Bus Routes 123

Leader:

LIU YIXUAN (2I314)

Members:

CHUA RUI HONG (2I303)

JAVIER CHAN YAN KAI (2I305)

PIERRE YEAP YU SONG (2I321)

Hwa Chong Institution (High School)

CONTENTS

Page

1. Introduction

1.1 Description

The aim of the project is to find the shortest route between every bus stop in Singapore. A C++ computer programming algorithm which embodies Dijkstra’s Algorithm will be created to achieve the aim.

1.2 Rationale, Objectives and research questions

1.2.1 Objectives

The objectives of this project are as follows:

- Assist people find the most efficient ways to travel via bus to get from one bus stop, to another
- To use graph theory to find the most efficient ways to get from one point to another in Singapore
- Create an algorithm to help plan a bus journey

1.2.2 Research Questions

1. To find the shortest route to get from one bus stop to another factoring in only the distance.

2. To find the shortest route to get from one bus stop to another factoring in the distance, walking to nearby bus stops and fewer change of buses.

3. To find the shortest route to get from one bus stop to another taking into consideration distance, walking to nearby bus stops, fewer transfer between buses, expressway usage and crowding.

2. Literature Review

CS 360- Spring 2014 is a programming course at the University of Tennessee. The course instructors used Floyd Warshall's Algorithm to find the shortest path between two points. A presentation done by a teacher in the Massachusetts Institute of Technology says that Dijkstra's Algorithm works claims the algorithm works intuitively due to triangle inequality and the fact the subpath or the shortest path is itself a shortest path. A blog of a member of Cornell University states that Google Maps utilise Dijkstra's Algorithm to find the shortest way to get from one place to another.

3. Methodology and Study

3.1 Methodology

Dijkstra's Algorithm was used to assist the creation of the mathematical algorithm.

3.1.1 Dijkstra's Algorithm

Dijkstra's Algorithm finds the shortest path from only one point to all other points.

With reference to Figure 1a, the arrows indicate a direct route to get from one place to another. The number indicates the distance from one node to another. The green nodes refer to visited nodes.

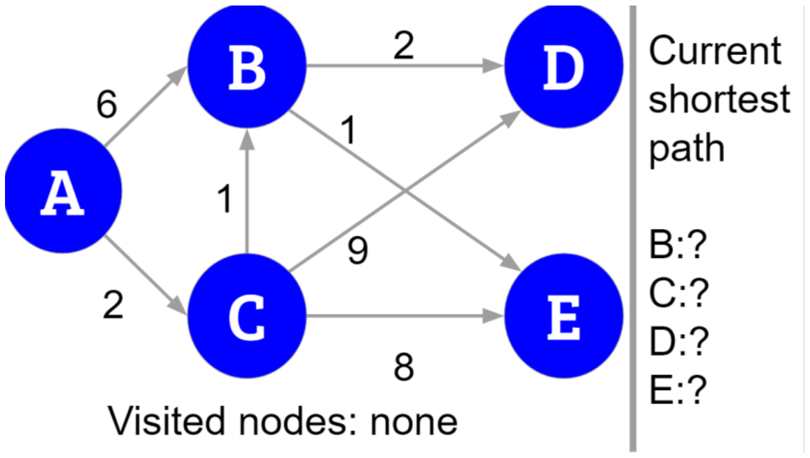


Figure 1a

It will start off by visiting the first node, A.

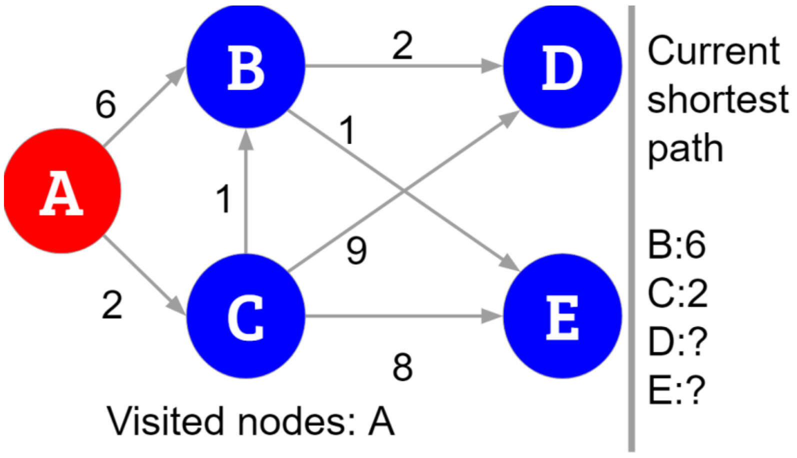


Figure 1b

The node in red is the starting point. From A the only places one can go to is to B and C. So for now, the shortest path to B is 6, to C is 2. From A to D and to E there isn’t an arrow linking them, indicating there is no direct route to those two places.

