Kruskal's Algorithm for Minimum Spanning Tree
Kruskal's Algorithm

- Uses a ‘forest’ (a set of trees).
  - Initially, each vertex in the graph is its own tree.
  - Keep merging trees together, until end up with a single tree.
- Pick the smallest edge that connects two different trees.

- Time complexity: \( O(\text{Elg}V) \)
  
  Note: \( \text{Elg}E = O(\text{Elg}E^2) = O(2\text{Elg}V) = O(\text{Elg}V) \)

  Depends on: 1. Sort edges (with what method?) or use a Min-Heap? 2. Find-Set and Union => \( O(\text{lg}V) \) (with union-by-rank or weighted-union) – See the Union-Find slides for more information.

```
MST_Kruskal(G,w)  // N = |V|  ----> O(\text{Elg}V)  (for adj list representation)
1   A = empty set of edges
2   int id[N], sz[N]
3   For v = 0 -> N-1
4       id(v) = v; sz(v)=1
5   Sort edges of G in increasing order of weight
6   For each edge (u,v) in increasing order of weight  ---> O(E)
7       if Find_Set(u,id) == Find_Set(v,id)  -------> \( \Theta(\text{lg}V) \)
8           add edge (u,v) to A
9           union(u,v,id,sz)  -------> \( \Theta(\text{lg}V) \)  (\( \Theta(1) \) when given the representatives)
10  return A
```
Kruskal's Algorithm

Uses a ‘forest’ (a set of trees).

– Initially, each vertex in the graph is its own tree.
– Keep merging trees together, until end up with a single tree.
  • Pick the smallest edge that connects two different trees

• The abstract description is simple, but the implementation affects the runtime.
  – How to maintain the forest
    • See the Union-Find algorithm.
  – How to find the smallest edge connecting two trees:
    • Sort edges: Y/N?
    • Put edges in a min-heap?
Kruskal’s Algorithm

Idea:
- Initially, each vertex in the graph is its own tree.
- Keep merging trees together, until end up with a single tree (pick the smallest edge connecting different trees).

See Union-Find slides as well.

MST_Kruskal(G,w) // N = |V|
1  A = empty set of edges
2  int id[N], sz[N]
3  For v = 0 -> N-1
4     id(v) = v;  sz(v)=1
5  Sort edges of G in increasing order of weight
6  For each edge (u,v) in increasing order of weight
7     if Find_Set(u,id) == Find_Set(v,id)
8         add edge (u,v) to A
9     union(u,v,id,sz)
10 return A
Kruskal’s
Edge (4,6,10) was not picked b.c. it makes a cycle.
Kruskal’s Algorithm and the Union-Find Structure

• Note the Union-Find method is under the “Data Structures for Disjoint Sets” in CLRS, Chapter 21, page 561,
Kruskal’s Algorithm Example 2
Kruskal's Algorithm: Example 2 workout
Kruskal's Algorithm: Example 2 workout