模版·Kruskal(最小生成树)

```
class Edge implements Comparable<Edge>{
    int src;
    int dest;
    int weight;

    public Edge(int src,int dest,int weight){
        this.src=src;
        this.dest=dest;
        this.weight=weight;
    }

    @Override
    public int compareTo(Edgeother){
        return this.weight-other.weight;
    }
}

class Graph{
    int vertices;
    List<Edge> edges;

    public Graph(int vertices){
        this.vertices=vertices;
        this.edges=new ArrayList<>();
    }

    public void addEdge(int src,int dest,int weight){
        edges.add(new Edge(src,dest,weight));
    }
```

```
    public int findMST(){
        DisjointSet disjointSet=new DisjointSet(vertices);
        Collections.sort(edges);

        int mstWeight=0;
        for(Edge edge:edges){
            int src=edge.src;
            int dest=edge.dest;

            if(!Objects.equals(disjointSet.find(src),disjointSet.find(dest))){
                disjointSet.union(src,dest);
                mstWeight+=edge.weight;
            }
        }

        return mstWeight;
        }
    }

class DisjointSet{
    private int[] parent;
    private int[] rank;

    public DisjointSet(int n){
        parent=new int[n];
        rank=new int[n];

        for(int i=0;i<n;i++){
            parent[i]=i;
            rank[i]=0;
        }
    }
```

```
    public int find(int u){
        if(u!=parent[u]){
            parent[u]=find(parent[u]);
        }

        return parent[u];
    }

    public void union(int x,int y){
    int xRoot=find(x);
    int yRoot=find(y);

    if(rank[xRoot]<rank[yRoot]){
        parent[xRoot]=yRoot;
    }else if(rank[xRoot]>rank[yRoot]){
        parent[yRoot]=xRoot;
        }else{
            rank[xRoot]++;
            parent[yRoot]=xRoot;
        }
    }
}

public class KruskalMST{

    public static void main(String[] args){
        Scanner scanner=new Scanner(System.in);

        System.out.print("顶点总数:");
        int vertices=scanner.nextInt();

        Graph graph=new Graph(vertices);
```

```
            System.out.print("边总数:");
            int edgesCount=scanner.nextInt();

            for(int i=0;i<edgesCount;i++){
                System.out.print("出发点:");
                int src=scanner.nextInt();

                System.out.print("目的点:");
                int dest=scanner.nextInt();

                System.out.print("权重:");
                int weight=scanner.nextInt();

                graph.addEdge(src-1,dest-1,weight);
            }

            int mstWeight=graph.findMST();
            System.out.println("最小生成树边权重之和:"+mstWeight);
        }
}
```