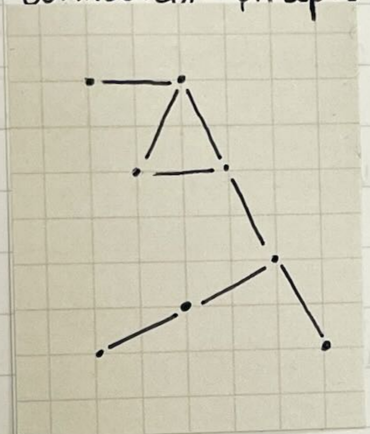
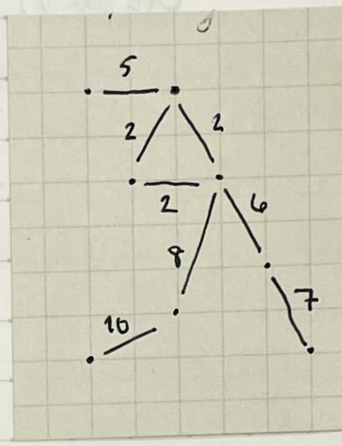
Date 2025.05.07

Minimum Spanning Tree (MST

Connected Graph





- May or may not be unique
- Have unique weights — unique solution
- Minimum weight subgraph in which all nodes are connected

Basic Idea

- Let the solution set $A = \{\}$ (solution set of edges)
  while A is <u>not</u> a MST
- Find an edge (u,v) that is safe for A
  $A = A \cup (u,v)$ (A equals A union edge (u,v))

How To Find A "Safe Edge"?

↳ Two methods — two algorithms

Finding A Safe Edge (Think of bipartite)

- A cut $(S, V-S)$ is a partition into disjoint sets $S, V-S$
- Any edge (u,v) either <u>cross cut</u> or <u>respect cut</u>
  - A cut respects 'A' if no edge in 'A' crosses the cut
    - An edge is a light edge if it cross cuts with minimum weight

Date

Greedy Algorithms

- Kruskal's $O(E \lg V)$
- Prims $O(E \lg V)$
  - ↳ $O(V^2)$ – Adjacency Matrix
  - ↳ $O(E + \lg V)$ – Fibinaci Heap

↳ Useful Info.

PQ assign. as "implemented by us"

↳ bin heap + Adjacency List

Kruskal's Algorithm

- Requires a disjoint set (union find) data structure

$A = \{\}$

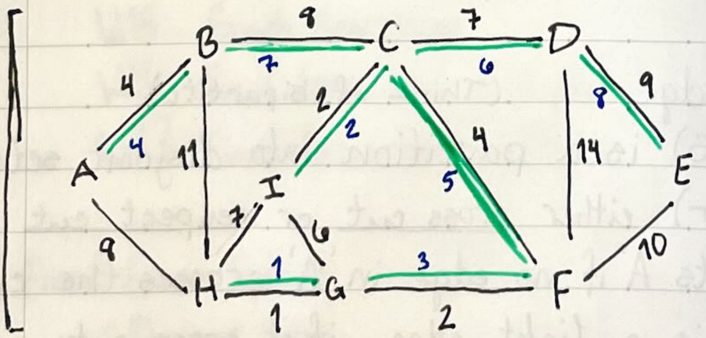
Disjoint Set ds;

★ This is for weighted, undirected graphs

```
for each node v ∈ V
    ds.MakeSet(v);
sort E in increasing order
for each edge (u,v) from the sort (in order)
    if (FindSet(u) != FindSet(v))
        A = A ∪ (u,v)   //Add an edge (u,v) to set A (solution set)
        ds.union(u,v)
```

union

Example Graph



★ Didn't add/union nodes I → G, H → I, & Etc. because they are already in the set

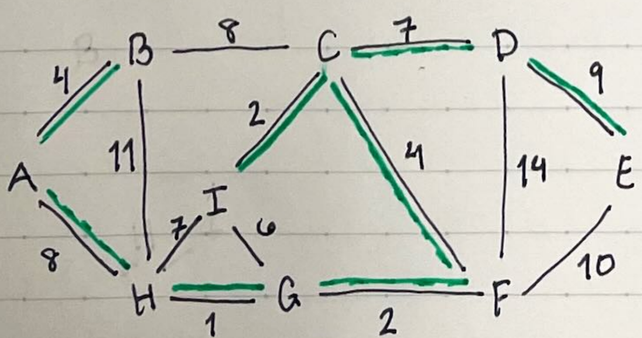
Total Cost MST: 37

Prims Algorithm (Prims(G))

- Take a starting node
- Use a minimum (distance) priority queue behind the scenes

```
Prims(graph G, vertex s)
   for each vertex v in V {
      d[v] = ∞;          //distance
      p[v] = Ø;          //parent
      PQ.insert(v);      //indexed by distance
   }
   PQ.decreaseKey(s, 0);     //start node distance at zero
   d[s] = 0;   //zero
   while (!PQ.empty())
      u=PQ.extractMin();
      for each node v ∈ adjacencyList[u];
         if (PQ.contains(v) && weight(u,v) < d[v])
            p[v]=u;
            d[v] = weight(u,v);
            PQ.decreaseKey(v, d[v]);
         }
      }
}
```

Start
C 0
i
f
g
h
d
a
b
e



Start at C

PQ: ~~c~~ ~~a~~ ~~b~~ ~~d~~ ~~e~~ ~~f~~ ~~g~~ ~~h~~ ~~i~~

Total Cost: 37