***Kruskal Minimum Spanning Tree***

import java.util.\*;

import java.lang.\*;

import java.io.\*;

class Main {

class Edge implements Comparable<Edge>

{

int src, dest, weight;

public int compareTo(Edge compareEdge)

{

return this.weight - compareEdge.weight;

}

};

class subset

{

int parent, rank;

};

int V, E;

Edge edge[];

Main(int v, int e)

{

V = v;

E = e;

edge = new Edge[E];

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

edge[i] = new Edge();

}

int find(subset subsets[], int i)

{

if (subsets[i].parent != i)

subsets[i].parent

= find(subsets, subsets[i].parent);

return subsets[i].parent;

}

void Union(subset subsets[], int x, int y)

{

int xroot = find(subsets, x);

int yroot = find(subsets, y);

if (subsets[xroot].rank

< subsets[yroot].rank)

subsets[xroot].parent = yroot;

else if (subsets[xroot].rank

> subsets[yroot].rank)

subsets[yroot].parent = xroot;

else {

subsets[yroot].parent = xroot;

subsets[xroot].rank++;

}

}

void KruskalMST()

{

Edge result[] = new Edge[V];

int e = 0;

int i = 0;

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

result[i] = new Edge();

Arrays.sort(edge);

subset subsets[] = new subset[V];

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

subsets[i] = new subset();

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

{

subsets[v].parent = v;

subsets[v].rank = 0;

}

i = 0;

while (e < V - 1)

{

Edge next\_edge = edge[i++];

int x = find(subsets, next\_edge.src);

int y = find(subsets, next\_edge.dest);

if (x != y) {

result[e++] = next\_edge;

Union(subsets, x, y);

}

}

System.out.println("Following are the edges in "

+ "the constructed MST");

int minimumCost = 0;

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

{

System.out.println(result[i].src + " -- "

+ result[i].dest

+ " == " + result[i].weight);

minimumCost += result[i].weight;

}

System.out.println("Minimum Cost Spanning Tree "

+ minimumCost);

}

public static void main(String[] args)

{

int V = 4;

int E = 5;

Main graph = new Main(V, E);

graph.edge[0].src = 0;

graph.edge[0].dest = 1;

graph.edge[0].weight = 10;

graph.edge[1].src = 0;

graph.edge[1].dest = 2;

graph.edge[1].weight = 6;

graph.edge[2].src = 0;

graph.edge[2].dest = 3;

graph.edge[2].weight = 5;

graph.edge[3].src = 1;

graph.edge[3].dest = 3;

graph.edge[3].weight = 15;

graph.edge[4].src = 2;

graph.edge[4].dest = 3;

graph.edge[4].weight = 4;

graph.KruskalMST();

}

}