UNIVERSIDAD NACIONAL DE SAN AGUSTÍN DE AREQUIPA

FACULTAD DE PRODUCCION Y SERVICIOS

ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS

Curso: Laboratorio de Análisis y Diseño de Algoritmos

Práctica 11

Estudiante: Tapara Quispe, Fabiola Grissel

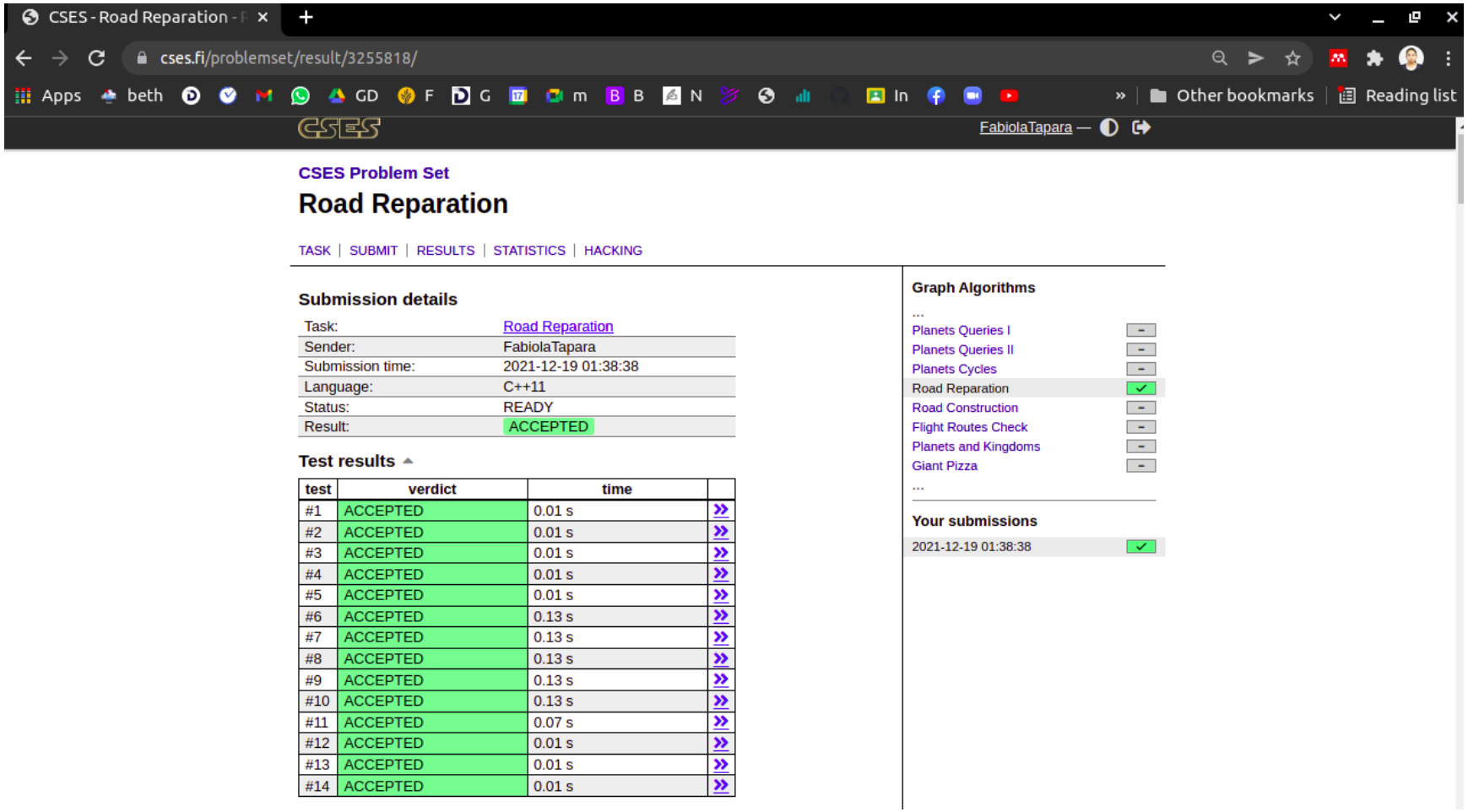
Docente: Alex Josue Florez Farfan

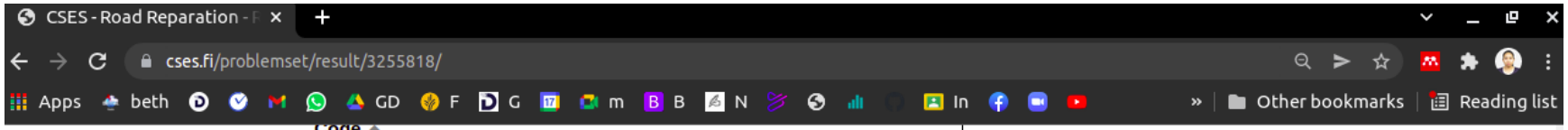
Sección: “B”