Since A is closer to C than B, C will be visited.

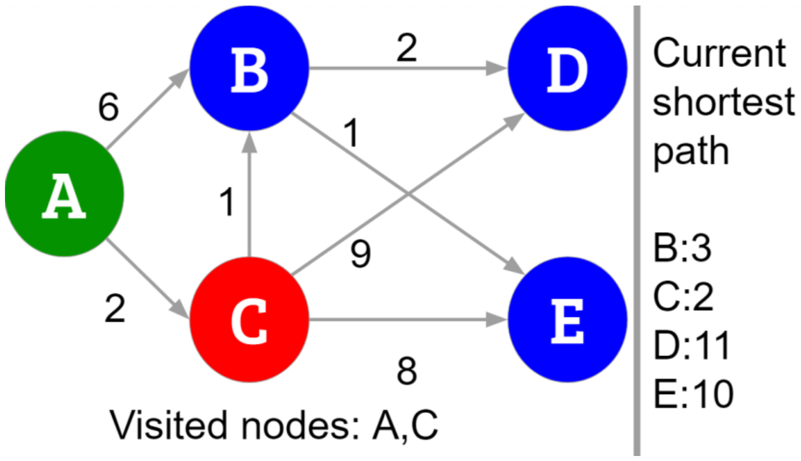


Figure 1c

From C, nodes B,D,E can be visited. The distance from A to B via the route from A -> C -> B will be checked. It is found that the distance is 3. This is shorter than from A -> B with no stopover. Now it is possible to get to D from A by passing through C. Getting from A to D using the route A -> C -> D, the distance travelled is 2+9, which is 11. That is the current shortest path from A to D. The same happens for A to E.

Now, the nearest unvisited node, B, will be visited.

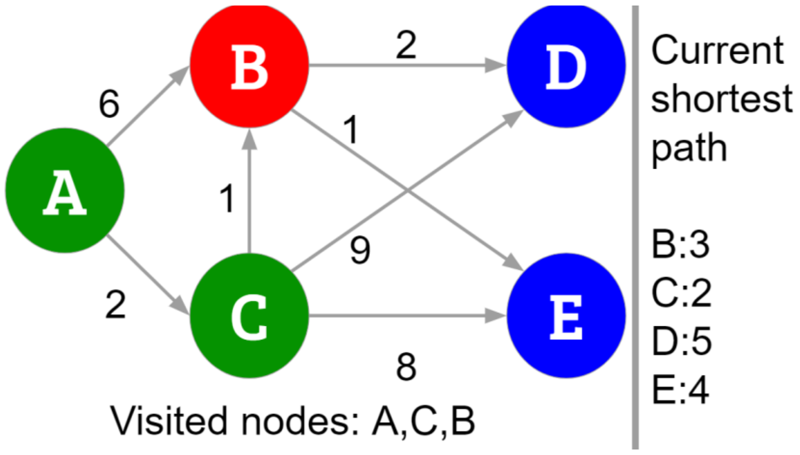


Figure 1d

From B, there are two direct routes to unvisited nodes (blue). The distance to get to D from A using this route A -> C -> B -> D will be calculated. The distance covered is 5. When compared to the previous route, which covered a distance of 11, this route is shorter. Thus, the shortest path from A to D has been replaced with this new path, and the distance is 5. To get to E, the route A -> C -> B -> E is used, producing a shorter route and the current shortest path from A to E will be updated.

From B, nodes D and E, which are unvisited, will be visited. From D, no unvisited node can be reached, resulting in no changes of any of the current shortest path, It is the same for E.

Since all the nodes have been visited, the shortest route from A to all other nodes is found.

3.2 Results

The programs created requires the user to enter the bus stop number of the original bus stop and intended bus stop destination.

