#include <bits/stdc++.h>

using namespace std;

#define V 5

int parent[V];

// Find set of vertex i

int find(int i)

{

while (parent[i] != i)

i = parent[i];

return i;

}

// Does union of i and j.

//It returns false if i and j are already in same set.

void union1(int i, int j)

{

int a = find(i);

int b = find(j);

parent[a] = b;

}

// Finds MST using Kruskal's algorithm

void kruskalMST(int cost[][V])

{

int mincost = 0; // Cost of min MST.

// Initialize sets of disjoint sets.

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

parent[i] = i;

// Include minimum weight edges one by one

int edge\_count = 0;

while (edge\_count < V - 1) {

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

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

for (int j = 0; j < V; 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("\nMinimum cost : %d \n", mincost);

}

// driver program to test above function

int main()

{

/\* Let us create the following graph

2 3

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

| / \ |

6| 8/ \5 |7

| / \ |

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

9 \*/

int cost[][V] = {

{ 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 },

};

// Print the solution

kruskalMST(cost);

return 0;

}