Kruscal's Algorithm

Dr. Mattox Beckman

University of Illinois at Urbana-Champaign
Department of Computer Science

Objectives

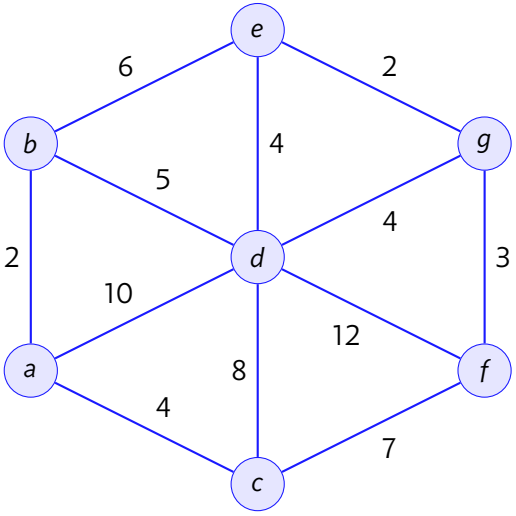
Your Objectives:

- Implement Kruscal's Algorithm

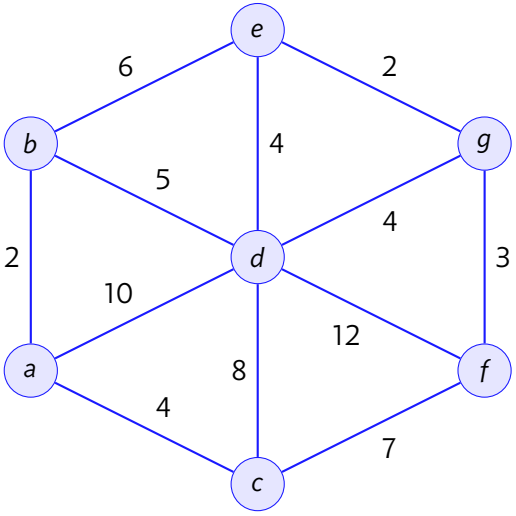
The Algorithm

- Insert all edges into a priority queue
- Initialize a disjoint set with all the edges
- While there are fewer than $|V| - 1$ edges in your MST:
  - Dequeue an edge.
  - If the incident vertices are not both part of the MST already, add the edge. (Use the disjoint set to keep track)