Arequipa - Perú

Diciembre 2021

Ejercicio 1

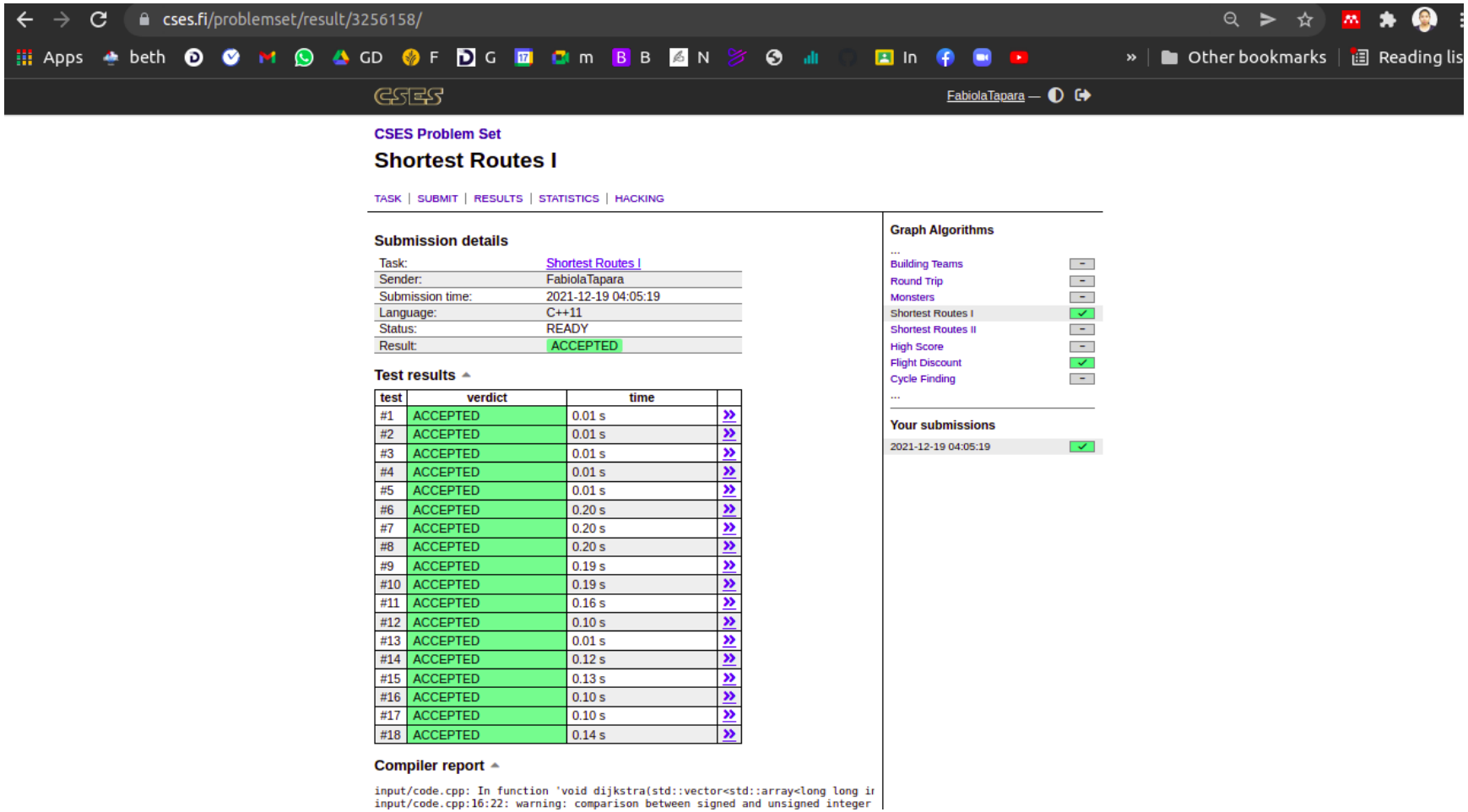

CSES - Road Reparation
cses.fi/problemset/result/3255818/
Other bookmarks
Reading list
FabiolaTapara
CSES Problem Set
Road Reparation
TASK | SUBMIT | RESULTS | STATISTICS | HACKING
Submission details
Task: Road Reparation
Sender: FabiolaTapara
Submission time: 2021-12-19 01:38:38
Language: C++11
Status: READY
Result: ACCEPTED
Test results
test verdict time
#1 ACCEPTED 0.01 s
#2 ACCEPTED 0.01 s
#3 ACCEPTED 0.01 s
#4 ACCEPTED 0.01 s
#5 ACCEPTED 0.01 s
#6 ACCEPTED 0.13 s
#7 ACCEPTED 0.13 s
#8 ACCEPTED 0.13 s
#9 ACCEPTED 0.13 s
#10 ACCEPTED 0.13 s
#11 ACCEPTED 0.07 s
#12 ACCEPTED 0.01 s
#13 ACCEPTED 0.01 s
#14 ACCEPTED 0.01 s
Graph Algorithms
...
Planets Queries I
Planets Queries II
Planets Cycles
Road Reparation
Road Construction
Flight Routes Check
Planets and Kingdoms
Giant Pizza
...
Your submissions
2021-12-19 01:38:38

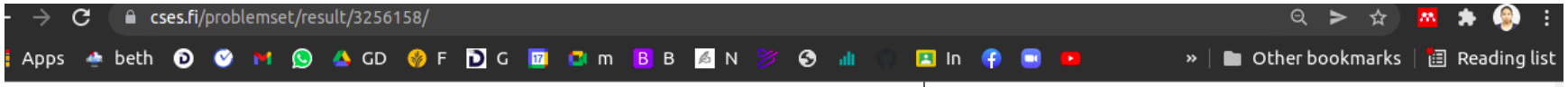


Code

```cpp
#include <bits/stdc++.h>
/*
 * Ejercicio 1 RoadReparation
 * Autor: Fabiola Tapara Quispe
 * Descripcion: Hay n ciudades y m carreteras entre ellas. Desafortunadame
 * se nos pide encontrar un a solución donde el costo total sea lo más pe
 * posible y asi reparar algunas de las carreteras para que haya una ruta
 * entre dos ciudades. Se utilizo el algoritmo de Kruskal
 * Fecha: 13/12/21
 */
#define int long long
#define MOD 1000000007
#define MAX 1e13

using namespace std;

vector<array<int, 3>> roads;//representa el grafo
int n, m;//variables globales n ciudades y m carreteras
bool visitado[100005];// Creamos un arreglo visitado que representara los
int path[100005]; // Creamos un arreglo path que representara los posible

int find(int x) {// recibe un Vertice de origen
    if(path[x] == x) {// En caso el valor en el Path para el Vertice 'x'
        return x;//retorna el mayor Vertice donde podemos llegar a partir
    }
    return path[x] = find(path[x]);// Si es diferente de x significa que
     //mas lejos desde el Vertice inicial y se vuelve a hacer la busqueda
}

void unionEdge(int x, int y) { //une los Vertices de Origen y Destino
    //guarda dentro del Path en caso que una dos puntos diferentes
    //No admite aristas que forme bucles

    int a = find(x);// Vertices mas alejados con la funcion find
    int b = find(y);// Vertices mas alejados con la funcion find
    if(a != b) {//no debe ser hacia el mismo Vertice porque seria un bucl
        path[a] = b;
    }
}

int MST() {
    for(int i = 0; i < n; i++) {
        path[i] = i;
        visitado[i] = false;
    }
    sort(roads.begin(), roads.end());
    int cost = 0;
```

