampilan menu “Bayar Tiket”

4.2 Source Code

```
#include <stdio.h>
#include <stdlib.h>
#include <malloc.h>
#include <string.h>
#include <conio.h>

struct stasiun{
    char *nama; // menggunakan pointer
};

struct node{
    int idval;
    char user[100];
    int berangkat;
    int tujuan;
    struct node *left;
    struct node *right;
};

void dijkstra(int adjacency[11][11], int vertex, int startnode, int endnode);
struct node *newdata(int idval, const char *user, int berangkat, int tujuan){
    struct node *data = malloc(sizeof(struct node));
    data->idval = idval;
    strcpy(data->user, user);
    data->berangkat = berangkat;
    data->tujuan = tujuan;
    data->left = data->right = 0;

    return data;
}

struct node *adddata(struct node *root, int idval, const char *user, int berangkat,
int tujuan, int height){
    if (height<3){
        if(root==0){
            printf("\n---Add New Data Success---\n\n");
            return newdata(idval, user, berangkat, tujuan);
        }

        else{
            while(height<91){
                if(root->idval < idval){
                    root->right=adddata(root->right,idval,user,berangkat,tujuan,height+1);
                }
                else if(root->idval > idval){
                    root->left = adddata(root->left,idval,user,berangkat,tujuan,height+1);
                }
                break;
            }
        }
    }
}
```

```

    }
    else if(height==3){
        printf("\n\n---Maximum Tree Height is 90---");
    }

    return root;
}
struct node *search(struct node *root, int idval){
    if (root!=0) {
        if(root->idval==idval){
            return root;
        }
        else {
            struct node * tmp;
            tmp = search(root->left, idval);
            if (tmp==0) {
                tmp=search(root->right, idval);
            }
            return tmp;
        }
    }
}

struct node *successor(struct node * root) {
    struct node * tmp = root;
    while (tmp && tmp -> right != NULL){
        tmp = tmp -> right;
    }
    return tmp;
}

struct node *buy(struct node * root, int idval){
    if (root == NULL) {
        return root;
    }
    if (idval < root -> idval) {
        root -> left = buy(root -> left, idval);
    }
    else if (idval > root -> idval) {
        root -> right = buy(root -> right, idval);
    }
    else {
        if (root -> left == NULL) {
            struct node * tmp = root -> right;
            printf("\tSt%d - %s sudah membayar\n\n\n", root -> idval, root ->
user);
            free(root);
            return tmp;
        }
        else if (root -> right == NULL) {
            struct node * tmp = root -> left;
            printf("\tSt%d - %s sudah membayar\n\n\n", root -> idval, root ->
user);
            free(root);
            return tmp;
        }
    }
}

```



```

};
int startnode;
int endnode;
char namast[100];
char namast2[100];
struct stasiun st[100];

st[0].nama = "Utta";
st[1].nama = "Etogawa";
st[2].nama = "Shiojiri";
st[3].nama = "Otsuki";
st[4].nama = "Chiba";
st[5].nama = "Ueno";
st[6].nama = "Shinagawa";
st[7].nama = "Harajuku";
st[8].nama = "Meguro";
st[9].nama = "Shinjiku";
st[10].nama = "Nikko";

if(startnode==0){
    strcpy(namast,st[0].nama);
}
else if(startnode==1){
    strcpy(namast,st[1].nama);
}
else if(startnode==2){
    strcpy(namast,st[2].nama);
}
else if(startnode==3){
    strcpy(namast,st[3].nama);
}
else if(startnode==4){
    strcpy(namast,st[4].nama);
}
else if(startnode==5){
    strcpy(namast,st[5].nama);
}
else if(startnode==6){
    strcpy(namast,st[6].nama);
}
else if(startnode==7){
    strcpy(namast,st[7].nama);
}
else if(startnode==8){
    strcpy(namast,st[8].nama);
}
else if(startnode==9){
    strcpy(namast,st[9].nama);
}

```

```

{0, 0, 0, 0, 5, 0, 15, 0, 0, 0, 20},
{0, 0, 0, 0, 20, 15, 0, 10, 0, 0, 0},
{0, 0, 0, 0, 0, 0, 10, 0, 15, 0, 0},
{0, 0, 0, 0, 0, 0, 0, 15, 0, 20, 0},
{0, 10, 0, 0, 0, 0, 0, 0, 20, 0, 15},
{0, 0, 0, 0, 0, 20, 0, 0, 0, 15, 0}

