**DATA STRUCTURE**

**LAB 15**

**Name :** Faizan Tariq

**Roll NO:** 22F-3858

**TASK 1 (PRIMS):**

#include <iostream>

using namespace std;

struct Node {

int data;

Node\* next;

Node() {

data = 0;

next = NULL;

}

Node(int data) {

this->data = data;

next = NULL;

}

};

class StackAdt {

Node\* top;

public:

StackAdt() {

top = NULL;

}

bool isEmpty() {

return top == NULL;

}

bool push(int data) {

Node\* newNode = new Node(data);

newNode->next = top;

top = newNode;

return true;

}

bool pop(int& tempData) {

if (isEmpty()) {

return false;

}

Node\* tempNode = top;

tempData = tempNode->data;

top = tempNode->next;

delete tempNode;

return true;

}

void displayStack() {

Node\* cur = top;

cout << endl;

while (cur != NULL) {

cout << cur->data << " ";

cur = cur->next;

}

cout << endl;

}

~StackAdt() {

int n;

while (top != NULL) {

pop(n);

}

delete top;

}

};

struct Edge {

int des;

int weight;

};

class Graph {

int V; // number of vertices

Edge\*\* adj;

public:

Graph(int V) : V(V) {

adj = new Edge \* [V];

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

adj[i] = new Edge[V];

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

adj[i][j].des = -1;

adj[i][j].weight = -1;

}

}

}

void addEdge(int src, int des, int weight) {

adj[src][des].des = des;

adj[src][des].weight = weight;

adj[des][src].des = src;

adj[des][src].weight = weight;

}

void prim() {

int\* dist;

dist = new int[V];

bool\* isVisited = new bool[V];

int\* parent = new int[V];

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

dist[i] = INT\_MAX;

isVisited[i] = false;

parent[i] = -1;

}

dist[0] = 0;

StackAdt s; //this stack store MST vertices

s.push(0); // startfrom first vertex in graph

while (!s.isEmpty()) {

int u;

s.pop(u);

isVisited[u] = true;

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

if (!isVisited[v] && adj[u][v].weight != -1 && adj[u][v].weight < dist[v]) {

dist[v] = adj[u][v].weight;

parent[v] = u;

s.push(v);

}

}

}

cout << "\nResults:"<<endl;

for (int i = 1; i < V; ++i) {

cout << "Edge " << parent[i] << " - " << i << " : " << dist[i] << '\n';

}

}

};

int main() {

Graph g(5);

g.addEdge(0, 1, 2);// source - destination - weight

g.addEdge(0, 3, 6);

g.addEdge(1, 2, 3);

g.addEdge(1, 3, 8);

g.addEdge(1, 4, 5);

g.addEdge(2, 4, 7);

g.addEdge(3, 4, 9);

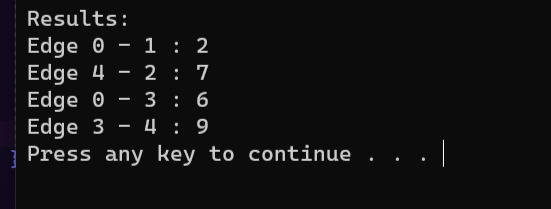
g.prim();

system("pause");

return 0;

}

OUTPUT:



**TASK 2 (Dijkstra):**

#include <iostream>

using namespace std;

struct Edge

{

int des;

int weight;

};

class Graph

{

int V;

Edge\*\* adj;

public:

Graph(int V) : V(V)

{

adj = new Edge \* [V];

for (int i = 0; i < V; ++i)

{

adj[i] = new Edge[V];

for (int j = 0; j < V; ++j)

{

adj[i][j].des = -1;

adj[i][j].weight = -1;

}

}

}

void addEdge(int src, int des, int weight)

{

adj[src][des].des = des;

adj[src][des].weight = weight;

adj[des][src].des = src;

adj[des][src].weight = weight;

}

void dijkstra(int src)

{

int\* dist;

dist = new int[V];

bool\* visited = new bool[V];

for (int i = 0; i < V; ++i)

{

dist[i] = INT\_MAX;

visited[i] = false;

}

dist[src] = 0;

for (int i = 0; i < V - 1; ++i)

{

int u = -1;

for (int j = 0; j < V; ++j)

{

if (!visited[j] && (u == -1 || dist[j] < dist[u]))

{

u = j;

}

}

visited[u] = true;

for (int v = 0; v < V; ++v)

{

if (!visited[v] && adj[u][v].weight != -1 && dist[u] + adj[u][v].weight < dist[v])

{

dist[v] = dist[u] + adj[u][v].weight;

}

}

}

cout << "Results:\n";

for (int i = 0; i < V; ++i)

{

cout << "Vertex " << i << ": " << dist[i] << '\n';

}

}

};

int main()

{

Graph g(4);

g.addEdge(0, 1, 2);

g.addEdge(0, 3, 6);

g.addEdge(2, 1, 12);

g.addEdge(3, 2, 9);

int start\_Ver = 0;

cout << "ENter Starting vertix :";

cin>>start\_Ver;

g.dijkstra(start\_Ver);

system("pause");

return 0;

}

**Output:**

**A screenshot of a computer

Description automatically generated**