http://www.cs.unh.edu/~ruml/cs758
Spanning Trees

- Problems
- Basic Approach

Kruskal's Algorithm

Prim's Algorithm
Problems

lightest total, lightest max, heaviest, ...

network connectivity
power, water distribution
wiring, VLSI

number of edges?
cycles?
Basic Approach

- Starting from $\emptyset$, grow spanning tree by adding edges

<table>
<thead>
<tr>
<th>Spanning Trees</th>
</tr>
</thead>
<tbody>
<tr>
<td>Problems</td>
</tr>
<tr>
<td><strong>Basic Approach</strong></td>
</tr>
<tr>
<td>Kruskal's Algorithm</td>
</tr>
<tr>
<td>Prim's Algorithm</td>
</tr>
</tbody>
</table>
Basic Approach

starting from $\emptyset$, grow spanning tree by adding edges

Theorem: take any cut that respects the nascent tree. A lightest edge crossing the cut can be added to the tree.
starting from $\emptyset$, grow spanning tree by adding edges

Theorem: take any cut that respects the nascent tree. A lightest edge crossing the cut can be added to the tree.

Proof: if a MST $T$ includes our edge, fine. Otherwise, consider an edge in $T$ that crosses cut. Replace it with ours. Still a spanning tree. Cost can't go up, so still minimum.
Kruskal’s Algorithm
connect separate components until spanned
connect separate components until spanned

1. \( T \leftarrow \emptyset \)
2. for each vertex \( v \), \text{MAKE-SET}(v)
3. for each edge \((u, v)\) in nondecreasing order of weight
4. \text{if} \quad \text{FIND-SET}(u) \neq \text{FIND-SET}(v)
5. \quad \text{add edge to } T
6. \quad \text{UNION}(u, v)
7. return \( T \)

correctness?
running time?
asst 10
next week: lecture on Tues, no recitation
Prim’s Algorithm
grow tree until connected
grow tree until connected

1. for each vertex \( v \), \( v.c \leftarrow \infty \) and \( v.\pi \leftarrow \text{nil} \)
2. \( 0.c \leftarrow 0 \)
3. \( Q \leftarrow \text{heap of all vertices} \)
4. while \( Q \) is not empty
5. \( u \leftarrow \text{remove vertex with minimum } c \)
6. for each neighbor \( v \) of \( u \)
7. if \( v \) is in \( Q \) and \( w(u, v) < v.c \)
8. \( v.c \leftarrow w(u, v) \)
9. \( v.\pi \leftarrow u \)
10. return \( \{(u, u.\pi) : v \in V - \{0\}\} \)

correctness? what is the invariant? running time?
For example:

■ What’s still confusing?
■ What question didn’t you get to ask today?
■ What would you like to hear more about?

Please write down your most pressing question about algorithms and put it in the box on your way out.

*Thanks!*