Minimum Spanning Tree Algorithms

MST Algorithms

- Kruskal
- Prim
- Boruvka

In case of unweighted graphs or just (not minimum) spanning tree use DFS or BFS tree.

Kruskal - Idea

```
Input: G = (V,E,l)
Output: T = (V,F)

sort(E)
F := ∅

for i:=1 to size(E)
  if (V, F ∪ {e[i]}) contains no cycle
    then F = F ∪ {e[i]}

return (V, F)
```

Union Find Structure (UFS)

- Two operations
  1. `find` - gives representative for a set
  2. `union` - unites two sets
- Tree structure
- `find`
  1. walk up in the tree till the root is found
  2. return root as representative
- `union`
  1. find both representatives
  2. link representative of smaller height to representative of greater height.
- Use path compression in find operations (link all elements on the path directly to the root)

Kruskal

Kruskal - sort and unite

```
// ... read edges into one array
sort(edges.begin(), edges.end());
Ufs ufs (vertices);
for (unsigned i = 0; i < edges.size(); ++i) {
  int u = ufs.find (edges[i].from);
  int v = ufs.find (edges[i].to);
  if (u != v) {
    ufs.unite (u, v);
    mst.add(u, v);
  }
}
```

Prim

```
// select an arbitrary start vertex and insert into
// priority queue (PQ) with weight 0, all other vertices
// are inserted into the PQ with weight infinity

while (!pq.empty()) {
  pq_pair curr = pq.top(); pq.pop();

  mst.add(curr, pred[curr]);

  for_each(edge e at curr) {
    if (pq.find(e.dest) > e.weight)
      pq.decrease_key(e.dest, e.weight);
      pred[e.dest] = curr;
  }
}
```

MST Properties

Properties to think about

- Shortest Path Tree vs. MST
- Uniqueness
- in-/exclusion of certain light/heavy edges