Name: ANYA AGARWAL

Section: B (13)

Roll no: 2021122

I. Assume that a project of road construction to connect some cities is given to your friend. Map of these cities and roads which will connect them (after construction) is provided to him in the form of a graph. Certain amount of rupees is associated with construction of each road. Your friend has to calculate the minimum budget required for this project. The budget should be designed in such a way that the cost of connecting the cities should be minimum and number of roads required to connect all the cities should be minimum (if there are N cities then only N-1 roads need to be constructed). He asks you for help. Now, you have to help your friend by designing an algorithm which will find minimum cost required to connect these cities. (use Prim's algorithm)

\*/

#include <iostream>

#include <vector>

#include <climits>

using namespace std;

int prim(const vector<vector<int>>& graph) {

int total\_cost = 0;

int n = graph.size();

vector<bool> visited(n, false);

vector<int> min\_cost(n, INT\_MAX);

min\_cost[0] = 0;

for (int count = 0; count < n - 1; ++count) {

int min\_vertex = -1;

for (int v = 0; v < n; ++v) {

if (!visited[v] && (min\_vertex == -1 || min\_cost[v] < min\_cost[min\_vertex])) {

min\_vertex = v;

}

}

visited[min\_vertex] = true;

for (int v = 0; v < n; ++v) {

if (graph[min\_vertex][v] && !visited[v] && graph[min\_vertex][v] < min\_cost[v]) {

min\_cost[v] = graph[min\_vertex][v];

}

}

}

for (int v = 1; v < n; ++v) {

total\_cost += min\_cost[v];

}

return total\_cost;

}

int main() {

int n;

cin >> n;

vector<vector<int>> graph(n, vector<int>(n));

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

for (int j = 0; j < n; ++j) {

cin >> graph[i][j];

}

}

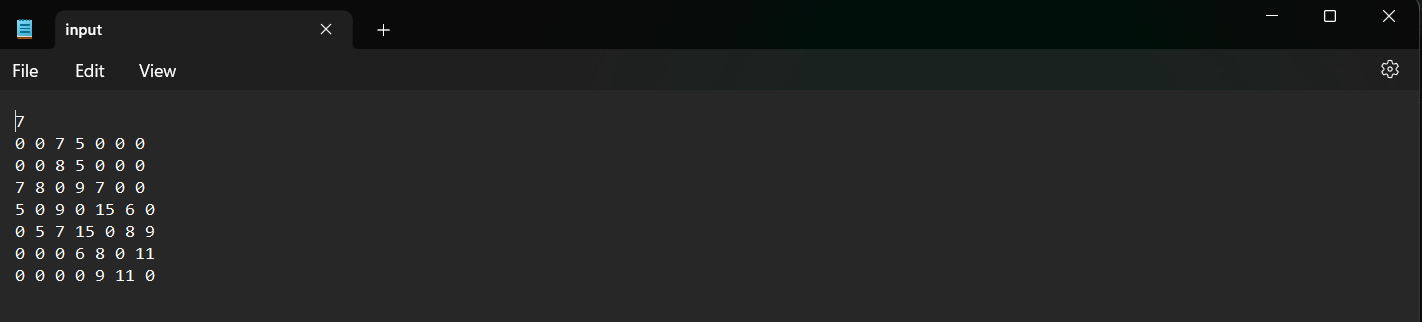
int minimum\_cost = prim(graph);

cout << minimum\_cost << endl;

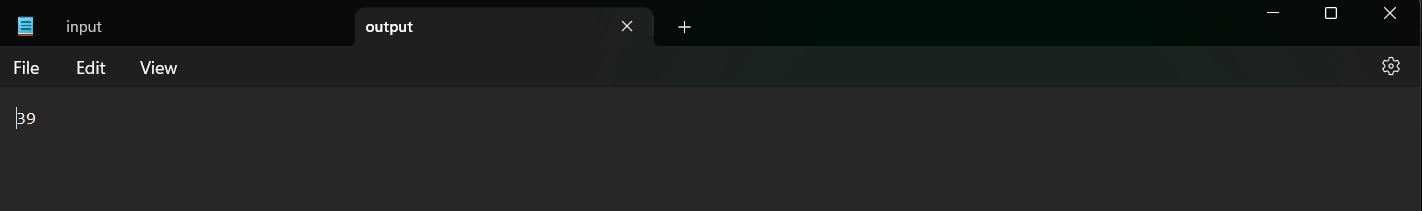
return 0;

