Notes Week 12

Minimum Spanning Trees

- spanning tree - $(V, T \subseteq E)$ - a connected, acylic subgraph containing all the vertices
  - There are $V - 1$ edges (Weight of ST is the sum of all edge weights in T)
  - A minimum spanning tree exists if and only if G is connected
  - Greed is sometimes a good thing!
- We will explore two greedy algorithms for finding a minimum spanning tree
  - Kruskal (forest) - don't necessarily have a connected subtree until the end
    - Sort edges first
    - Start adding edges (don't create a cycle)
    - Repeat step b until you have a spanning tree
  - Prim's (tree) - always have a connected subtree (we will see, this one is best)
    - Start with any root
    - Choose smallest weight edge coming out of it
    - Choose smallest weight edge coming out of any connected vertex (don't create a cycle)
    - Repeat step c until you have a spanning tree

Kruskal's Algorithm

Algorithm 1 Kruskal's Algorithm

```
procedure KRUSKAL
    A ← ∅
    for each vertex v ∈ G do
        make_set(v)
    end for
    Sort edges of G in increasing order of weight
    for each edge (u, v) do
        if Find_Set(u) ≠ Find_Set(v) then
            A ← A ∪ {(u, v)}
            union(u, v)
        end if
    end for
end procedure
```

- The sort takes the longest, so the running time is $O(Elog(E))$
- Note that $E = O(V^2)$ - A compelte graph has $\frac{n(n-1)}{2}$ edges
- If two vertices are in the same set, and are connected, adding an edge between them would create a cycle
- A union operation joins two sets into one (adding an edge merges two trees into one)
- This is a greedy algorithm
- Find Set(u) runs in **Constant Time** - maintain, for each vertex, a pointer to the set it belongs to
- Union(u,v) runs in **Constant Time** - implement as linked lists, attach one to the end of another