3.2.1 Results to Research Question 1

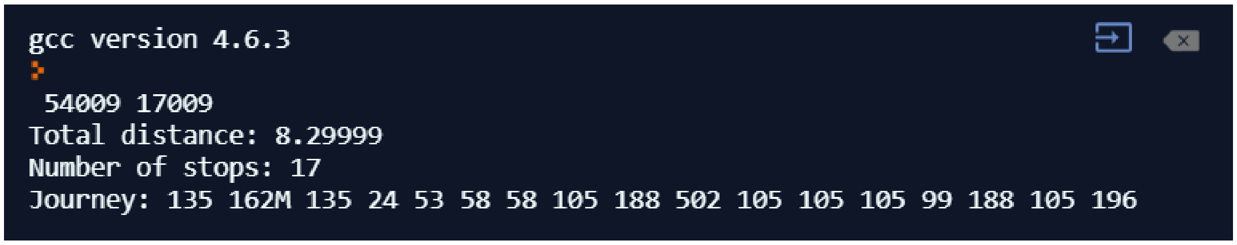
Dijkstra's Algorithm is applied into this algorithm to find the shortest way to get from one place to another.

The program will generate an output, for example:

"89 89 76 45 89"

The number of times a number appears consecutively refer to the number of stops one have to take on that bus. In this case, bus 89 will be taken for two stops before alighting, followed by 76 for one stop, then alight again then take bus 45 for one stop before alighting again and taking bus 89 again for one more stop.

The code is effective in doing its job. Take getting from Clementi to Ang Mo Kio as an example. There is a direct route to get from Clementi to Ang Mo Kio. That is by taking bus 166, which will travel a total of 28.6km. The program generated a shorter route.



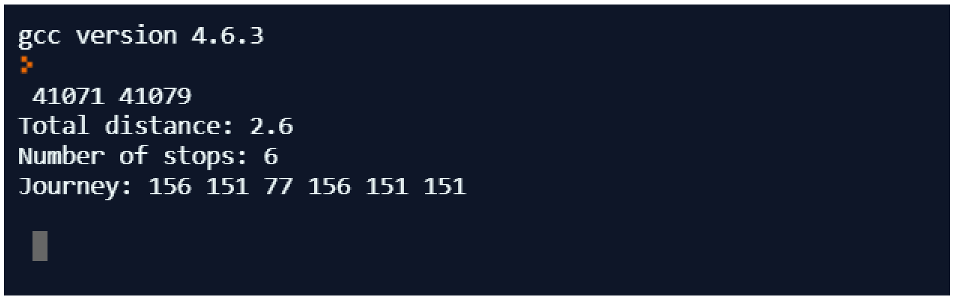
```
gcc version 4.6.3
 
 54009 17009
Total distance: 8.29999
Number of stops: 17
Journey: 135 162M 135 24 53 58 58 105 188 502 105 105 105 99 188 105 196
```

Figure 2

This journey will be shorter in distance.

However, the code does not account for walking between close bus stops, resulting in certain routes being very inefficient.

An example is the shortest route to get from Opposite National JC bus stop to National JC bus stop.



```
gcc version 4.6.3
 
 41071 41079
Total distance: 2.6
Number of stops: 6
Journey: 156 151 77 156 151 151
```

Figure 3

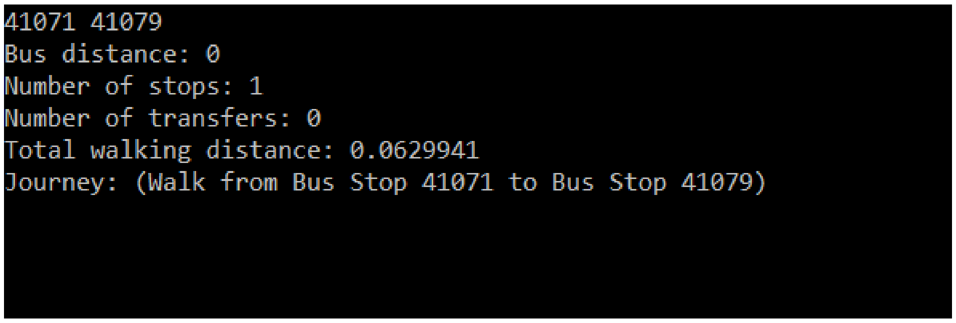
The program generates a route that travels 6 stops and take 5 different buses even though there is an overhead bridge connecting Opposite National JC bus stop to National JC bus stop.

Also, the number of different bus routes used is huge.

