# MET Institute of Computer Science

|  |  |
| --- | --- |
| Program No | 22 |
| Roll No | 1333 |
| Unit | 7 |
| Program | Graph storage structures - Adjency Matrix |

**Source Code:**

#include<iostream>

#include<conio.h> #define MAX 10

using namespace std;

/\* Node Template-NOT REQUIRED \*/

/\* Graph Template \*/ class Graph

{

int adj[MAX][MAX]; int n;

int e; public: Graph()

{

int i, j;

for(i = 0; i < MAX; i++)

{

for(j = 0; j < MAX; j++)

{

adj[i][j] = 0;

}

}

n = 0; e = 0;

}

void CreateGraph(); void Display();

# MET Institute of Computer Science

};

/\* Function \*/

//creategraph

void Graph :: CreateGraph()

{

cout << "Enter the number of nodes: "; cin >> n; //n is declared in the graph class cout << "Enter the number of edges: "; cin >> e; //n is declared in the graph class int i, source, dest;

for(i = 1; i<=e; i++)

{

cout << "Input Edge " << i << endl; cout << "Enter source node: ";

cin >> source;

cout << "Enter destination node: "; cin >> dest;

adj[source][dest]=1; adj[dest][source]=1;

}

}

//display

void Graph :: Display()

{

int i, j;

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

{

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

{

cout << adj[i][j] << " ";

}

cout << endl;

}

}

# MET Institute of Computer Science

/\* Menu \*/ int main()

{

int num, ch; Graph g; while(1)

{

system("cls");

cout << "\*\*\* Graph \*\*\*\n\n";

cout << "1. Create a graph" << endl; cout << "2. Display" << endl;

cout << "3. Exit" << endl;

cout << "\nEnter your choice: "; cin >> ch;

switch(ch)

{

case 1: g.CreateGraph(); getch();

break; case 2: g.Display();

getch(); break; case 3:

exit(1); default:

cout << "Incorrect option!"; getch();

break;

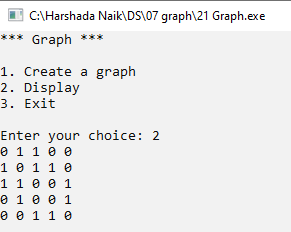
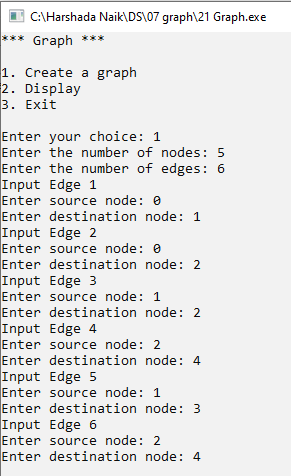
}

}

}

**Output:**

# MET Institute of Computer Science



**MET Institute of Computer Science**

|  |  |
| --- | --- |
| Program No | 23 |
| Roll No | 1333 |
| Unit | 7 |
| Program | **Depth first Traversal of a graph** |

**Source Code:**

#include<iostream>

#include<conio.h> #define MAX 10

using namespace std;

/\* stack Template \*/ class Stack

{

int a[MAX]; int tos; public:

Stack()

{

tos = -1;

}

void Push(int x); int Pop();

int Empty();

};

//Push

void Stack :: Push(int x)

{

tos++; a[tos] = x;

}

int Stack :: Pop()

{

# MET Institute of Computer Science

int x = a[tos]; tos--;

return x;

}

int Stack :: Empty()

{

if(tos == -1)

{

return 1;

}

else

{

return 0;

}

}

/\* Node Template-NOT REQUIRED \*/

/\* Graph Template \*/ class Graph

{

int adj[MAX][MAX]; int n;

int e;

int visited[MAX]; public:

Graph()

{

int i, j;

for(i = 0; i < MAX; i++)

{

for(j = 0; j < MAX; j++)

{

adj[i][j] = 0;

}

# MET Institute of Computer Science

n = 0; e = 0;

}

void CreateGraph(); void Display();

void DFT(int x);

};

/\* Function \*/

void Graph :: CreateGraph()

{

cout << "Enter the number of nodes: ";

cin >> n; //n is declared in the graph class cout << "Enter the number of edges: ";

cin >> e; //n is declared in the graph class int i, source, dest;

for(i = 1; i<=e; i++)

{

cout << "Input Edge " << i << endl; cout << "Enter source node: ";

cin >> source;

cout << "Enter destination node: "; cin >> dest;

adj[source][dest]=1; adj[dest][source]=1;

}

}

//display

void Graph :: Display()

{

int i, j;

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

{

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

{

cout << adj[i][j] << " ";

# MET Institute of Computer Science

cout << endl;

}

}

//DFT

void Graph :: DFT(int x)

{

Stack s; int i;

//update visited to 0 for(i=0; i<n; i++)

{

visited[i] = 0;

}

s.Push(x); visited[x] = 1; while(!s.Empty())

{

x = s.Pop(); cout << x << " ";

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

{

if(adj[x][i]==1 && visited[i]==0)

{

s.Push(i); visited[i] = 1;

}

}

}

}

/\* Menu \*/ int main()

{

int num, ch; Graph g; while(1)

{

# MET Institute of Computer Science

system("cls");

cout << "\*\*\* DFT of Graph \*\*\*\n\n"; cout << "1. Create a graph" << endl; cout << "2. Display" << endl;

cout << "3. DFT" << endl; cout << "4. exit" << endl;

cout << "\nEnter your choice: "; cin >> ch;

switch(ch)

{

case 1: g.CreateGraph(); getch();

break; case 2: g.Display();

getch(); break; case 3:

cout << "Enter the node: "; cin >> num;

g.DFT(num);

getch(); break; case 4:

exit(1); default:

cout << "Incorrect option!"; getch();

break;

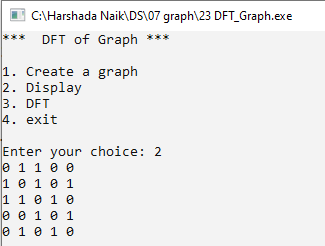
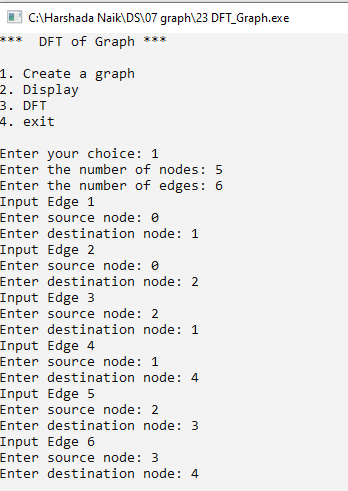
}

}

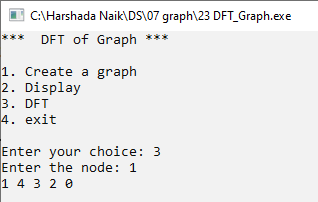
}

**Output:**

# MET Institute of Computer Science



**MET Institute of Computer Science**



# MET Institute of Computer Science

|  |  |
| --- | --- |
| Program No | 24 |
| Roll No | 1333 |
| Unit | 7 |
| Program | **Breadth first Traversal of a graph** |

**Source Code: #include<iostream>**

#include<conio.h> #define MAX 10

using namespace std;

/\* Queue Template \*/ class Queue

{

int a[MAX]; int front, rear; public:

Queue()

{

front = rear = -1;

}

void Enqueue(int x); int Dequeue();

int Empty();

};

void Queue::Enqueue(int x)

{

if(front == -1)

{

front++;

}

rear++; a[rear] = x;

# MET Institute of Computer Science

}

int Queue::Dequeue()

{

if(front == -1)

{

cout << "Underflow"; return -1;

}

int x = a[front]; if(front == rear)

{

front = rear = -1;

}

else

{

front++;