}

Input:



Output:



Name: ANYA AGARWAL

Section: B (13)

Roll no: 2021122

II. Implement the previous problem using Kruskal's algorithm.

\*/

#include <iostream>

#include <fstream>

#include <vector>

#include <algorithm>

using namespace std;

struct Edge {

int src, dest, weight;

};

struct DisjointSet {

vector<int> parent, rank;

DisjointSet(int n) {

parent.resize(n);

rank.resize(n);

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

parent[i] = i;

rank[i] = 0;

}

}

int find(int u) {

if (parent[u] != u) {

parent[u] = find(parent[u]);

}

return parent[u];

}

void merge(int u, int v) {

int u\_root = find(u);

int v\_root = find(v);

if (rank[u\_root] < rank[v\_root]) {

parent[u\_root] = v\_root;

} else if (rank[u\_root] > rank[v\_root]) {

parent[v\_root] = u\_root;

} else {

parent[v\_root] = u\_root;

rank[u\_root]++;

}

}

};

int kruskal(vector<vector<int>>& graph) {

int n = graph.size();

// Store edges in a vector

vector<Edge> edges;

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

for (int j = i + 1; j < n; ++j) {

if (graph[i][j] != 0) {

edges.push\_back({i, j, graph[i][j]});

}

}

}

// Sort edges by weight

sort(edges.begin(), edges.end(), [](const Edge& e1, const Edge& e2) {

return e1.weight < e2.weight;

});

DisjointSet ds(n);

int min\_spanning\_weight = 0;

for (const auto& edge : edges) {

int src\_root = ds.find(edge.src);

int dest\_root = ds.find(edge.dest);

// Check if adding this edge creates a cycle

if (src\_root != dest\_root) {

min\_spanning\_weight += edge.weight;

ds.merge(src\_root, dest\_root);

}

}

return min\_spanning\_weight;

}

int main() {

ifstream infile("input.txt");

ofstream outfile("output.txt");

int n;

infile >> n;

vector<vector<int>> graph(n, vector<int>(n));

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

for (int j = 0; j < n; ++j) {

infile >> graph[i][j];

}

}

int minimum\_cost = kruskal(graph);

outfile << minimum\_cost << endl;

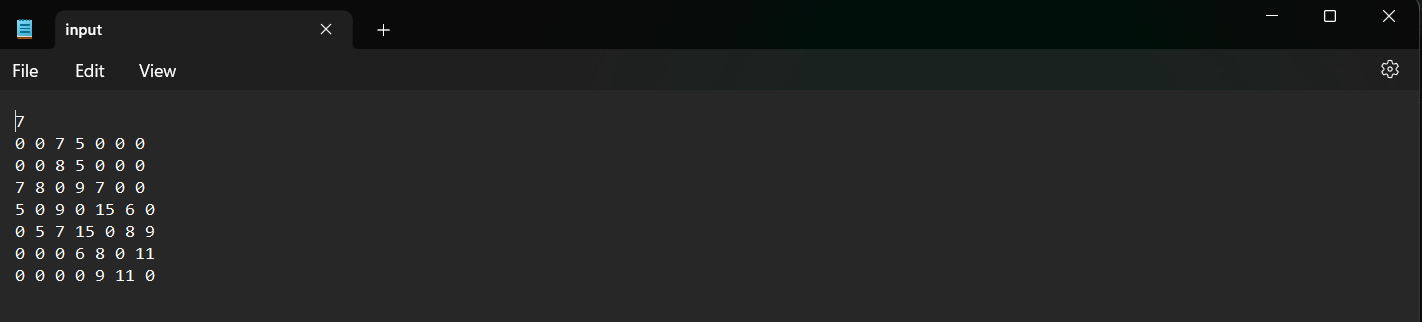
infile.close();

outfile.close();

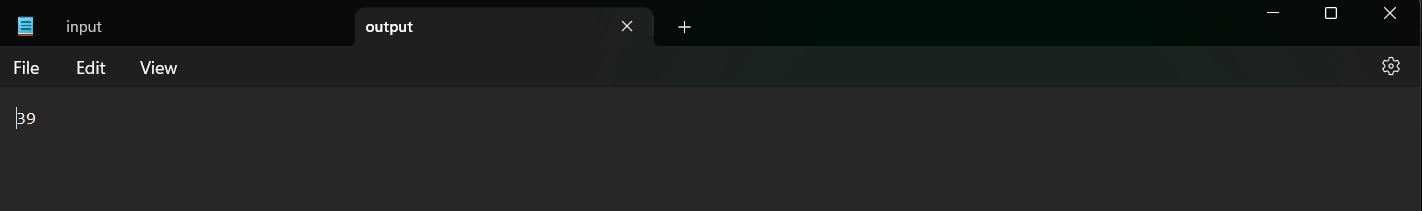
return 0;

}

Input:



Output:



Name: ANYA AGARWAL

Section: B (13)

Roll no: 2021122

/\*

III. Assume that same road construction project is given to another person. The amount he will earn from this project is directly proportional to the budget of the project. This person is greedy, so he decided to maximize the budget by constructing those roads who have highest construction cost. Design an algorithm and implement it using a program to find the maximum budget required for the project.

\*/

#include <iostream>

#include <fstream>

#include <vector>

using namespace std;

int maximumBudget(const vector<vector<int>>& graph) {

int n = graph.size();

int budget = 0;

// Iterate over the upper triangle of the adjacency matrix

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

for (int j = i; j < n; ++j) {

budget += graph[i][j]; // Add construction cost to budget

}

}

return budget;

}

int main() {

ifstream infile("input.txt");

ofstream outfile("output.txt");

int n;

infile >> n; // Number of vertices

// Input adjacency matrix

vector<vector<int>> graph(n, vector<int>(n));

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

for (int j = 0; j < n; ++j) {

infile >> graph[i][j];

}

}

int max\_budget = maximumBudget(graph);

outfile << max\_budget << endl;

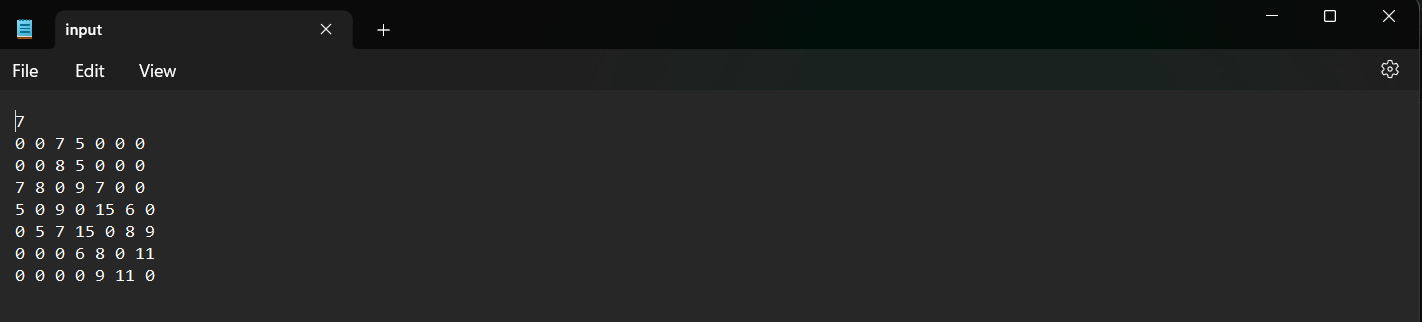
infile.close();

outfile.close();

return 0;

}

Input:



Output:

