• **DISCLAIMER:** These notes are not necessarily an accurate representation of what I said during the class. They are mostly what I intend to say, and have not been carefully edited.

• Main topics:
  - Homework 2 updated. Major change: parts of problem 2.
  - Cut property with proof.
  - Kruskal’s algorithm
  - Prim’s algorithm

• This week in a nutshell:
  - Cut property: edge in MST iff its minimum on some cut.
  - Kruskal’s algorithm: sort by weights, greedy.
  - Prim’s algorithm: start with single vertex, greedily add minimum edge leading out of current piece.
  - Running time: straight implementation $O(nm)$ for Kruskal, $O(n^2)$ for Prim.
  - Union find: merge smaller into bigger, $O(\log n)$ per union, $O(m \log n)$ Kruskal.
  - Priority queue: Prim’s algorithm in $O(m \log n)$ as well.
  - Can also use priority queue to speed up shortest path with positive edge weights, $O(m \log n)$: Dijkstra’s algorithm.

• Comments from Last Time:
  - Definition of cuts are confusing: AGREED.
We start by proving the correctness of the cut property. Its key definition is a cut, set of edges whose removal disconnects the graph. While its a set of edges, it’s easiest defined via the components formed by removing the edges.

Definition 0.1. A cut given by $S \subseteq V, E(S, V \setminus S)$, is the set of edges between $S$ and $V \setminus S$.

Theorem 0.2. For any cut $S$, the minimum weight edge on the cut is in the MST.

The proof relies on the following operation that transforms between trees.

1. Add an edge $e = uv$.
2. Identify the path in $T$ between $u$ and $v$.
3. Remove any edge from this path.

This still gives a tree because the number of edges remains the same, and connectivity is preserved.

Proof. The proof is by contradiction.

Suppose $T$ is a minimum spanning tree that does not use $e = uv$. Without loss of generality by symmetry, assume $u \in S$ and $v \notin S$.

Then consider the path in $T$ between $u$ and $v$. This path goes from $S$ to $V \setminus S$, so must contain an edge $e'$ in the cut $E(S, V \setminus S)$.

Since $e$ is the minimum weight edge in $E(S, V \setminus S)$, we have

$$w_{e'} > w_e,$$

so adding $e$ to $T$ and removing $e'$ from the cycle gives a tree with smaller weight, a contradiction.

This proof leads to the following algorithm due to Kruskal.

1. Initialize $H = \emptyset$.
2. Sort edges in increasing order of weights.
3. Loop through edges in increasing order of weights.
   (a) If endpoints of $e$ are not connected in $H$.
      i. Add $e$ to $H$. 

\[\square\]
To prove the correctness of this algorithm, consider an edge \(e = uv\) that is added. When it is considered, the connectivity of the forest so far is the same as the connectivity of all edges with weights \(< w_e\). The fact that \(u\) and \(v\) are not connected in this graph means there exists a cut separating \(u\) and \(v\) in the graph

\[G^<w_e = \{V, \{e' \in E, w_{e'} < w_e\}\}.\]

This means this cut in \(G\) only contains edges with weights greater or equal to \(w_e\), so \(w_e\) is the minimum weight edge on it.

The fact that this algorithm terminates with a connected graph also means the cut property goes the other way as well. An edge is in the MST if and only if it’s the minimum weight edge on some cut.

Implemented directly, we spend \(O(m \log n)\) sorting the edges, then \(O(n)\) time to check connectivity between vertices in \(H\) at each step. As we do this \(m\) times, the total cost is \(O(nm)\).