}

return x;

}

int Queue::Empty()

{

if(front == -1)

{

return 1;

}

else

{

return 0;

}

}

/\* 1. Node Template-Not Required \*/

# MET Institute of Computer Science

/\* 2. Graph Template \*/ class Graph

{

int adj[MAX][MAX]; int n;

int e;

int visited[MAX]; public:

Graph()

{

int i,j; for(i=0;i<MAX;i++)

{

for(j=0;j<MAX;j++)

{

adj[i][j] = 0;

}

}

n = 0;

e = 0;

}

void CreateGraph(); void Display();

void BST(int x);

};

/\* 3. Function \*/

void Graph :: CreateGraph()

{

cout << "Enter the number of nodes: ";

cin >> n; // n is declared in the Graph Class cout << "Enter the number of edges: ";

cin >> e; // e is declared in the Graph Class int i,source,dest;

for(i=1;i<=e;i++)

{

cout << "Input Edge " << i << endl;

# MET Institute of Computer Science

cout << "Enter source node: "; cin >> source;

cout << "Enter destination node: "; cin >> dest;

adj[source][dest] = 1;

adj[dest][source] = 1;

}

}

void Graph :: Display()

{

int i,j; for(i=0;i<n;i++)

{

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

{

cout << adj[i][j] << " ";

}

cout << endl;

}

}

void Graph::BST(int x)

{

Queue q; int i;

//Update visited to 0 for(i=0;i<n;i++)

{

visited[i]=0;

}

q.Enqueue(x); visited[x] = 1; while(!q.Empty())

{

x = q.Dequeue(); cout << x << " ";

# MET Institute of Computer Science

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

{

if(adj[x][i] == 1 && visited[i] == 0)

{

q.Enqueue(i); visited[i] = 1;

}

}

}

}

/\* 4. Main \*/ int main()

{

int ch, num; Graph b; while(1)

{

system("cls");

cout << "\*\*\* Graphs \*\*\*\n\n";

cout << "1. Create a Graph"<< endl; cout << "2. Display a Graph" << endl;

cout << "3. Breadth First Traversal" << endl; cout << "4. Exit" << endl;

cout << "Enter your choice: "; cin >> ch;

switch(ch)

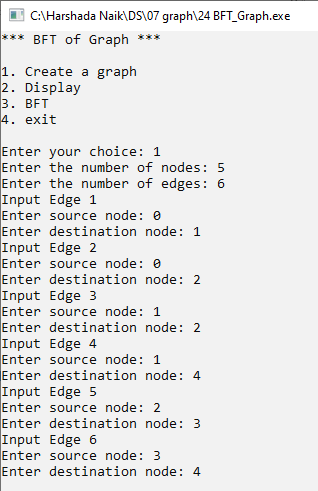
{

case 1: b.CreateGraph(); getch();

break; case 2: b.Display();

getch(); break; case 3:

# MET Institute of Computer Science



cout << "Enter the node: "; cin >> num;

b.BST(num);

getch(); break; case 4:

exit(1); default:

cout << "Incorect Choice!" << endl; getch();

break;

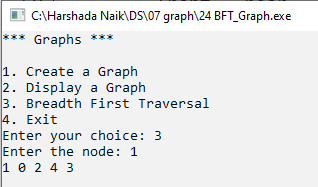
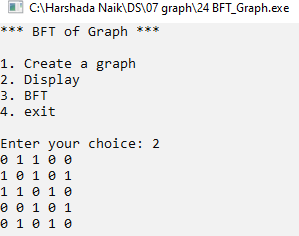
}

}

}

**Output:**

# MET Institute of Computer Science



**MET Institute of Computer Science**

|  |  |
| --- | --- |
| Program No | 25 |
| Roll No | 1333 |
| Unit | 7 |
| Program | **MST - Kruskal's Algorithm** |

**Source Code:**

#include<iostream> #include<conio.h>

#define MAX 10

#define INF 999

using namespace std;

