CODE(Kruskal’s):

#include <bits/stdc++.h>

using namespace std;

int Root[5];

int find(int i)

{

while (Root[i] != i){

i = Root[i];

}

return i;

}

void union1(int i, int j)

{

int a = find(i);

int b = find(j);

Root[a] = b;

}

void kruskalMST(int cost[][5])

{

int mincost = 0;

for (int i = 0; i < 5; i++)

Root[i] = i;

int edge\_count = 0;

while (edge\_count < 5 - 1) {

int min = INT\_MAX, a = -1, b = -1;

for (int i = 0; i < 5; i++) {

for (int j = 0; j < 5; j++) {

if (find(i) != find(j) && cost[i][j] < min) {

min = cost[i][j];

a = i;

b = j;

}

}

}

union1(a, b);

printf("Edge %d:(%d, %d) cost:%d \n",

edge\_count++, a, b, min);

mincost += min;

}

printf("\n Minimum cost= %d \n", mincost);

}

int main()

{

int cost[][5] = {

{ INT\_MAX, 2, INT\_MAX, 6, INT\_MAX },

{ 2, INT\_MAX, 3, 8, 5 },

{ INT\_MAX, 3, INT\_MAX, INT\_MAX, 7 },

{ 6, 8, INT\_MAX, INT\_MAX, 9 },

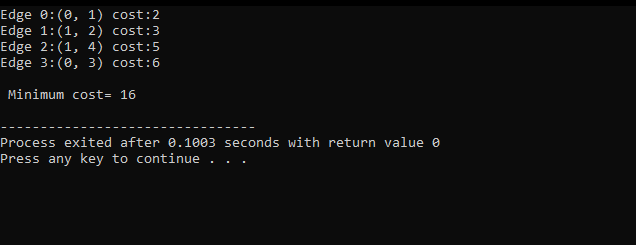
{ INT\_MAX, 5, 7, 9, INT\_MAX },

};

kruskalMST(cost);

return 0;

}



CODE(Prim’s):

#include <bits/stdc++.h>

using namespace std;

#define V 5

int minKey(int key[], bool mstSet[])

{

// Initialize min value

int min = INT\_MAX, min\_index;

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

if (mstSet[v] == false && key[v] < min)

min = key[v], min\_index = v;

return min\_index;

}

void printMST(int parent[], int graph[V][V])

{

cout<<"Edge \tWeight\n";

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

cout<<parent[i]<<" - "<<i<<" \t"<<graph[i][parent[i]]<<" \n";

}

void primMST(int graph[V][V])

{

int parent[V];

int key[V];

bool mstSet[V];

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

key[i] = INT\_MAX, mstSet[i] = false;

key[0] = 0;

parent[0] = -1;

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

{

int u = minKey(key, mstSet);

mstSet[u] = true;

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

if (graph[u][v] && mstSet[v] == false && graph[u][v] < key[v])

parent[v] = u, key[v] = graph[u][v];

}

printMST(parent, graph);

}

int main()

{

/\* Let us create the following graph

2 3

(0)--(1)--(2)

| / \ |

6| 8/ \5 |7

| / \ |

(3)-------(4)

9 \*/

int graph[V][V] = { { 0, 2, 0, 6, 0 },

{ 2, 0, 3, 8, 5 },

{ 0, 3, 0, 0, 7 },

{ 6, 8, 0, 0, 9 },

{ 0, 5, 7, 9, 0 } };

primMST(graph);

return 0;

}

Output:

