Week 5 - Friday

COMP 4500

Last time

- What did we talk about last time?
- Scheduling to minimize lateness
- Dijkstra's algorithm
- Started minimum spanning trees

Questions?

Back to MST

Cut Property

- Assume all edge weights are distinct.
- Let S be a subset of nodes that is neither empty nor equal to V.
- Let edge $e = (v, w)$ be the minimum-cost edge with one end in S and the other in $V - S$.
- Every minimum spanning tree contains e.

Proof of Cut Property

- Let T be a spanning tree that does not contain e. We will try to find an edge e' in T that is more expensive than e that we can swap with e to make a cheaper spanning tree.
- The ends of e are v and w. Since T is a spanning tree, there must be a path P in T from v to w. Following P, we will eventually reach a node w' that is in $V - S$.
- Let $v' \in S$ be the node just before w' on P.
- Let $e' = (v', w')$ be the edge between v' and w'.

Proof continued

- If we exchange e for e', we get edges $T' = (T - \{e'\}) \cup \{e\}$.
- T' is connected since T was connected and any path that used to cross (v', w') can follow the part of P from v' to v, the edge e, and then the part of P from w to w'.
- T' is acyclic since the only cycle in $T' \cup \{e'\}$ is the one made up of e and path P, but it's gone since e' was deleted.
- Both e and e' have one end in S and the other in $V - S$, but e is the cheapest edge with this property, so its weight is lower.
- Thus, T' has lower cost than any spanning tree T that does not include e. ■

Kruskal's algorithm produces an MST

- **Proof:** Whenever we add an edge $e = (v, w)$, let S be the set of nodes that v has a path to before adding e. Node $v \in S$. But $w \notin S$, because e would otherwise create a cycle. Since e is the cheapest edge with one end in S and the other in $V - S$, the Cut Property says it must be part of every minimum spanning tree.
- Thus, Kruskal's algorithm adds exactly those edges that must be part of every minimum spanning tree. ■

Cycle Property

- Assume that all edge costs are distinct. Let C be any cycle in G, and let edge $e = (v, w)$ be the most expensive edge in C. Then e does not belong to any minimum spanning tree of G.

Proof of Cycle Property

- Let T be a spanning tree that contains e. We can show that it doesn't have minimum cost.
- If we delete e from T, it partitions nodes into two components, S, containing v, and $V - S$, containing w.
- The edges of cycle C, with e removed, form a path P from v to w. There must be some edge e' on P that crosses from S to $V - S$.

Proof continued

- Consider the set of edges $T' = (T - \{e\}) \cup \{e'\}$.
- T' must be connected and have no cycles; thus, T' is a spanning tree.
- Since e is the most expensive edge in C, e' is cheaper, and T' is cheaper than T. ■

MST reflections

- Using the **Cut Property**, it's easy to show the correctness of Prim's algorithm
- Using the **Cycle Property**, it's easy to show the correctness of the Reverse Kruskal's algorithm
- It turns out that **any** algorithm that follows the Cut Property to add edges to a spanning tree **or** any algorithm that follows the Cycle Property to remove edges from a graph (or any combination of the two) will find an MST

What about when some edges have the same cost?

- In all MST algorithms, if there is a choice between edges with the same cost, either can be chosen
  - Provided that connectivity/cycle constraints are met
- A way to demonstrate this is to add tiny random amounts to the weights of all edges, much smaller than the difference between any non-equal cost edges
- These random changes serve as tie-breakers between edges of the same cost
  - However, they will not change the structure so that larger edges would have been chosen

Review

Big Oh

- Order the following functions by rate of growth:
  - $(\sqrt{2})^{\log n}$
  - n^2
  - $n!$
  - $(\log n)!$
  - $\left(\frac{3}{2}\right)^n$
  - n^3
  - $(\log n)^2$
  - $\log(n!)$
  - 2^{2^n}
  - $\log(\log n)$
  - $n \cdot 2^n$
  - $n^{\log(\log n)}$
  - $\log n$
  - 1
  - $2^{\log n}$
  - $(\log n)^{\log n}$
  - $4^{\log n}$
  - $(n+1)!$
  - $\sqrt{\log n}$
  - n
  - 2^n
  - $n \log n$

Big Oh

- Determine the running time of various loops

- Example:

```
int counter = 0;
for(int i = 0; i < n*n; ++i)
   for(j = 1; j <= i; ++j)
        counter++;
```

Graphs

- Know basic definitions of graphs
  - Nodes
  - Edges
  - Directed vs. undirected
  - Adjacency matrix vs. adjacency lists
  - Trees
  - Connected
  - Strongly connected

Graph algorithms to know

- BFS
- DFS
- Determining bipartiteness
- Find connected components
- Find strongly connected components
- Topological sort

Example proof

- For all $n \in \mathbb{Z}$, if $n^3 + 5$ is odd, then n is even.
- **Hint:** Try a proof by contradiction.

Example proof

- Prove that, for all integers $n \geq 2$,

$$\sum_{i=1}^{n-1} i(i+1) = \frac{n(n-1)(n+1)}{3}$$

- **Hint:** Try a proof by induction.

Example proof

- Prove that, for all integers $n \geq 1$,

$$\frac{\sum_{i=1}^{n} 2i - 1}{\sum_{i=1}^{n} 2n + 2i - 1} = \frac{1}{3}$$

- **Hint:** Try a proof by induction.

Example proof

- Prove that a graph with two or more nodes where each node has a degree of $\frac{n}{2}$ or higher must be connected.
- **Hint:** Try a proof by contradiction.

Upcoming

Next time…

- Exam 1!

Reminders

- **Finish Assignment 2**
  - **Due tonight before midnight**
- Review chapters 1 through 3