Minimum Spanning Tree

Spanning tree of a connected undirected graph G = a subgraph that is a tree and connects all verticies.

There can be many spanning trees of a graph.

BFS and DFS both generate spanning trees BFS is typically 'short and bushy' DFS is typically 'long and stringy'

A *minimum spanning tree* is a spanning tree of a graph with the smallest weight.

$$Weight = \sum_{edges} weight(edge)$$

- A town has a set of houses and a set of roads
- A road connects 2 and only 2 houses
- A road connecting houses u and v has a repair cost $w(u, v)$

Goal: Repair enough (and no more) roads such that

1. Everyone stays connected: can reach every hoes from all other houses
2. Total repair cost is minimized

For an unweighted graph, any spanning tree is a minimum spanning tree.

Finding an MST is an optimization problem Two greedy algorithms:

Kruskal's consider edges in ascending order, at each step select the next edge as long as it does not create cycle

Prim's start with any vertex S and greedily grow a tree from S. At each step, add the edge of th eleast weight to connect an isolated vertex.

```
Kruskal's Algorithm
start with T = V (no edges)
for each edge in increasing order by weight
    if adding edge does not create a cycle
        add edge to T
```

```
MST-KRUSKAL(G, w) //  w = weights
A = {}
for each vertex v in G.V
    MAKE-SET(v)
sort the edges of G.E into nondecreasing order by weight w
for each edge (u, v) in G.E, taken in nondecreasing order by weight
    if FIND-SET(u) != FIND-SET(v)
        A = A U {(u, v)}
        UNION(u, v)
return A
```

Runtime Analysis

Kruskal Running time

$$\begin{aligned}
\text{Sorting } &= O(E\log E) = O(E\log v^2) = O(2E\log V) = O(E\log V)\\
\text{Disjoint-set operations} &= O(m\alpha(n)) = O((2V+2E-1)\alpha(V)) = O(E\alpha(V))\\
O(E\log V) + O(E\alpha(V)) &= O(E\log V)
\end{aligned}$$

MSTs are unique only if all edge weights are distinct.