Algorithms analysis	Section	01
	Student number	22000080
Project Graph Search	Name	Kim, Min Chae

If your explanation is less informative and insufficient, then you may not get any points.
Also, you should provide discussion, otherwise you will get penalty.

Item	Your answer
Problem (1 or 2)	1
Project Title	Shortest time between Busan subway stations
Want to present? (Yes or No)	No
Num. of lines in your code	133
Num. of functions in your code	1
Key algorithm	Dijkstra's algorithm

// Project Title: write appropriate name or just put 'puzzle game' if you solve problem 2.
// Key algorithm: Use only several words. Ex1) Breath First Search Ex2) Kruskal's Algorithm

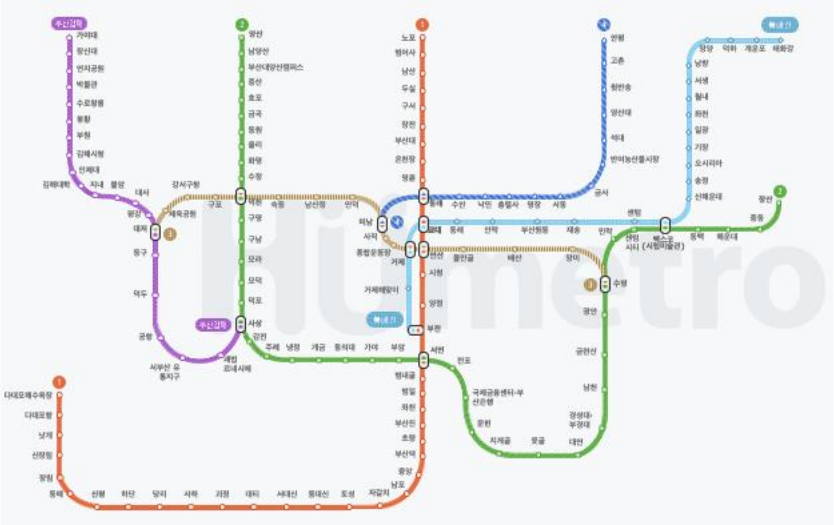
□ If you solve problem 2 then write the detail information below.

- Problem Definition

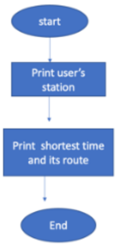
  Busan, my hometown, has several subway lines and several stations where I can transfer. At this time, when several subway stations overlap, the time taken varies depending on which subway line is taken. Therefore, an algorithm is needed to indicate at which station it takes the shortest time to transfer.

- Goal

  After showing the transfer station list to the user, the station is input as a number. After that, it shows the shortest distance from the transfer station to another transfer station and the time it takes. The reason why all stations are not targeted is that they cannot change routes. The time it takes between subway stations was provided on the official Busan subway site. (Humetro, https://www.humetro.busan.kr/homepage/cyberstation/map.do)



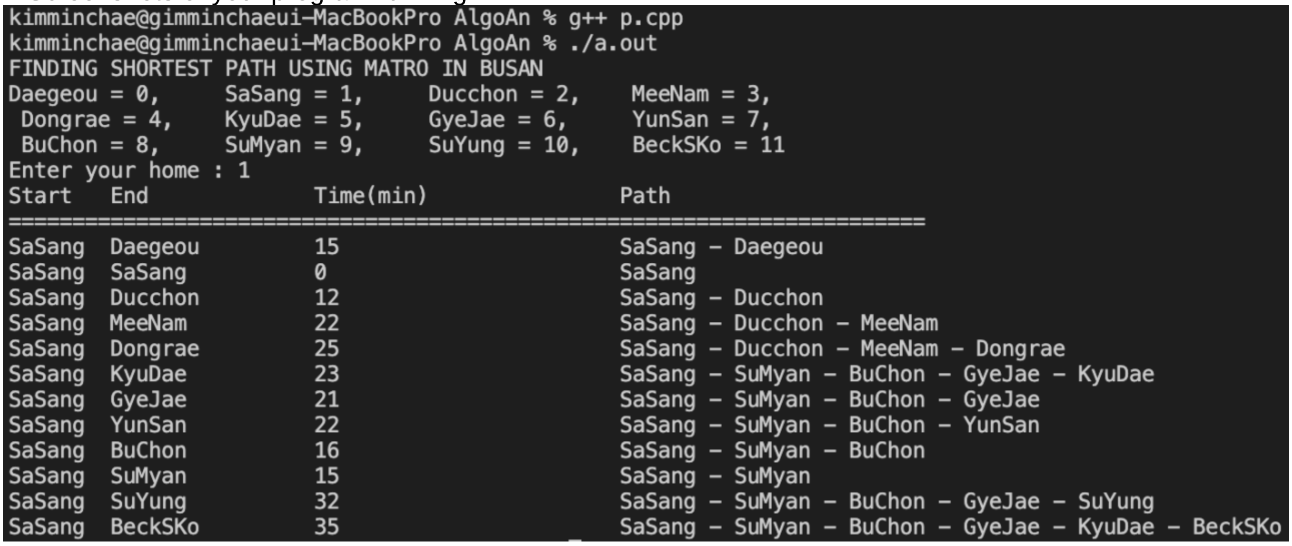
□ Flowchart of your algorithm



□ Describe your algorithm

Find the shortest-time station W of V-S, put it in S, and then update the time for all stations of V-S of W until now. At this time, if the time so far + each time is less than the original time, renew it. Assuming that there is no path if -1 is stored in the queue, push whenever the path is found and push again from beginning to end if there is a shorter time. Initially, it is assumed that the path to itself has already been investigated by making S true. If the start vertex and the I vertex can be connected, the next goal is i. First, take the shortest distance based on the starting point and store it in D[i]. At this time, the length of the edge that cannot be connected (in this case, time) should be infinite, and it is assumed that a very large number can be put in, but if the time that is not oneself is zero, it cannot be connected. Then, if W is set and there is a path to the w vertex, compare and temporarily store the results so far in the queue. Then, perform a shorter comparison and completely replace p[i] with p[w]+(w,i) edges. Then, the result is produced by POPing the result p[i] in order.

□ Screenshots of your program running



```
kimminchae@gimminchaeui-MacBookPro AlgoAn % g++ p.cpp
kimminchae@gimminchaeui-MacBookPro AlgoAn % ./a.out
FINDING SHORTEST PATH USING MATRO IN BUSAN
Daegeou = 0,     SaSang = 1,     Ducchon = 2,     MeeNam = 3,
 Dongrae = 4,    KyuDae = 5,     GyeJae = 6,      YunSan = 7,
 BuChon = 8,     SuMyan = 9,     SuYung = 10,     BeckSKo = 11
Enter your home : 1
Start   End             Time(min)               Path
========================================================================
SaSang  Daegeou         15                      SaSang - Daegeou
SaSang  SaSang          0                       SaSang
SaSang  Ducchon         12                      SaSang - Ducchon
SaSang  MeeNam          22                      SaSang - Ducchon - MeeNam
SaSang  Dongrae         25                      SaSang - Ducchon - MeeNam - Dongrae
SaSang  KyuDae          23                      SaSang - SuMyan - BuChon - GyeJae - KyuDae
SaSang  GyeJae          21                      SaSang - SuMyan - BuChon - GyeJae
SaSang  YunSan          22                      SaSang - SuMyan - BuChon - YunSan
SaSang  BuChon          16                      SaSang - SuMyan - BuChon
SaSang  SuMyan          15                      SaSang - SuMyan
SaSang  SuYung          32                      SaSang - SuMyan - BuChon - GyeJae - SuYung
SaSang  BeckSKo         35                      SaSang - SuMyan - BuChon - GyeJae - KyuDae - BeckSKo
```

□ Discussion about the results

// Please discuss about your algorithm, any trouble you encountered, limitation of your algorithm,
// any possible better solution (algorithm). Please share what you feel, what you learn from this project.

It is said that using the priority queue can be made easier, but I did not use it here because I did not have the ability to use the priority. In addition, there were cases where the inf was set and written as #deine10000 and so on, but because I was programming for a variable called time that could be increased indefinitely, I chose another method and chose a method to make the inf zero. At first, I tried to show the shortest route between the stations I chose, but I thought it would be more practical to store and use the information after showing all the results from the transfer station where I live. Next time, I would like to write a program to derive the shortest distance route and time for Seoul subway lines with more transfer stations using the priority queue.