3.2.2 Results to Research Question 2

In order to address the above two problems, the program was modified. The program will now find bus stops within a 500-metre radius and check for a shorter route from a nearby stop to the targeted destination. This process will repeat till the shortest path is generated. Now, there is a limit on the number of times one can alight and take another bus (i.e. the number of transfers between buses is capped at a certain number)

This will generate a much more efficient route. Take the example of from Opposite National JC to National JC.



```
41071 41079
Bus distance: 0
Number of stops: 1
Number of transfers: 0
Total walking distance: 0.0629941
Journey: (Walk from Bus Stop 41071 to Bus Stop 41079)
```

Figure 4

The program now gives an instruction to walk to National JC bus stop, making the route much more efficient and shorter.

The user is given the choice to walk to nearby bus stops, allowing for a greater range of bus services being opened up for usage by the user.

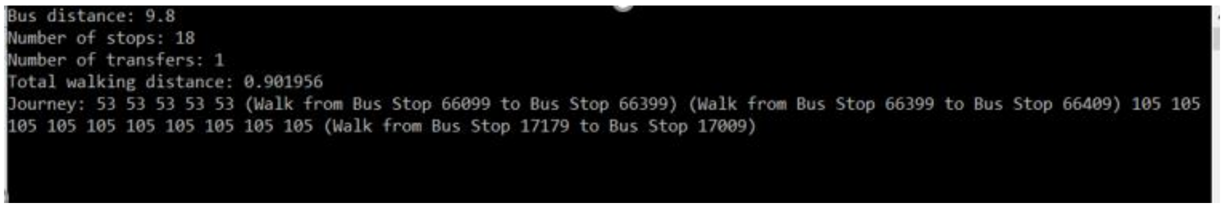
However, the bus arrival time is not accounted for in this program, as well as the previous program. Also, one’s walking speed and the feasibility of walking from a certain bus stop to another is not accounted for. The program made certain assumptions as the necessary information is not provided by LTA and other sources.

3.2.3 Results to Research Question 3

In the third research question, expressway usage and crowding were taken into consideration.

Buses which utilised expressways were recorded down. During peak hour, there would usually be traffic jam, thus the "score" for going by that route is increased, making it less likely for the user to be recommended to use that route. During non-peak hour, the "score" will be reduced, making it more likely for the user to be recommended to use that route.

The number of tap ins and tap outs at all bus stops around Singapore was analysed and bus stops with a huge passenger influx were identified. These bus stops would be deemed as "crowded". In the algorithm, this will affect the algorithm's recommended route, making it less likely for the user to be recommended to use that route. The data was also used to infer how many road users will there be on the road to help determine which roads are "congested".



```
Bus distance: 9.8
Number of stops: 18
Number of transfers: 1
Total walking distance: 0.901956
Journey: 53 53 53 53 53 (Walk from Bus Stop 66099 to Bus Stop 66399) (Walk from Bus Stop 66399 to Bus Stop 66409) 105 105
105 105 105 105 105 105 105 105 (Walk from Bus Stop 17179 to Bus Stop 17009)
```

Figure 5

Based on what time is it, the algorithm can produce different results to assist the user in avoiding congested roads and crowded bus stops.

The distance for each walk between bus stops have been reduced from the original 500m in RQ2 to 400m in RQ3, reducing the need for the user to walk long distances.

4. Conclusion

Dijkstra’s Algorithm provides the user with the shortest route to get from one bus stop to another. However, the lack of appropriate data resulted in many limitations and assumptions, causing certain inaccuracies. Even though the original aims are fulfilled, there is more potential in the project. More factors such as live bus arrival timing could be added.

5. References

Presentation on Dijkstra Algorithm. Retrieved June 29, 2018 from http://www-math.mit.edu/~rothvoss/18.304.3PM/Presentations/1-Melissa.pdf

Lecture 23: All Pairs Shortest Paths - Floyd-Warshall Algorithm. Retrieved June 29, 2018 from http://faculty.ycp.edu/~dbabcock/PastCourses/cs360/lectures/lecture23.html

Bus arrival information achieves 95% accuracy (16 July 2015). Retrieved June 29, 2018 from https://www.lta.gov.sg/apps/news/page.aspx?c=2&id=f8e78c77-b89b-4c88-9f1d-c4b0ce4dcfea

Dynamic Datasets. (n.d.). Retrieved June 21, 2018, from https://www.mytransport.sg/content/mytransport/home/dataMall/dynamic-data.html

M. (n.d.). Your Public Transit Guide. Retrieved July 2, 2018, from https://moovit.com/

TransitLink eGuide - Bus Enquiry. (n.d.). Retrieved July 2, 2018, from https://www.transitlink.com.sg/eservice/eguide/service_idx.php

Calculating the distance between 2 latitudes and longitudes that are saved in a text file? (n.d.). Retrieved August 13, 2018, from https://stackoverflow.com/questions/10198985/calculating-the-distance-between-2-latitudes-and-longitudes-that-are-saved-in-a

Appendix

Appendix A

Algorithm 1

```
#include <bits/stdc++.h>

#define tlen 24336

using namespace std;

class comp {

public:

    bool operator()(pair<pair<float, int>, vector<string> >
A,pair<pair<float, int>, vector<string> > B) {

        if (abs(A.first.first-B.first.first)<0.01) {

          if (A.second.rbegin()[1]==A.second.back()) {

            return false;

          }

          else return true;

        }

        else return A.first.first>B.first.first;

    }

};

int main() {

  int b, c, k=0;

  float a;

  vector<pair<float, pair<int, string> > > d[tlen];

  int e[tlen]={};

  float f[tlen+1]={};

  string g[tlen+1]={};

  int h[tlen+1]={};

  for (int i=0;i<tlen;i++) {

    cin>>f[i]>>g[i]>>h[i];

    //cout << f[i]<<" "<<g[i]<<" "<<h[i]<<"\n";
  }
  for (int i=0;i<tlen;i++) {
    if (find(e, e+tlen, h[i])>=e+tlen) {
      e[k]=h[i];
      k++;
    }
    if (f[i+1]>=f[i]) {
      if (find(e, e+tlen, h[i+1])>=e+tlen) {
        e[k]=h[i+1];
        k++;
      }
      d[find(e, e+tlen, h[i])-e].push_back(make_pair(f[i+1]-
f[i],make_pair(find(e, e+tlen, h[i+1])-e,g[i])));
    }
    if (i>0) {
      if (f[i]>=f[i-1]) {
        d[find(e, e+tlen, h[i])-e].push_back(make_pair(f[i]-f[i-
1],make_pair(find(e, e+tlen, h[i-1])-e,g[i])));
      }
    }
  }
  /*for (int i=0;i<k;i++) {
    cout << e[i] << ":\n";
    for (pair<float, pair<int, string> > &j : d[i]) {
      cout << j.first << " " << j.second.first << " " << j.second.second
<< "\n";
    }
    cout << "\n";
  }*/
  int l;
  while (cin>>c>>l) {
    float dist[k];
    vector<string> n;

    vector<string> s;

    fill(dist, dist+k, -1);

    priority_queue<pair<pair<float, int>, vector<string> >,
vector<pair<pair<float, int>, vector<string> > >, comp > pq;

    pq.push(make_pair(make_pair(0,find(e, e+tlen, c)-e), s));

    dist[find(e, e+tlen, c)-e]=0;

    while (!pq.empty()) {

      a=pq.top().first.first;

      b=pq.top().first.second;

      vector<string> m=pq.top().second;

      pq.pop();

      if (e[b]==l) {

        n=m;

        break;

      }

      if((dist[b]!=-1)&&(dist[b]<a)) continue;

      for (pair<float, pair<int, string> > &i : d[b]) {

        if ((dist[i.second.first]==-
1)||(dist[b]+i.first<dist[i.second.first])) {

          dist[i.second.first]=dist[b]+i.first;

          vector<string> o=m;

          o.push_back(i.second.second);

          pq.push(make_pair(make_pair(dist[i.second.first],
i.second.first), o));

        }

      }

    }

    cout <<"Total distance: "<< dist[find(e, e+tlen, l)-e]<<"\n";

    cout <<"Number of stops: "<< n.size() <<"\n";

    cout << "Journey: ";

    for (int i=0;i<n.size();i++) {
      cout << n[i] << " ";
    }
    cout << "\n\n";
  }
}
```