/\* Node Template-NOT REQUIRED \*/

/\* Graph Template \*/

class Graph

{

int adj[MAX][MAX]; int n;

int e;

int parent[MAX]; public:

Graph()

{

int i, j;

for(i = 0; i < MAX; i++)

{

for(j = 0; j < MAX; j++)

{

adj[i][j] = INF;

}

# MET Institute of Computer Science

}

n = 0; e = 0;

for(i=0; i<MAX; i++)

{

parent[i] = -1;

}

}

void CreateGraph(); void Display();

void Kruskal(); int Find(int x);

int Union(int x, int y);

};

/\* Function \*/

//creategraph

void Graph :: CreateGraph()

{

cout << "Enter the number of nodes: ";

cin >> n; //n is declared in the graph class cout << "Enter the number of edges: ";

cin >> e; //n is declared in the graph class int i, source, dest, weight;

for(i = 1; i<=e; i++)

{

cout << "Input Edge " << i << endl; cout << "Enter source node: ";

cin >> source;

cout << "Enter destination node: "; cin >> dest;

cout << "Enter the weight: "; cin >> weight; adj[source][dest] = weight;

# MET Institute of Computer Science

adj[dest][source] = weight;

}

}

//display

void Graph :: Display()

{

int i, j;

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

{

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

{

if(adj[i][j] == INF)

{

cout << "0 ";

}

else

{

cout << adj[i][j] << " ";

}

}

cout << endl;

}

}

void Graph :: Kruskal()

{

int minimum, i, j, ne = 1; int a, b, u, v;

int mincost = 0; while(ne < n)

{

for(i = 0, minimum = 999; i<n; i++)

{

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

{

# MET Institute of Computer Science

if(adj[i][j] < minimum)

{

minimum = adj[i][j]; a = u = i;

b = v = j;

}//end of if

}//end of j

}//end of for i u = Find(u); v = Find(v); if(Union(u,v))

{

cout << "Edge " << ne++ << " (" << a << "," << b << ") = " << minimum << endl; mincost += minimum;

}

adj[a][b] = adj[b][a] = INF;

}//end of while loop

cout << "MST Minimum Cost = " << mincost;

}

int Graph :: Find(int x)

{

while(parent[x] != -1)

{

x = parent[x];

}

return x;

}

int Graph :: Union(int x, int y)

{

if(x!=y)

{

parent[y] = x; return 1;

}

# MET Institute of Computer Science

}

/\* Menu \*/ int main()

{

int num, ch; Graph g; while(1)

{

system("cls");

cout << "\*\*\* MST using Kruskal's Algorithm \*\*\*\n\n"; cout << "1. Create a graph" << endl;

cout << "2. Display" << endl;

cout << "3. MST using Kruskal's Algorithm" << endl; cout << "4. Exit" << endl;

cout << "\nEnter your choice: "; cin >> ch;

switch(ch)

{

case 1: g.CreateGraph(); getch();

break; case 2: g.Display();

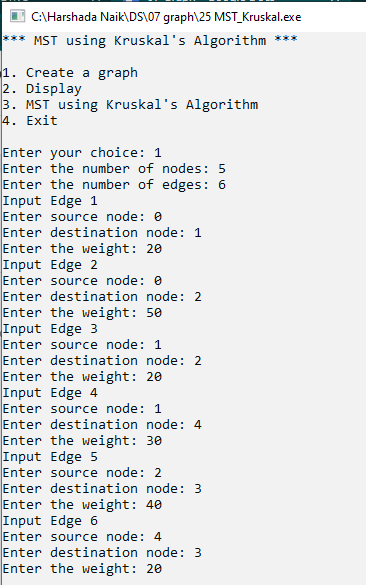
getch(); break; case 3: g.Kruskal();

getch(); break; case 4:

exit(1); default:

cout << "Incorrect option!"; getch();

# MET Institute of Computer Science



break;

}

}

}

**Output:**

# MET Institute of Computer Science