□ Codes
// Please copy & paste you code here.
// You should also submit the separate executable C or C++ files, TA will try run your code.

```
#include <iostream>
#include <queue>
using namespace std;

#define MAX 12

int graph[MAX][MAX] = { { 0, 15, 8, 0, 0, 0, 0, 0, 0, 0, 0, 0 }, //Daegeou
                        { 15, 0, 12, 0, 0, 0, 0, 0, 0, 15, 0, 0 }, //SaSang
                        { 8, 12, 0, 10, 0, 0, 0, 0, 0, 0, 0, 0 }, //Ducchon
                        { 0, 0, 10, 0, 3, 0, 7, 0, 0, 0, 0, 0 }, //MeeNam
                        { 0, 0, 0, 3, 0, 2, 0, 0, 0, 0, 0, 0 }, //Dongrae
                        { 0, 0, 0, 0, 0, 0, 2, 2, 0, 0, 0, 12 }, //KyuDae
                        { 0, 0, 5, 5, 0, 2, 0, 2, 0, 0, 11, 0 }, //GyeJae
                        { 0, 0, 0, 0, 0, 2, 2, 0, 6, 0, 10, 0 }, //YunSan
                        { 0, 0, 0, 0, 0, 0, 5, 6, 0, 1, 0, 0 }, //BuChon
                        { 0, 15, 0, 0, 0, 0, 0, 0, 1, 0, 20, 0 }, //SuMyan
                        { 0, 0, 0, 0, 0, 0, 11, 0, 0, 20, 0, 6 }, //SuYung
                        { 0, 0, 0, 0, 0, 12, 0, 0, 0, 0, 6, 0 } }; //BeckSKo

string name[MAX] = {"Daegeou", "SaSang", "Ducchon", "MeeNam", "Dongrae",
                    "KyuDae", "GyeJae", "YunSan", "BuChon", "SuMyan",
                    "SuYung", "BeckSKo"};

void dijkstra(int start){
    bool S[MAX] = { 0 }; //already find : true
    int D[MAX];
    queue<int> P[MAX];

    S[start] = true; //if start and end is equal, time is 0

    for (int i = 0; i < MAX; ++i) {
        if(graph[start][i] != 0) P[i].push(i);
    }

    for (int i = 0; i < MAX; ++i) {
        D[i] = graph[start][i];
    }

    for (int j = 1; j < MAX; ++j) {
        int w = start;

        for (int i = 0; i < MAX; ++i) {
            if (D[i] != 0 && !S[i]) {
                if (D[w] == 0 || D[w] > D[i]) w = i;
            }
        }

        S[w] = true;

        for (int i = 0; i < MAX; ++i) {
            if (graph[w][i] != 0 && !S[i]) {
                if ((D[i] == 0) || (D[i] > D[w] + graph[w][i])) {
                    if (D[w] != 0) {

                        if (!P[i].empty()) {
                            queue<int> pre_path = P[i];
                            int prev, next;
                            int pre_d = 0, post_d = 0;

                            prev = start;

                            while (!pre_path.empty()) {
                                next = pre_path.front();
                                pre_d += graph[prev][next];
                                prev = next;
                                pre_path.pop();
                            }

                            queue<int> post_path = P[w];
                            prev = start;

                            while (!post_path.empty()) {
                                next = post_path.front();
                                post_d += graph[prev][next];
                                prev = next;
                                post_path.pop();
                            }

                            post_d += graph[w][i];

                            if (pre_d > post_d) {
                                post_path = P[w];
                                post_path.push(i);
                                P[i] = post_path;
                            }
                        }
                        else {
                            P[i] = P[w];
                            P[i].push(i);
                        }
                        D[i] = D[w] + graph[w][i];
                    }
                }
            }
        }
    }

    //print
    cout << "Start\tEnd\t\tTime(min)\t\tPath" << endl;
    cout <<
"================================================================================" <<
endl;
```

```
    for (int i = 0; i < MAX; ++i) {
        cout << name[start] << "\t" << name[i] << "\t\t" << D[i] << "\t\t\t";

        if (!P[i].empty()) {
            queue<int> temp = P[i];
            cout << name[start];

            do {
                cout << " - " << name[temp.front()];
                temp.pop();
            } while (!temp.empty());

            cout << endl;
        }
        else {
            if (i != start) cout << "not existing Path" << endl;
            else cout << name[start] << endl;
        }
    }
}

int main(){
    cout << "FINDING SHORTEST PATH USING MATRO IN BUSAN" << endl;
    cout << "Daegeou = 0, \t SaSang = 1, \t Ducchon = 2, \t MeeNam = 3, \n Dongrae = 4, \t KyuDae
= 5, \t GyeJae = 6, \t YunSan = 7, \n BuChon = 8, \t SuMyan = 9, \t SuYung = 10, \t BeckSKo = 11" <<
endl;
    cout << "Enter your home : ";
    int home;
    cin >> home;
    dijkstra(home);

    return 0;
}
```