Algorithm 2

```
#include <bits/stdc++.h>

#define tlen 24336

#define slen 57916

using namespace std;

class comp {

public:

    bool operator()(pair<pair<double, int>, pair<pair<double, int>,
vector<string> > > A,pair<pair<double, int>, pair<pair<double, int>,
vector<string> > > B) {

        return A.first.first>B.first.first;

    }

};

inline void djikstra(int k, int c, int l, int e[tlen], vector<pair<double,
pair<int, string> > > d[tlen]) {

      double dist[k];

      double a, xa;

      int b, xb;

    vector<string> n;

    vector<string> s;

    fill(dist, dist+k, -1);

    priority_queue<pair<pair<double, int>, pair<pair<double, int>,
vector<string> > >, vector<pair<pair<double, int>, pair<pair<double, int>,
vector<string> > > >, comp > pq;

    pq.push(make_pair(make_pair(0,find(e, e+tlen, c)-e),
make_pair(make_pair(0,0), s)));

    dist[find(e, e+tlen, c)-e]=0;

    while (!pq.empty()) {

      a=pq.top().first.first;

      b=pq.top().first.second;

        xa=pq.top().second.first.first;

        xb=pq.top().second.first.second;

      vector<string> m=pq.top().second.second;
      pq.pop();
      if (e[b]==1) {
        n=m;
        break;
      }
      if((dist[b]!=-1)&&(dist[b]<a)) continue;
      for (int j=0;j<d[b].size();j++) {
           pair<double, pair<int, string> > i=d[b][j];
           if (i.second.second[0]=='(') {
                 if ((dist[i.second.first]==-
1)||(dist[b]+i.first<dist[i.second.first])) {
                   dist[i.second.first]=dist[b]+i.first;
                   vector<string> o=m;
                   o.push_back(i.second.second);
                   pq.push(make_pair(make_pair(dist[i.second.first],
i.second.first), make_pair(make_pair(xa+i.first,xb), o)));
                 }
           }
           else if (!m.empty()){
                 if (m.back()!=i.second.second) {
                       if ((dist[i.second.first]==-
1)||(dist[b]+i.first+4<dist[i.second.first])) {
                         dist[i.second.first]=dist[b]+i.first+4;
                         vector<string> o=m;
                         o.push_back(i.second.second);
                         pq.push(make_pair(make_pair(dist[i.second.first],
i.second.first), make_pair(make_pair(xa,xb+1), o)));
                       }
                 }
                 else if ((dist[i.second.first]==-
1)||(dist[b]+i.first<dist[i.second.first])) {
                   dist[i.second.first]=dist[b]+i.first;
                   vector<string> o=m;
```

```
                    o.push_back(i.second.second);
                    pq.push(make_pair(make_pair(dist[i.second.first],
i.second.first), make_pair(make_pair(xa,xb), o)));
                  }
            }
            else {
              dist[i.second.first]=dist[b]+i.first;
              vector<string> o=m;
              o.push_back(i.second.second);
              pq.push(make_pair(make_pair(dist[i.second.first],
i.second.first), make_pair(make_pair(xa,xb), o)));
            }
      }
    }
    cout <<"Bus distance: "<< dist[find(e, e+tlen, l)-e]-xb*4-xa<<"\n";
    cout <<"Number of stops: "<< n.size() <<"\n";
      cout <<"Number of transfers: "<< xb<<"\n";
      cout <<"Total walking distance: "<< xa/5.6<<"\n";
    cout << "Journey: ";
    for (int i=0;i<n.size();i++) {
      cout << n[i] << " ";
    }
    cout << "\n\n";
}
int main() {
  int b, c, k=0;
  double a;
  vector<pair<double, pair<int, string> > > d[tlen];
  int e[tlen]={};
  double f[tlen+1]={};
  string g[tlen+1]={};
  int h[tlen+1]={};
  ifstream in("rq2in.txt");
  streambuf *cinbuf = cin.rdbuf();
  cin.rdbuf(in.rdbuf());
  for (int i=0;i<tlen;i++) {
    cin>>f[i]>>g[i]>>h[i];
    //cout << f[i]<<" "<<g[i]<<" "<<h[i]<<"\n";
  }
  for (int i=0;i<tlen;i++) {
    if (find(e, e+tlen, h[i])>=e+tlen) {
      e[k]=h[i];
      k++;
    }
    if (f[i+1]>=f[i]) {
      if (find(e, e+tlen, h[i+1])>=e+tlen) {
        e[k]=h[i+1];
        k++;
      }
      d[find(e, e+tlen, h[i])-e].push_back(make_pair(f[i+1]-
f[i],make_pair(find(e, e+tlen, h[i+1])-e,g[i])));
    }
    if (i>0) {
      if (f[i]>=f[i-1]) {
        d[find(e, e+tlen, h[i])-e].push_back(make_pair(f[i]-f[i-
1],make_pair(find(e, e+tlen, h[i-1])-e,g[i])));
      }
    }
  }
  double xa;
  int xb;
  int xc;
  for (int i=0;i<slen;i++) {
    cin >> xa >> xb >> xc;
```

```
      if (find(e, e+tlen, xb)<e+tlen) {
            stringstream ss;
        ss<<"(Walk from Bus Stop "<< xb<<" to Bus Stop "<<xc<<")";
        d[find(e, e+tlen, xb)-e].push_back(make_pair(xa,make_pair(find(e,
e+tlen, xc)-e,ss.str())));
    }
  }
  /*for (int i=0;i<k;i++) {
    cout << e[i] << ":\n";
    for (pair<double, pair<int, string> > &j : d[i]) {
      cout << j.first << " " << j.second.first << " " << j.second.second
<< "\n";
    }
    cout << "\n";
  }*/
  cin.rdbuf(cinbuf);
  int l;
  while (cin>>c>>l) {
    djikstra(k,c,l,e,d);
  }
}
```

