Minimum Spanning Tree

Suppose we have a set of cities, roads.

Big snowstorm hits
- Can't afford to have all roads open but need access b/w every pair of cities
- Roads have costs

Goal: find set of roads of minimum total cost s.t. there is a path b/w every pair of cities.

What can we say about this set?
⇒ Will make up a tree ⇒ No cycles!

Why?

All three edges cannot be in minimum cost set since any two allows access to all three nodes.
Minimum Spanning Tree (MST)

\( G = (V, E) \)

\( V = \{ v_1, v_2, \ldots, v_n \} \)

\( E = \{ (v_i, v_j) : v_i, v_j \in V \} \)

For \((v_i, v_j) \in E\), \(c(v_i, v_j) > 0\) is cost of \((v_i, v_j)\).

Goal: Find a subset of edges \( T \subseteq E \) such that:

\((V, T)\) is connected and total cost \( \leq \sum c(v_i, v_j) \) is minimized.

Two Greedy Approaches work:

1. Start with any node, continuously choose the cheapest node to add to current connected graph.
2. Continuously choose the cheapest edge that doesn't cause a cycle.

Start with (1) = how to implement?
(1) Very similar to Dijkstra's!

What changes?

```plaintext
Prims-NST (G = V, E) //G is undirected weighted graph:
1. Create empty set T
2. For all v ∈ V: v.cost = ∞, v.spanned = false, v.pred = null
3. Choose random vertex s; s.cost = 0, s.spanned = true.
4. VHeap = BuildHeap(IVI) // Min-Heap ordered by cost.
5. for i = 1 to IVI
6. V = VHeap, delete Min()
7. V.spanned = true
8. if V = s, add (v, v.pred) to T
9. for each neighbor u of v:
10. if (!u.spanned and u.cost > Cv,n)
11. u.cost = Cv,n // VHeap is updated.
12. u.pred = v

'How to get edges of T?'
```

ex.

\[ \text{edges in T:} (c,s) \ (d,c) \ (b,d) \ (e,d) \]
Run Time

Same as Dijkstra's: $O(\frac{1}{2}V \log V + E \log V)$
(2) $c_{T,x} < \underline{w}_{u,v}$

Then $x \cdot \text{cost} < v \cdot \text{cost} \Rightarrow \text{contradiction}$.
**Kruskal's (V, E)**

1. Create empty set T.
2. \( \text{EdgeHeap} = \text{BuildHeap}(E) \)
3. while ( \( |T| < |V|-1 \) )
   
   \( (u,v) = \text{EdgeHeap}.\text{deleteMin}() \)
   
   add \((u,v)\) to T if it does not create a cycle
   
   (via Disjoint Sets Data Structure)

**Time:**

\[
O(\ell E \log(\ell E)) \quad \text{w/d. sets} \quad O(|V| (\log(\ell E) + |V| + |E|)) \\
= O(|V|^2 + |V||E|)
\]

**Optimal?**

Similar proof applies.

Always choosing min cost edge that connects tree to non-tree

\[T = \{ (c,d), (b,d), (s,c), (d,e) \} \]

\[\text{not } (b,c)\]