Ejercicio 2

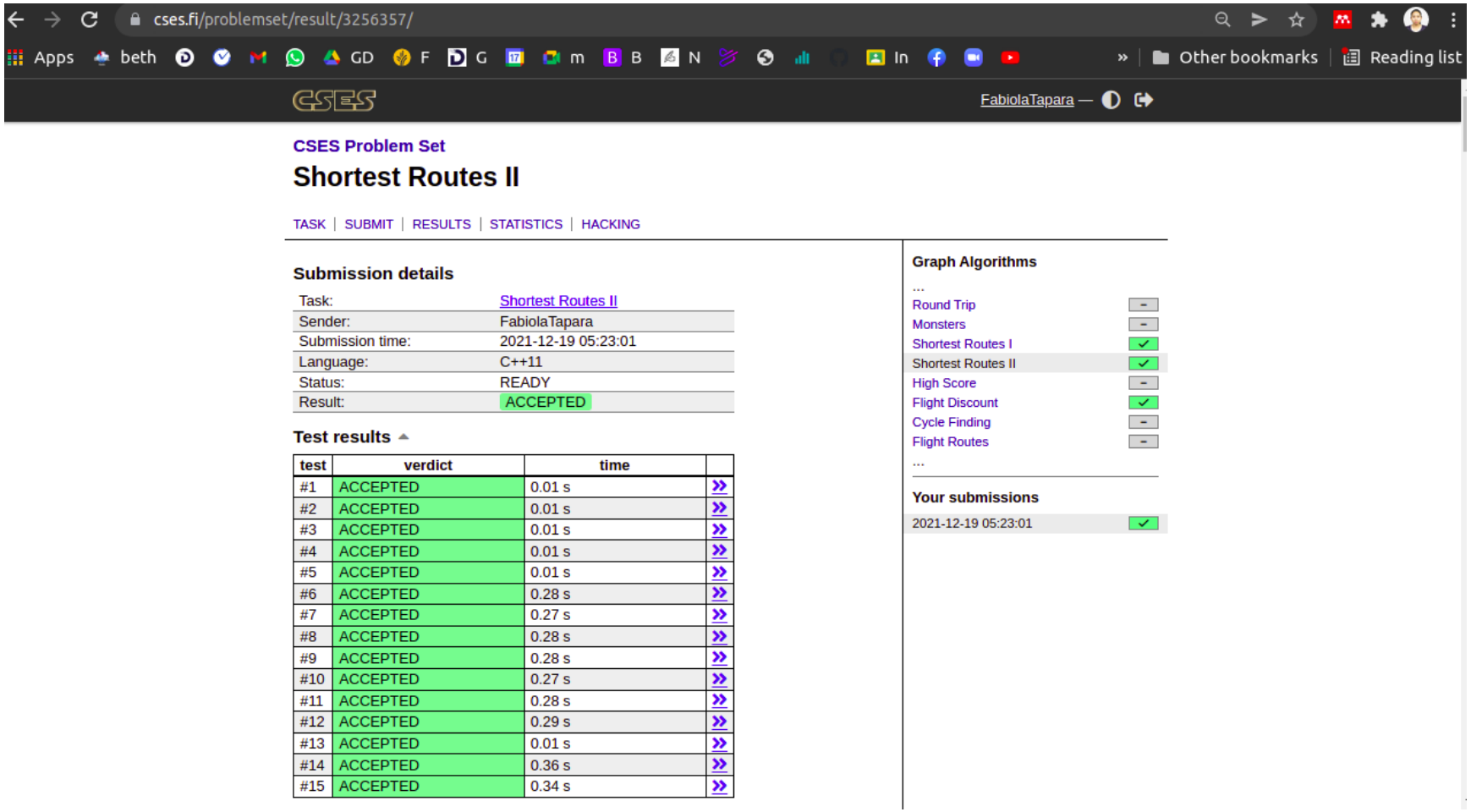

cses.fi/problemset/result/3256158/
FabiolaTapara
CSES Problem Set
Shortest Routes I
TASK | SUBMIT | RESULTS | STATISTICS | HACKING
Submission details
Task: Shortest Routes I
Sender: FabiolaTapara
Submission time: 2021-12-19 04:05:19
Language: C++11
Status: READY
Result: ACCEPTED
Test results
test | verdict | time
#1 ACCEPTED 0.01 s
#2 ACCEPTED 0.01 s
#3 ACCEPTED 0.01 s
#4 ACCEPTED 0.01 s
#5 ACCEPTED 0.01 s
#6 ACCEPTED 0.20 s
#7 ACCEPTED 0.20 s
#8 ACCEPTED 0.20 s
#9 ACCEPTED 0.19 s
#10 ACCEPTED 0.19 s
#11 ACCEPTED 0.16 s
#12 ACCEPTED 0.10 s
#13 ACCEPTED 0.01 s
#14 ACCEPTED 0.12 s
#15 ACCEPTED 0.13 s
#16 ACCEPTED 0.10 s
#17 ACCEPTED 0.10 s
#18 ACCEPTED 0.14 s
Compiler report
input/code.cpp: In function 'void dijkstra(std::vector<std::array<long long ir
input/code.cpp:16:22: warning: comparison between signed and unsigned integer
Graph Algorithms
Building Teams
Round Trip
Monsters
Shortest Routes I
Shortest Routes II
High Score
Flight Discount
Cycle Finding
Your submissions
2021-12-19 04:05:19

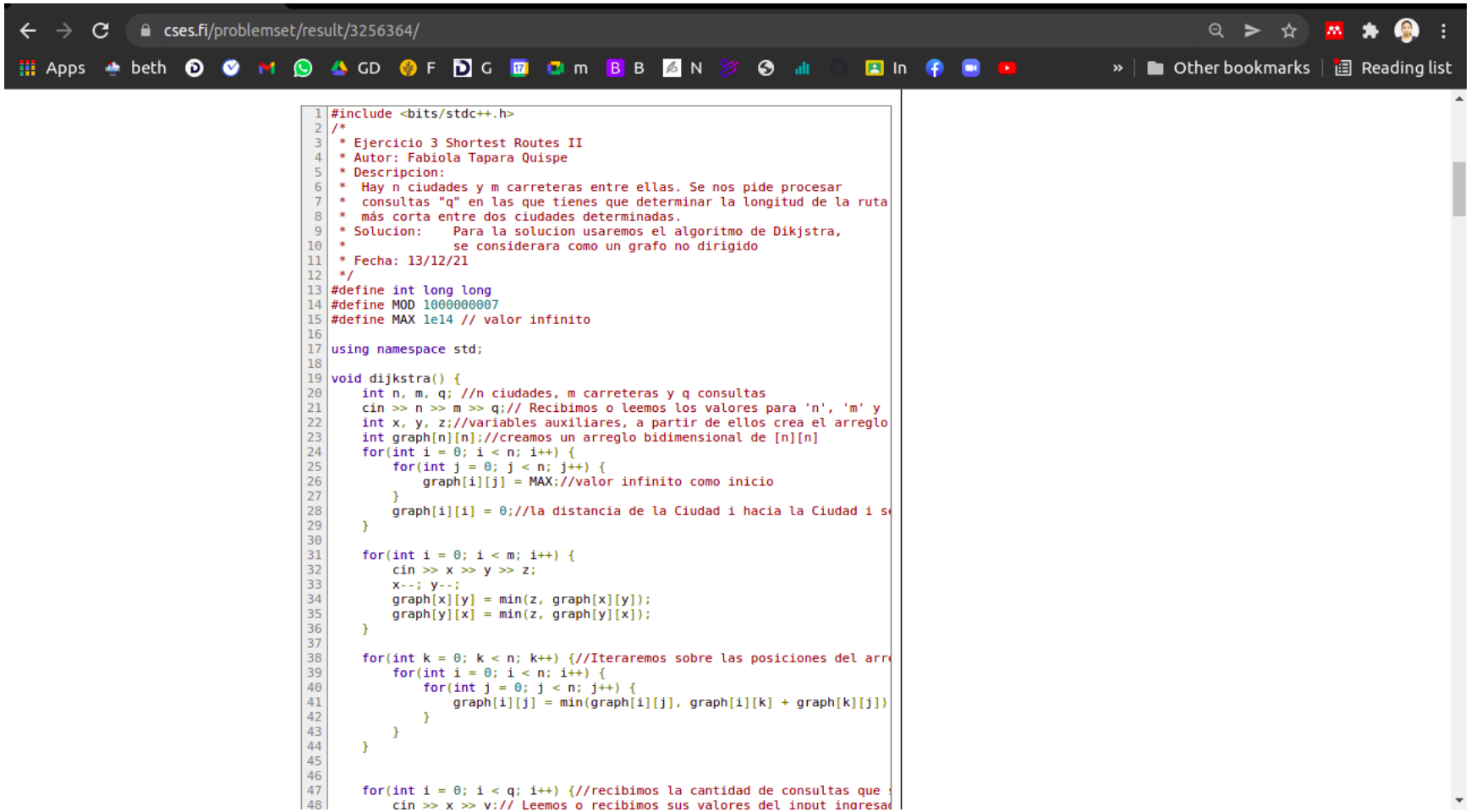


Code

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

#define int long long
#define MOD 1000000007
#define MAX 1e14

using namespace std;

vector<array<int, 2>> G[100005];
int n, m;

void dijkstra(vector<array<int, 2>> G[], int costs[], int root) {
    set<int> visited;

    priority_queue<array<int, 3>, vector<array<int, 3>>, greater<array<in
    for(int i = 0; i < G[root].size(); i++) {
        pq.push({G[root][i][1], root, G[root][i][0]});
    }

    for(int i = 0; i < n; i++) {
        costs[i] = MAX;
    }

    costs[root] = 0;
    visited.insert(root);

    while(!pq.empty()) {
        auto top = pq.top();
        pq.pop();
        if(visited.find(top[2]) == visited.end()) {
            costs[top[2]] = min(costs[top[2]], top[0]);
        }
        else continue;

        visited.insert(top[2]);
        for(int i = 0; i < G[top[2]].size(); i++) {
            pq.push({costs[top[2]] + G[top[2]][i][1], top[2], G[top[2]][i
        }
    }
}

void solve() {
    cin >> n >> m;
    int x, y, z;
    for(int i = 0; i < m; i++) {
        cin >> x >> y >> z;
        x--; y--;
        G[x].push_back({y, z});
    }

    int dist[n];
    dijkstra(G, dist, 0);