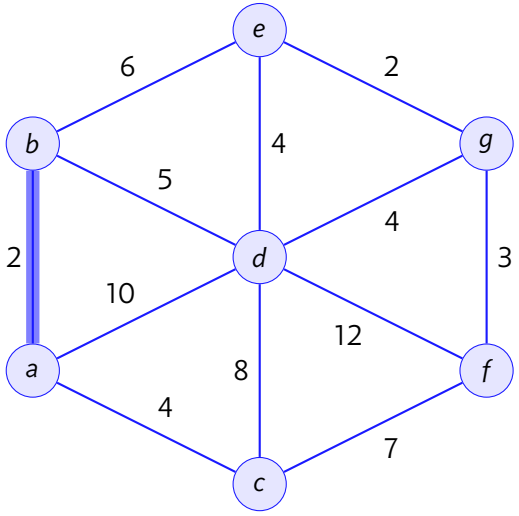
Example



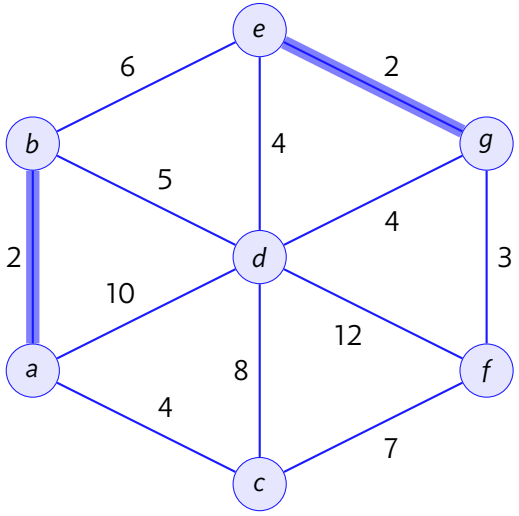
Example



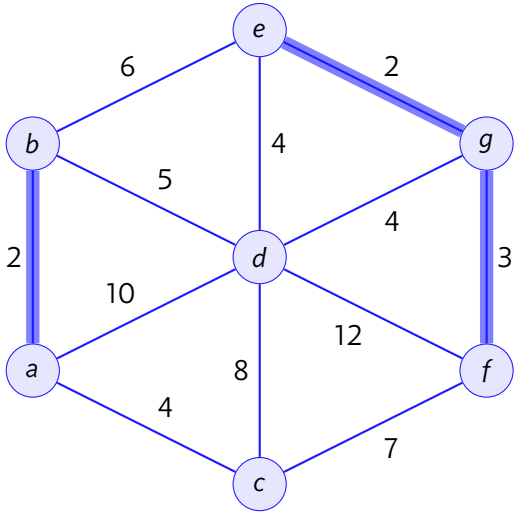
Example



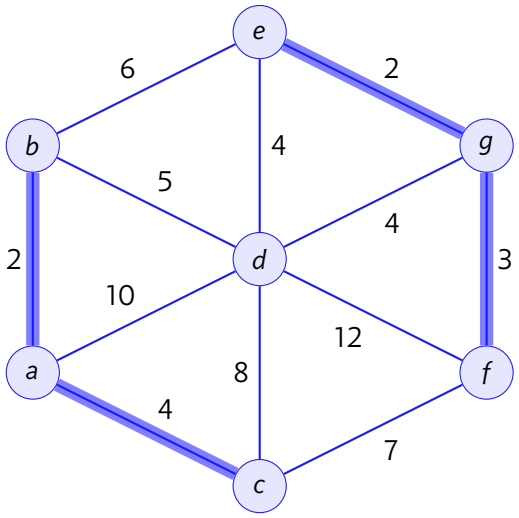
Example



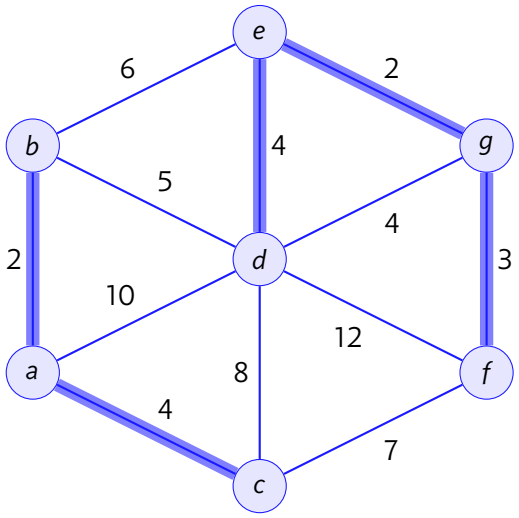
Example



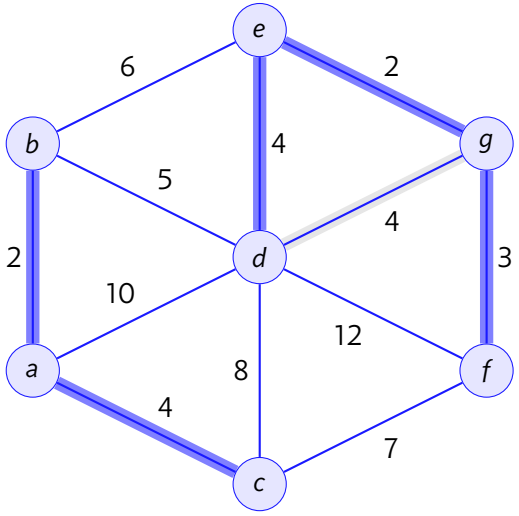
Example



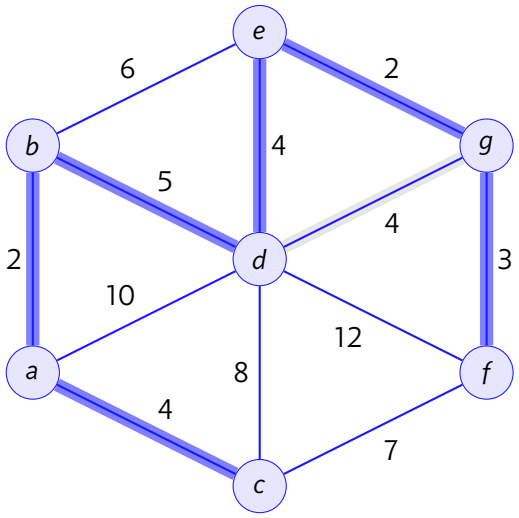
Example



Example



Example



Implementation (from the textbook)

```
vector< pair<int, ii> > EdgeList;
vector< pair<int, ii> > result;
for (int i = 0; i < E; i++) {
    scanf("%d %d %d", &u, &v, &w);
    EdgeList.push_back(make_pair(w, ii(u, v)));
}
sort(EdgeList.begin(), EdgeList.end());
int mst_cost = 0;
UnionFind UF(V);
for (int i = 0; i < E; i++) {
    pair<int, ii> front = EdgeList[i];
    if (!UF.isSameSet(front.second.first, front.second.second)) {
        mst_cost += front.first;
        result.push_back(front);
        UF.unionSet(front.second.first, front.second.second);
}  }
```

Implementation in Kotlin

```
0 data class Edge(val src: Int, val dst: Int, val w: Int)
1 fun main() {
2   val E = readln().toInt()
3   repeat (E) {
4     val (u,v,w) = readln().split(' ').map { it.toInt() }
5     edges.add(Edge(u,v,w))
6   }
7   val sortedEdges = edges.sortedBy { it.w }
8   val result = mutableListOf<Edge>()
9   var mst_cost = 0
10  for (edge in sortedEdges) {
11    if (! UF.isSameSet(edge.src, edge.dst) ) {
12        result.add(edge)
13        UF.union(edge.src, edge.dst)
14        mst_cost += edge.w
15  } } }
```