public Iterable<Edge> adf(int v)

Return adj

Public class Edge implements Comparable<Edge>

{

public final int v, int w;

Private final double weight

Public Edge ( int v, int w, double wieght)

{

This.w = w;

This.v = v;

This.weight = weight;

}

Public int either()

Publicint other ( int vertex)

Pblic double weight()

Return wieght;

Private static class ByWieght+Comparator implements Comparator<Edge>

{

Public int compare (Edge e, Edge f)

{

If e.wieght < f.weight) return -1;

if (e.weight > f. weight ) return 1;

Return 0;

}

}

//this line below is inside the Edge Class.

Public final static Comparaator<edge> BY\_WEIGHT = new ByWeight Comparator()

Publi cint compareTo( edge that)

{

If this.weight < that.weight

Reutnr -1;

Else if theis.weight > thtat.weight

Return 1;

Else

Return 0;

}

main()

{

For (int v = 0; v< G.V(); v++)

{

` for(Edge e : G.adj(v))

{

Int w = e.other (v);

.

. .

.

}

}

}

Minimum Spanning Tree (MST)

Given an Undirected Graph G with positive edge weights. (Connected)

Goal: find a set of edges that connects all of the vertices in G. Such that the total weight of this set of edges is minimum.