    for(int i = 0; i < n; i++) {
```

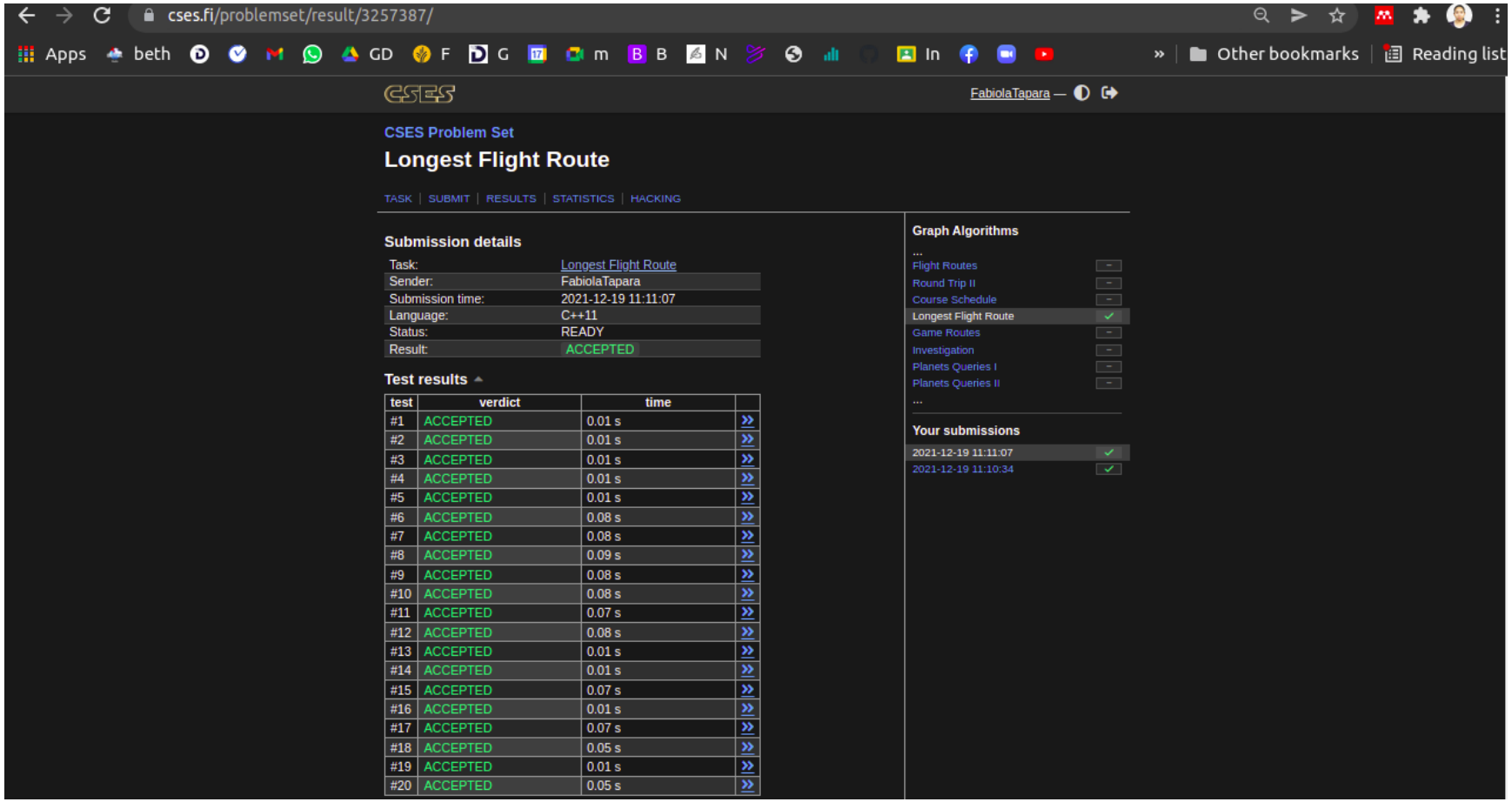
Ejercicio 3

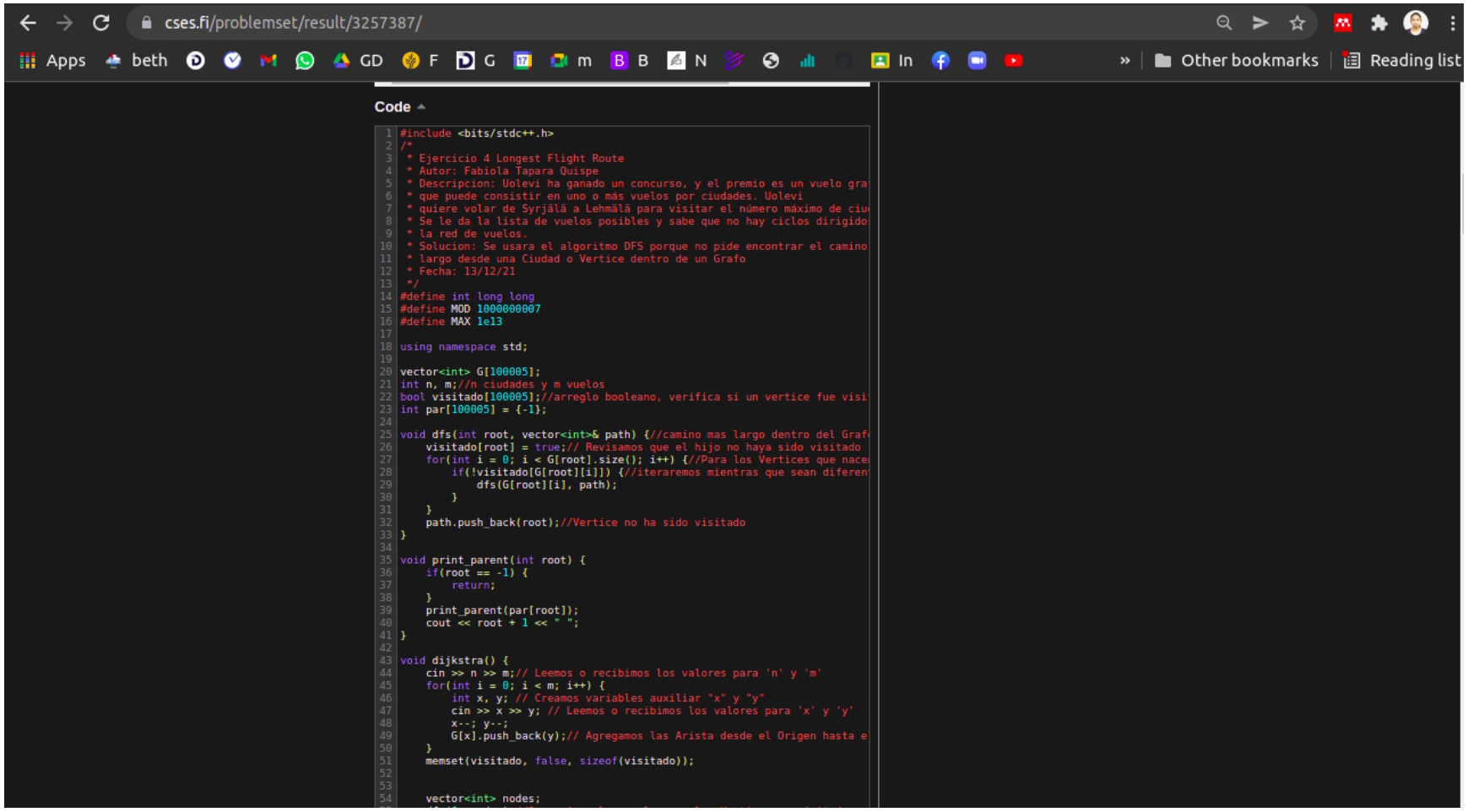

cses.fi/problemset/result/3256357/
Apps beth GD F G m B B N In Other bookmarks Reading list
CSES
FabiolaTapara
CSES Problem Set
Shortest Routes II
TASK | SUBMIT | RESULTS | STATISTICS | HACKING
Submission details
Task: Shortest Routes II
Sender: FabiolaTapara
Submission time: 2021-12-19 05:23:01
Language: C++11
Status: READY
Result: ACCEPTED
Test results
test | verdict | time
#1 ACCEPTED 0.01 s
#2 ACCEPTED 0.01 s
#3 ACCEPTED 0.01 s
#4 ACCEPTED 0.01 s
#5 ACCEPTED 0.01 s
#6 ACCEPTED 0.28 s
#7 ACCEPTED 0.27 s
#8 ACCEPTED 0.28 s
#9 ACCEPTED 0.28 s
#10 ACCEPTED 0.27 s
#11 ACCEPTED 0.28 s
#12 ACCEPTED 0.29 s
#13 ACCEPTED 0.01 s
#14 ACCEPTED 0.36 s
#15 ACCEPTED 0.34 s
Graph Algorithms
...
Round Trip
Monsters
Shortest Routes I
Shortest Routes II
High Score
Flight Discount
Cycle Finding
Flight Routes
...
Your submissions
2021-12-19 05:23:01




```cpp
#include <bits/stdc++.h>
/*
 * Ejercicio 3 Shortest Routes II
 * Autor: Fabiola Tapara Quispe
 * Descripcion:
 *  Hay n ciudades y m carreteras entre ellas. Se nos pide procesar
 *  consultas "q" en las que tienes que determinar la longitud de la ruta
 *  más corta entre dos ciudades determinadas.
 * Solucion:    Para la solucion usaremos el algoritmo de Dikjstra,
 *              se considerara como un grafo no dirigido
 * Fecha: 13/12/21
 */
#define int long long
#define MOD 1000000007
#define MAX 1e14 // valor infinito

using namespace std;

void dijkstra() {
    int n, m, q; //n ciudades, m carreteras y q consultas
    cin >> n >> m >> q;// Recibimos o leemos los valores para 'n', 'm' y
    int x, y, z;//variables auxiliares, a partir de ellos crea el arreglo
    int graph[n][n];//creamos un arreglo bidimensional de [n][n]
    for(int i = 0; i < n; i++) {
        for(int j = 0; j < n; j++) {
            graph[i][j] = MAX;//valor infinito como inicio
        }
        graph[i][i] = 0;//la distancia de la Ciudad i hacia la Ciudad i s
    }

    for(int i = 0; i < m; i++) {
        cin >> x >> y >> z;
        x--; y--;
        graph[x][y] = min(z, graph[x][y]);
        graph[y][x] = min(z, graph[y][x]);
    }

    for(int k = 0; k < n; k++) {//Iteraremos sobre las posiciones del arr
        for(int i = 0; i < n; i++) {
            for(int j = 0; j < n; j++) {
                graph[i][j] = min(graph[i][j], graph[i][k] + graph[k][j])
            }
        }
    }