```

```

        else if(startnode==10){
            strcpy(namast,st[10].nama);
        }

        //
        if(endnode==0){
            strcpy(namast2,st[0].nama);
        }
        else if(endnode==1){
            strcpy(namast2,st[1].nama);
        }
        else if(endnode==2){
            strcpy(namast2,st[2].nama);
        }
        else if(endnode==3){
            strcpy(namast2,st[3].nama);
        }
        else if(endnode==4){
            strcpy(namast2,st[4].nama);
        }
        else if(endnode==5){
            strcpy(namast2,st[5].nama);
        }
        else if(endnode==6){
            strcpy(namast2,st[6].nama);
        }
        else if(endnode==7){
            strcpy(namast2,st[7].nama);
        }
        else if(endnode==8){
            strcpy(namast2,st[8].nama);
        }
        else if(endnode==9){
            strcpy(namast2,st[9].nama);
        }
        else if(endnode==10){
            strcpy(namast2,st[10].nama);
        }

        if(root==0){
            return;
        }
        startnode = root->berangkat;
        endnode = root->tujuan;

        display(root->left);

printf("\n=====\\n")
;
        printf("Id Pelangan           = St %d\\n",root->idval);
        printf("Nama pelanggan           = %s\\n",root->user);
        printf("Stasiun Pemberangkatan       = Stasiun %s\\n",st[startnode].nama);
        printf("Stasiun Tujuan               = Stasiun %s\\n",st[endnode].nama);
        dijkstra(adjacency, vertex, root->berangkat, root->tujuan);
        display(root->right);

```

```

}
void dijkstra(int adjacency[11][11], int vertex, int startnode , int endnode){
    int temp[11][11], jarak[11], pred[11], visited[11];
    int count, jarakmin, nextnode, x, y;
    char namast[100];
    char namast2[100];
    struct stasiun st[100];
    st[0].nama = "Utta";
    st[1].nama = "Etogawa";
    st[2].nama = "Shiojiri";
    st[3].nama = "Otsuki";
    st[4].nama = "Chiba";
    st[5].nama = "Ueno";
    st[6].nama = "Shinagawa";
    st[7].nama = "Harajuku";
    st[8].nama = "Meguro";
    st[9].nama = "Shinjiku";
    st[10].nama = "Nikko";

    //berangkat
    if(startnode==0){
        strcpy(namast,st[0].nama);
    }
    else if(startnode==1){
        strcpy(namast,st[1].nama);
    }
    else if(startnode==2){
        strcpy(namast,st[2].nama);
    }
    else if(startnode==3){
        strcpy(namast,st[3].nama);
    }
    else if(startnode==4){
        strcpy(namast,st[4].nama);
    }
    else if(startnode==5){
        strcpy(namast,st[5].nama);
    }
    else if(startnode==6){
        strcpy(namast,st[6].nama);
    }
    else if(startnode==7){
        strcpy(namast,st[7].nama);
    }
    else if(startnode==8){
        strcpy(namast,st[8].nama);
    }
    else if(startnode==9){
        strcpy(namast,st[9].nama);
    }
    else if(startnode==10){
        strcpy(namast,st[10].nama);
    }

    //tujuan

```



```

if(endnode==0){
    strcpy(namast2,st[0].nama);
}
else if(endnode==1){
    strcpy(namast2,st[1].nama);
}
else if(endnode==2){
    strcpy(namast2,st[2].nama);
}
else if(endnode==3){
    strcpy(namast2,st[3].nama);
}
else if(endnode==4){
    strcpy(namast2,st[4].nama);
}
else if(endnode==5){
    strcpy(namast2,st[5].nama);
}
else if(endnode==6){
    strcpy(namast2,st[6].nama);
}
else if(endnode==7){
    strcpy(namast2,st[7].nama);
}
else if(endnode==8){
    strcpy(namast2,st[8].nama);
}
else if(endnode==9){
    strcpy(namast2,st[9].nama);
}
else if(endnode==10){
    strcpy(namast2,st[10].nama);
}

for (x = 0; x < vertex; x++){
    for (y = 0; y < vertex; y++){
        if (adjacency[x][y] == 0){
            temp[x][y] = 9999;
        }
        else{
            temp[x][y] = adjacency[x][y];
        }
    }
}
for (x = 0; x < vertex; x++){
    jarak[x] = temp[startnode][x];
    pred[x] = startnode;
    visited[x] = 0;
}
jarak[startnode] = 0;
visited[startnode] = 1;
count = 1;
while (count<vertex-1){
    jarakmin = 9999;

```

```

        for (x = 0; x < vertex; x++){
            if (jarak[x] < jarakmin && !visited[x]){
                jarakmin = jarak[x];
                nextnode = x;
            }
        }
        visited[nextnode] = 1;
        for (x = 0; x < vertex; x++){
            if (!visited[x]){
                if (jarakmin + temp[nextnode][x] < jarak[x]){
                    jarak[x] = jarakmin + temp[nextnode][x];
                    pred[x] = nextnode;
                }
            }
        }
        count++;
    }

    for (x = 0; x <= endnode; x++){
        if (x == endnode){
            printf("\nRute Stasiun = (%s)", st[x].nama);
            y = x;

            do{
                y = pred[y];
                printf(" <- (%s)", st[y].nama);
            }
            while(y != startnode);
            printf("\n\nJarak dari stasiun %s ke stasiun %s      =
%dkm",namast, namast2, jarak[x]);
            printf("\nHarga tiket dari stasiun %s ke stasiun %s =
Rp.%d",namast, namast2,500 * jarak[x]);

            printf("\n=====
===\n\n");
        }
    }
}

void pathfinder(int adjacency[11][11], int vertex, int startnode , int endnode){
    int temp[11][11], jarak[11], pred[11], visited[11];
    int count, jarakmin, nextnode, x, y;
    char namast[100];
    char namast2[100];
    struct stasiun st[100];
    st[0].nama = "Utta";
    st[1].nama = "Etogawa";
    st[2].nama = "Shiojiri";
    st[3].nama = "Otsuki";
    st[4].nama = "Chiba";
    st[5].nama = "Ueno";
    st[6].nama = "Shinagawa";
    st[7].nama = "Harajuku";
    st[8].nama = "Meguro";
    st[9].nama = "Shinjiku";

```

```
st[10].nama = "Nikko";

//berangkat
if(startnode==0){
    strcpy(namast,st[0].nama);
}
else if(startnode==1){
    strcpy(namast,st[1].nama);
}
else if(startnode==2){
    strcpy(namast,st[2].nama);
}
else if(startnode==3){
    strcpy(namast,st[3].nama);
}
else if(startnode==4){
    strcpy(namast,st[4].nama);
}
else if(startnode==5){
    strcpy(namast,st[5].nama);
}
else if(startnode==6){
    strcpy(namast,st[6].nama);
}
else if(startnode==7){
    strcpy(namast,st[7].nama);
}
else if(startnode==8){
    strcpy(namast,st[8].nama);
}
else if(startnode==9){
    strcpy(namast,st[9].nama);
}
else if(startnode==10){
    strcpy(namast,st[10].nama);
}

//tujuan
if(endnode==0){
    strcpy(namast2,st[0].nama);
}
else if(endnode==1){
    strcpy(namast2,st[1].nama);
}
else if(endnode==2){
    strcpy(namast2,st[2].nama);
}
else if(endnode==3){
    strcpy(namast2,st[3].nama);
}
else if(endnode==4){
    strcpy(namast2,st[4].nama);
}
else if(endnode==5){
    strcpy(namast2,st[5].nama);
}
```

```

    }
    else if(endnode==6){
        strcpy(namast2,st[6].nama);
    }
    else if(endnode==7){
        strcpy(namast2,st[7].nama);
    }
    else if(endnode==8){
        strcpy(namast2,st[8].nama);
    }
    else if(endnode==9){
        strcpy(namast2,st[9].nama);
    }
    else if(endnode==10){
        strcpy(namast2,st[10].nama);
    }

    for (x = 0; x < vertex; x++){
        for (y = 0; y < vertex; y++){
            if (adjacency[x][y] == 0){
                temp[x][y] = 9999;
            }
            else{
                temp[x][y] = adjacency[x][y];
            }
        }
    }
    for (x = 0; x < vertex; x++){
        jarak[x] = temp[startnode][x];
        pred[x] = startnode;
        visited[x] = 0;
    }
    jarak[startnode] = 0;
    visited[startnode] = 1;
    count = 1;
    while (count<vertex-1){
        jarakmin = 9999;
        for (x = 0; x < vertex; x++){
            if (jarak[x] < jarakmin && !visited[x]){
                jarakmin = jarak[x];
                nextnode = x;
            }
        }
        visited[nextnode] = 1;
        for (x = 0; x < vertex; x++){
            if (!visited[x]){
                if (jarakmin + temp[nextnode][x] < jarak[x]){
                    jarak[x] = jarakmin + temp[nextnode][x];
                    pred[x] = nextnode;
                }
            }
        }
        count++;
    }
}

```

```

        for (x = 0; x <= endnode; x++){
            if (x == endnode){

printf("\n\n=====");
                printf("\nstasiun pemberangkatan = %s",namast);
                printf("\nstasiun tujuan          = %s",namast2);
                printf("\n\nRute Stasiun = (%s)", st[x].nama);
                y = x;

                do{
                    y = pred[y];
                    printf(" <- (%s)", st[y].nama);
                }
                while(y != startnode);
                printf("\n\nJarak dari stasiun %s ke stasiun %s          =
%dkm",namast, namast2, jarak[x]);
                printf("\nHarga tiket dari stasiun %s ke stasiun %s = Rp.
%d",namast, namast2, 500 * jarak[x] );

printf("\n=====\\n\\n");

            }

        }

}

int main(){
    int pilih;
    int berangkat;
    int tujuan;
    int idval;
    char user[100];
    int height=0;
    struct node *root = 0;
    int vertex = 11;
    int adjacency[11][11]={ {0, 5, 0, 0, 0, 0, 0, 0, 0, 0, 0},
                                {5, 0, 20, 0, 0, 0, 0, 0, 0, 10, 0},
                                {0, 20, 0, 13, 0, 0, 0, 0, 0, 0, 0},
                                {0, 0, 13, 0, 13, 0, 0, 0, 0, 0, 0},
                                {0, 0, 0, 13, 0, 5, 20, 0, 0, 0, 0},
                                {0, 0, 0, 0, 5, 0, 15, 0, 0, 0, 20},
                                {0, 0, 0, 0, 20, 15, 0, 10, 0, 0, 0},
                                {0, 0, 0, 0, 0, 0, 10, 0, 15, 0, 0},
                                {0, 0, 0, 0, 0, 0, 0, 15, 0, 20, 0},
                                {0, 10, 0, 0, 0, 0, 0, 0, 20, 0, 15},
                                {0, 0, 0, 0, 0, 20, 0, 0, 0, 15, 0}

};

    while (pilih!=6){
        printf("====Station Finder====\\n\\n");
        printf("-=====\\n\\n");
        printf("Stasiun: \\n");
        printf("0.Utta station\\n");
        printf("1.Etogawa station\\n");
        printf("2.Shiojiri station\\n");
        printf("3.Otsuki station\\n");
        printf("4.Chiba station\\n");
    }
}

```

```

printf("5.Ueno station\n");
printf("6.Shinagawa station\n");
printf("7.Harajuku station\n");
printf("8.Meguro station\n");
printf("9.Shinjiku station\n");
printf("10.Nikko station\n\n");
printf("Order: \n");
printf("1. Cari rute stasiun\n");
printf("2. Membeli tiket\n");
printf("3. Tampilkan Data\n");
printf("4. Hapus Data\n");
printf("5. Bayar Tiket\n");
printf("6. Keluar\n");
printf("Your input: ");
scanf("%d",&pilih);

if(pilih==1){
    while(1){
        printf(">> Pilih stasiun pemberangkatan [0...10]: ");
        scanf("%d",&berangkat); //startnode
        if (berangkat<0 || berangkat>10) {
            printf("\n---Station Doesnt Exist---\n\n");
        }

        else{
            printf(">> Pilih stasiun tujuan          [0...10]: ");
            scanf("%d",&tujuan);
            if (tujuan<0 || tujuan>10) {
                printf("\n---Station Doesnt Exist---\n\n");
            }
            else{
                pathfinder(adjacency, vertex, berangkat, tujuan);
            }
            break;
        }
    }
}

else if(pilih==2){
    while(1){ //jika inputan benar
        printf("Input ID St[1-9][0-9]          : St ");
        scanf("%d",&idval);
        if (idval<10 || idval>99){
            printf("\n---Id Value too much---\n\n");
        }
        else{
            while(1){
                printf("Input Nama Pelanggan [1...50]      : ");

                getchar();
                scanf(" %[^\n]",user);
                if (strlen(user)<0 || strlen(user)>50){
                    printf("\n---Name Value too much---\n\n");
                }
                else{

```

```

                                while(1){
                                    printf(">> Pilih stasiun
pemberangkatan [0...10]: ");
                                scanf("%d",&berangkat); //startnode
                                if (berangkat<0 ||
berangkat>10) {
                                    printf("\n---Station
Doesnt Exist---\n\n");
                                }
                                else{
                                    while(1){
                                        printf(">> Pilih
stasiun tujuan          [0...10]: ");
                                        scanf("%d",&tujuan);
                                        if (tujuan<0 ||
tujuan>10) {
                                            printf("\n-
--Station Doesnt Exist---\n\n");
                                        }
                                        else{
                                            root =
adddata(root, idval, user, berangkat, tujuan, height);
                                            break;
                                        }
                                        else
                                        {
                                            printf("\n-
--Add New Data Failed Please Try again\n\n");
                                            break;
                                        }
                                    }
                                }
                                break;
                            }
                        }
                    }
                }
            }
        }
    }
}
else if(pilih==3){
    display(root);
}
else if(pilih==4){
    if (root == NULL) {
        printf("\n--- There is no data in the node ---\n\n");
    }
    else {
        while (1) {
            printf("Input member ID St[1-9][0-9]: St");

```

```

scanf("%d", & idval);
printf("\n");
    if (idval < 10 || idval > 99){
        printf("-- ID doesn't exist ---\n\n");
    }
    else if (search(root, idval) == NULL){
        printf("--- Member ID is not found---
\n\n");

    }
    else if (search(root, idval) != NULL) {
        root = delete_data(root, idval);
        break;
    }
}

}

}
else if(pilih==5){
    if (root == NULL) {
        printf("\n--- There is no data in the node ---\n\n");
    }
    else {
        while (1) {
            printf("Input member ID St[1-9][0-9]: St");
            scanf("%d", & idval);
            printf("\n");
            if (idval < 10 || idval > 99){
                printf("-- ID doesn't exist ---\n\n");
            }
            else if (search(root, idval) == NULL){
                printf("--- Member ID is not found---
\n\n");

            }
            else if (search(root, idval) != NULL) {
                root = buy(root, idval);
                break;
            }
        }
    }
}

}

}

}

```


REFERENCE

Dijkstra's Algorithm. Programiz. (n.d.) Programiz.Com. Retrieved June 17, 2021, from.
<https://www.programiz.com/dsa/dijkstra-algorithm>

Mishra, N. (2016, November 16). Dijkstra's Algorithm in C. The Crazy Programmer. Retrieved June 17, 2021, from. <https://www.thecrazyprogrammer.com/2014/03/dijkstra-algorithm-for-finding-shortest-path-of-a-graph.html>

Binary Search Tree. (n.d.). Programiz.Com. Retrieved June 21, 2021, from
<https://www.programiz.com/dsa/binary-search-tree>

Assesment

LEMBAR PENILAIAN

STATION FINDER USING GRAPH

MATA KULIAH COMP6362 – DATA STRUCTURES

KELAS BB20

Semester Genap 2020 / 2021

DAFTAR MAHASISWA	NILAI				BOBOT				KREDIT				TOTAL KREDIT
	1	2	3	4	1	2	3	4	1	2	3	4	
2440085442 - Evan Raditya					20%	20%	24%	36%					
2440092132 - Primus Nathan					20%	20%	24%	36%					
2440069363 - Alief Kukuh NK					20%	20%	24%	36%					
TOTAL													

KETERANGAN :

- **Skala Penilaian : 0 sd 100**
- **Komponen**
 1. : Laporan
 2. : Produk
 3. : Pengetahuan
 4. : Solusi

Malang, 17 Juni 2021

Chasandra Puspitasari

D6365

- Job Desc :

1. Primus Nathan Orvala : Coding, Laporan, Pseudocode.
2. Evan Raditya : Coding, Flowchart, Laporan
3. Alief Kukuh : Coding, Laporan, Layout design, Pseudocode