Algorithm 3

Data sorting algorithms:

Collecting bus stop numbers:

1.
```
mystr="""


"""
data=mystr.split("\n")
abc=""
for i in range(len(data)):
if "•" not in data[i]:
continue
 abc+="=SPLIT(\""
li=data[i].split()
   abc+=li[1]
   abc+=","
   abc+=data[i].split("• ")[1]
   abc+="\", \",\")\n"
print abc
```

2.
```
  #include <bits/stdc++.h>
using namespace std;
int main(){
  int a=0,b=0;
  char arr[10000];
  string name;
 while(getline(cin,name)){
    a=0;
   b=0;

   cout<<"=SPLIT(\"";
for(int i=0;i<name.size();i++){
if (name[i]=='-')a=1;
if (name[i]=='('){b=1;cout<<",";}
if (name[i]==')'){cout<<"\", \",\")"<<endl;break;}
if (a==1&&name[i]!='('&&name[i]!=')'&&name[i]!='-')cout<<name[i];}



 }

}
```

3.
```
#include <bits/stdc++.h>
using namespace std;
int main(){
            int a=0,b=0;
            char arr[10000];
            string name;
  while(getline(cin,name)){
                        a=0;
    b=0;

    cout<<"=SPLIT(\"";
    string c=(name.substr(name.find(" (") + 2));
    cout<<c.substr(0, c.find(")"));
    cout << ",";
    string d=(name.substr(name.find(" - ") + 3));
    cout<<d.substr(0, d.find(" ("));
    cout<<"\", \",\")\n";




  }

}
```

Sorting number of trips between 2 bus stops:

1.
```
#include <bits/stdc++.h>
using namespace std;
char maps[99];

int main() {
      ifstream cin("new3.txt");
      ofstream cout("new2.txt");

  int r,n;
  int len;
   while (0==0)
    { n=0;r=0;
      cin.getline(maps,99,'\n');
      if (maps[0]=='_'){break;}
      len=strlen(maps);

                   for (int i=len-1;i>0;i--)
                   {{n++;}
                   if (maps[i]==','){r++;}
                    if(r==1) break;
                   }

if (n>=4)
{for (int i=0;i<len;i++){cout<<maps[i];
}cout<<"\n";
}


                                                            }
                                }
```

Sorting number of tap ins and tap outs in each bus stop:

1.
```
#include <bits/stdc++.h>
using namespace std;
char maps[99];

int main() {
            ifstream cin("book1.txt");
            ofstream cout("book2.txt");

  int r,n;
  char p;
  int len;
   while (0==0)
    { n=0;r=0;p=0;
      cin.getline(maps,99,'\n');
      if (maps[0]=='_'){break;}
      len=strlen(maps);
                        for (int i=len-1;i>0;i--)
                        {{n++;}

                        if (maps[i]==','){r++;}
                         if(r==1) break;
                         if(n==4){p=maps[len-n];
                         }
                        }
if (n==5&&p!='1'&&p!='4'&&p!='5'&&p!='6'&&p!='7'&&p!='8'&&p!='9')
{for (int i=0;i<len;i++){cout<<maps[i];
}cout<<"\n";
}                                                              }
                                    }
```