    for(int i = 0; i < q; i++) {//recibimos la cantidad de consultas que
        cin >> x >> y;// Leemos o recibimos sus valores del input ingresa
```

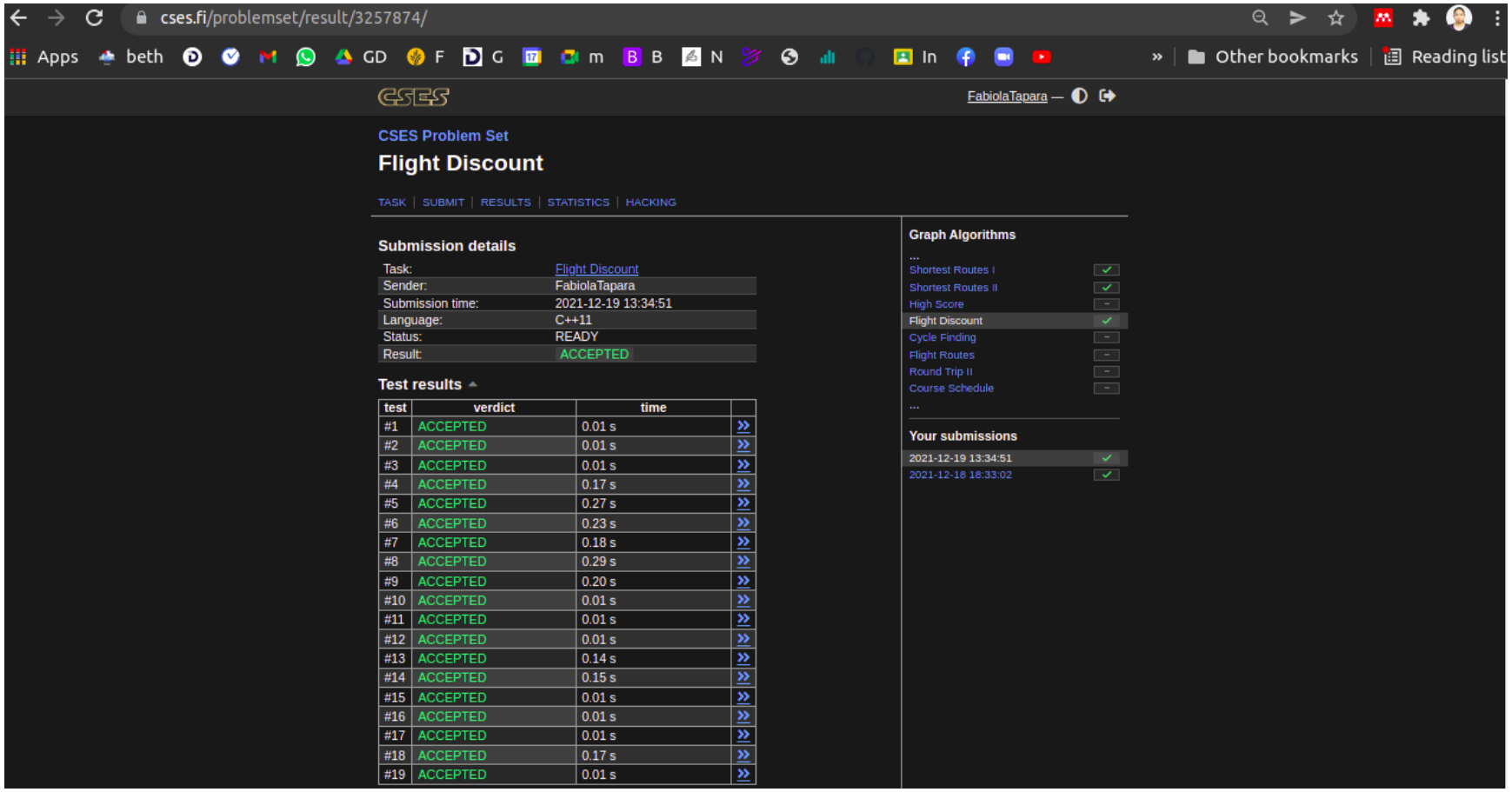
Ejercicio 4

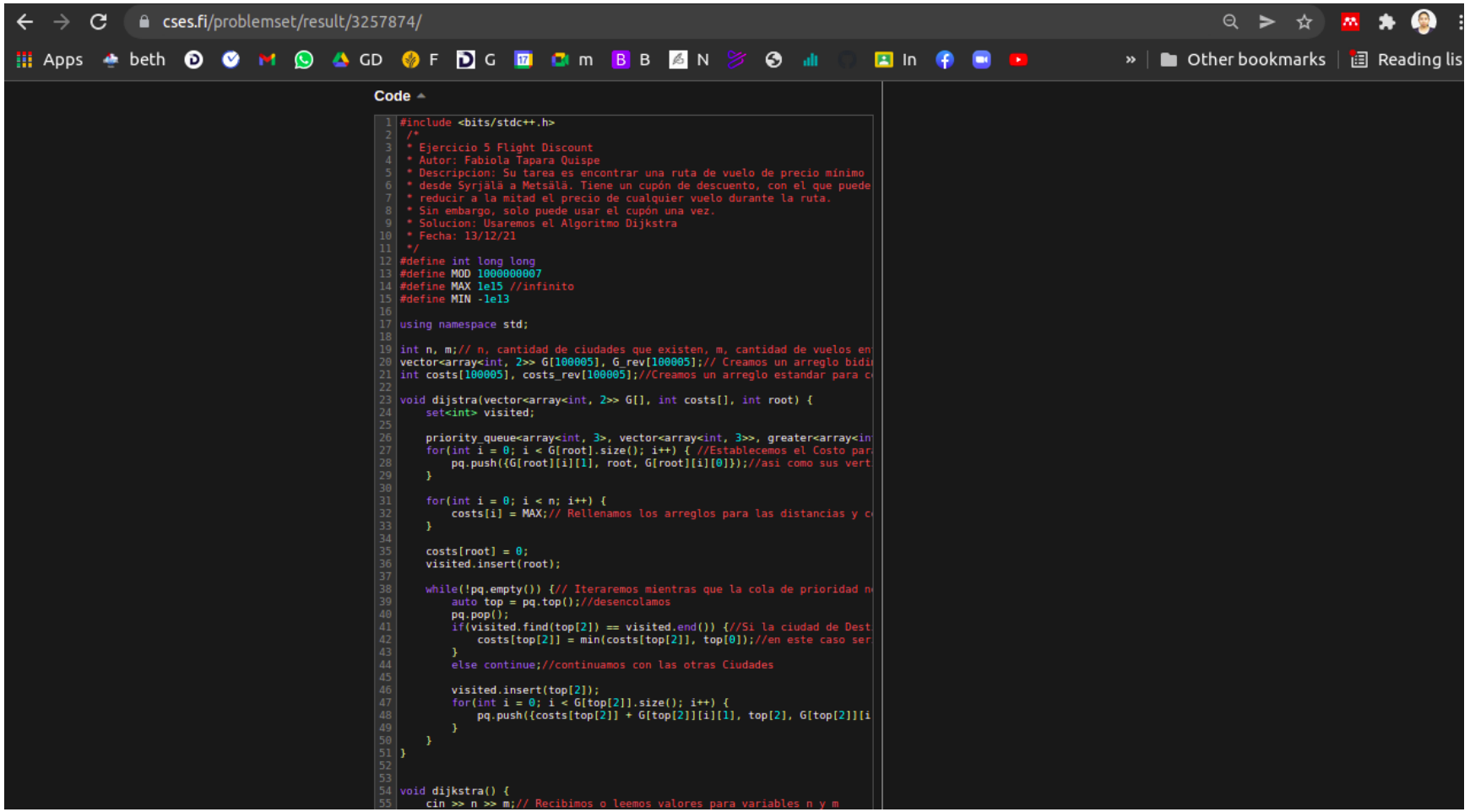

cses.fi/problemset/result/3257387/
Apps
beth
GD
F
G
m
B
N
In
Other bookmarks
Reading list
CSES
FabiolaTapara
CSES Problem Set
Longest Flight Route
TASK | SUBMIT | RESULTS | STATISTICS | HACKING
Submission details
Task: Longest Flight Route
Sender: FabiolaTapara
Submission time: 2021-12-19 11:11:07
Language: C++11
Status: READY
Result: ACCEPTED
Test results
test | verdict | time
#1 ACCEPTED 0.01 s
#2 ACCEPTED 0.01 s
#3 ACCEPTED 0.01 s
#4 ACCEPTED 0.01 s
#5 ACCEPTED 0.01 s
#6 ACCEPTED 0.08 s
#7 ACCEPTED 0.08 s
#8 ACCEPTED 0.09 s
#9 ACCEPTED 0.08 s
#10 ACCEPTED 0.08 s
#11 ACCEPTED 0.07 s
#12 ACCEPTED 0.08 s
#13 ACCEPTED 0.01 s
#14 ACCEPTED 0.01 s
#15 ACCEPTED 0.07 s
#16 ACCEPTED 0.01 s
#17 ACCEPTED 0.07 s
#18 ACCEPTED 0.05 s
#19 ACCEPTED 0.01 s
#20 ACCEPTED 0.05 s
Graph Algorithms
...
Flight Routes
Round Trip II
Course Schedule
Longest Flight Route
Game Routes
Investigation
Planets Queries I
Planets Queries II
...
Your submissions
2021-12-19 11:11:07
2021-12-19 11:10:34




```cpp
#include <bits/stdc++.h>
/*
 * Ejercicio 4 Longest Flight Route
 * Autor: Fabiola Tapara Quispe
 * Descripcion: Uolevi ha ganado un concurso, y el premio es un vuelo gra
 * que puede consistir en uno o más vuelos por ciudades. Uolevi
 * quiere volar de Syrjälä a Lehmälä para visitar el número máximo de ciu
 * Se le da la lista de vuelos posibles y sabe que no hay ciclos dirigido
 * la red de vuelos.
 * Solucion: Se usara el algoritmo DFS porque no pide encontrar el camino
 * largo desde una Ciudad o Vertice dentro de un Grafo
 * Fecha: 13/12/21
 */
#define int long long
#define MOD 1000000007
#define MAX 1e13

using namespace std;

vector<int> G[100005];
int n, m;//n ciudades y m vuelos
bool visitado[100005];//arreglo booleano, verifica si un vertice fue visi
int par[100005] = {-1};

void dfs(int root, vector<int>& path) {//camino mas largo dentro del Graf
    visitado[root] = true;// Revisamos que el hijo no haya sido visitado
    for(int i = 0; i < G[root].size(); i++) {//Para los Vertices que nace
        if(!visitado[G[root][i]]) {//iteraremos mientras que sean diferen
            dfs(G[root][i], path);
        }
    }
    path.push_back(root);//Vertice no ha sido visitado
}

void print_parent(int root) {
    if(root == -1) {
        return;
    }
    print_parent(par[root]);
    cout << root + 1 << " ";
}

void dijkstra() {
    cin >> n >> m;// Leemos o recibimos los valores para 'n' y 'm'
    for(int i = 0; i < m; i++) {
        int x, y; // Creamos variables auxiliar "x" y "y"
        cin >> x >> y; // Leemos o recibimos los valores para 'x' y 'y'
        x--; y--;
        G[x].push_back(y);// Agregamos las Arista desde el Origen hasta e
    }
    memset(visitado, false, sizeof(visitado));


    vector<int> nodes;
```

Ejercicio 5


cses.fi/problemset/result/3257874/
Apps
beth
GD
F
G
m
B
N
In
Other bookmarks
Reading list
CSES
FabiolaTapara
CSES Problem Set
Flight Discount
TASK | SUBMIT | RESULTS | STATISTICS | HACKING
Submission details
Task: Flight Discount
Sender: FabiolaTapara
Submission time: 2021-12-19 13:34:51
Language: C++11
Status: READY
Result: ACCEPTED
Test results
test | verdict | time
#1 ACCEPTED 0.01 s
#2 ACCEPTED 0.01 s
#3 ACCEPTED 0.01 s
#4 ACCEPTED 0.17 s
#5 ACCEPTED 0.27 s
#6 ACCEPTED 0.23 s
#7 ACCEPTED 0.18 s
#8 ACCEPTED 0.29 s
#9 ACCEPTED 0.20 s
#10 ACCEPTED 0.01 s
#11 ACCEPTED 0.01 s
#12 ACCEPTED 0.01 s
#13 ACCEPTED 0.14 s
#14 ACCEPTED 0.15 s
#15 ACCEPTED 0.01 s
#16 ACCEPTED 0.01 s
#17 ACCEPTED 0.01 s
#18 ACCEPTED 0.17 s
#19 ACCEPTED 0.01 s
Graph Algorithms
...
Shortest Routes I
Shortest Routes II
High Score
Flight Discount
Cycle Finding
Flight Routes
Round Trip II
Course Schedule
...
Your submissions
2021-12-19 13:34:51
2021-12-18 18:33:02




```cpp
#include <bits/stdc++.h>
/*
 * Ejercicio 5 Flight Discount
 * Autor: Fabiola Tapara Quispe
 * Descripcion: Su tarea es encontrar una ruta de vuelo de precio mínimo
 * desde Syrjälä a Metsälä. Tiene un cupón de descuento, con el que puede
 * reducir a la mitad el precio de cualquier vuelo durante la ruta.
 * Sin embargo, solo puede usar el cupón una vez.
 * Solucion: Usaremos el Algoritmo Dijkstra
 * Fecha: 13/12/21
 */
#define int long long
#define MOD 1000000007
#define MAX 1e15 //infinito
#define MIN -1e13

using namespace std;

int n, m;// n, cantidad de ciudades que existen, m, cantidad de vuelos en
vector<array<int, 2>> G[100005], G_rev[100005];// Creamos un arreglo bidi
int costs[100005], costs_rev[100005];//Creamos un arreglo estandar para c

void dijstra(vector<array<int, 2>> G[], int costs[], int root) {
    set<int> visited;

    priority_queue<array<int, 3>, vector<array<int, 3>>, greater<array<in
    for(int i = 0; i < G[root].size(); i++) { //Establecemos el Costo par
        pq.push({G[root][i][1], root, G[root][i][0]});//asi como sus vert
    }

    for(int i = 0; i < n; i++) {
        costs[i] = MAX;// Rellenamos los arreglos para las distancias y c
    }

    costs[root] = 0;
    visited.insert(root);

    while(!pq.empty()) {// Iteraremos mientras que la cola de prioridad n
        auto top = pq.top();//desencolamos
        pq.pop();
        if(visited.find(top[2]) == visited.end()) {//Si la ciudad de Dest
            costs[top[2]] = min(costs[top[2]], top[0]);//en este caso ser
        }
        else continue;//continuamos con las otras Ciudades

        visited.insert(top[2]);
        for(int i = 0; i < G[top[2]].size(); i++) {
            pq.push({costs[top[2]] + G[top[2]][i][1], top[2], G[top[2]][i
        }
    }
}


void dijkstra() {
    cin >> n >> m;// Recibimos o leemos valores para variables n y m
```