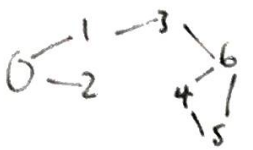
CS Lecture #30 Min spanning trees.

Warm up:

Given an undirected graph, determine if it contains cycles.




Approach 1: Do DFS from arbitrary vertex.

→ keep going until you see a marked vertex.

→ However, make sure you don't count the node you come from

Worst case RT: $\Theta(V+E)$

Approach 2: Use a WQU object

→ For each edge, check if the 2 vertices are connected.

- if not, union them
- if so, there is a cycle.

$O(V+E \log^* V)$ w/ path compression

## Spanning Trees

Given an undirected graph, a spanning tree T is a subgraph of G where

1) T is connected ] → tree properties
2) T is acyclic ]
3) Includes all of the vertices → span

A minimum spanning tree is the smallest spanning tree of a graph.

Min spanning tree vs. Shortest Paths tree.

- A shortest path tree depends on the start vertex
  - it tells you how to get from source to everything.
- There is no source for a MST.
- However, sometimes, an MST can be the SPT for a specific vertex.

The cut property:

- A cut is an assignment of a graph's nodes to 2 non-empty sets.
- A crossing edge is an edge which connects a node from one set to a node from the other set.

Cut property: given any cut, minimum weight crossing edge is in the MST

Generic MST Finding alg:

Start w/ no edges in MST

1) Find a cut that has no crossing edges in mst.
2) Add smallest crossing edge to mst.
3) Repeat until V-1 edges.

* We need to find a cut w/o crossing edges. Random is not best idea.

Prim's algorithm

Start from an arbitrary start node.

- Repeatedly add shortest edge that has one node inside the MST.
- Repeat until V-1 edges.

Implementation

Better way to do it w/ a PQ + fringe.

- Insert all vertices into a fringe PQ, storing vertices in order of distance from previous node (tree).
- Repeat: Remove closest vertex v from PQ, relax all edges pointing from v.

Prim's vs. Dijkstra's

→ They're exactly the same, except Dijkstra's considers distance from source rather than Prim's distance from tree.

Prim's Runtime

Overall: $O(V^* \log V + V^* \log V + E^* \log V)$

- Assuming $E > V$, this is $O(E \log V)$

## Kruskal's Algorithm

Insert all edges into PQ.
Repeat: Remove smallest weight edge. Add to MST if no cycle is created.
* We use a WQU to check for cycles.
  - isConnected will return true if there is a cycle.

Kruskals & Prim get the same MST

## Runtime

$O(E \log E)$

| Operation | Number of times | Time per operation | Total Time |
|---|---|---|---|
| Insert | $E$ | $O(\log E)$ | $O(E \log E)$ |
| Delete min | $O(E)$ | $O(\log E)$ | $O(E \log E)$ |
| Union | $O(V)$ | $O(\log^* V)$ | $O(V \log^* V)$ |
| isConnected | $O(E)$ | $O(\log^* V)$ | $O(E \log^* V)$ |

Total: $O(E + V \log^* V + E \log^* V) = O(E \log^* V)$