Minimum Spanning Tree
(undirected graph)
Path tree vs. spanning tree

We have constructed trees in graphs for shortest path to anywhere else (from vertex is the root)

Minimum spanning trees instead want to connect every node with the least cost (undirected edges)
Path tree vs. spanning tree

Example: build the least costly road that allows cars to get from any start to any finish
Safe edges

We can find (again) a greedy algorithm to solve MSTs.

We can repeatedly add safe edges to an existing solution:

1. find (u,v) as safe edge for A
2. Add (u,v) to A and repeat 1.
Safe edges

A cut $S$: $(S, V-S)$ for any vertices $S$

Cut $S$ respects $A$: no edge in $A$ has one side in $S$ and another in $V-S$
Safe edges

A cut $S$: $(S, V-S)$ for any verticies $S$
Cut $S$ respects $A$: no edge in $A$ has one side in $S$ and another in $V-S$

$S$ respects $A$ if no red edges

$S = \text{circles}$  \hspace{1cm}  $V-S = \text{squares}$
Safe edges

Theorem 23.1:
Let $A$ be a set of edges that is included in some MST

Let $S$ be a cut that respects $A$

Then the minimum edge that crosses $S$ and $V-S$ is a safe edge for $A$
Theorem 23.1: \[
\text{LHS} = S \\
\text{RHS} = V - S
\]

\text{blue = minimum safe edge} \\
\text{A = red edges}
Safe edges

Proof:
Let $T$ be a MST that includes $A$
Add minimum safe edge $(u,v)$
Let $(x,y)$ be the other edge on the cut
Remove $(x,y)$, and call this $T'$ thus:
\[ w(T') = w(T) + w(u,v) - w(x,y) \]
But $(u,v)$ min, so $w(u,v) \leq w(x,y)$
Thus, $w(T') \leq w(T)$ and we done
Kruskal

Idea:
1. If the minimum edge does not create a cycle, add it (else remove)
2. Repeat 1 until no more edges (or you have a MST)
MST-Kruskal(G,w)
A = { } for each v in G.V: Make-Set(V)
sort(G.E)
for (u,v) in G.E (w(u,v) increasing)
  if Find-Set(u) ≠ Find-Set(v)
    A= A U {(u,v)}
  Union(u,v)
Kruskal
Kruskal

Runtime:
Find-Set takes about $O(lg |V|)$ time (Ch. 21)

Thus overall is about $O(|E| \ lg |V|)$
Prim

Idea:
1. Select any vertex (as the root)
2. Add all edges to sorted TODO list
3. Remove edge from list, if no cycle induced goto 2.

Like Dijkstra, but without relaxation
MST-Primt(G, w, r)  // r is root
for each u in G.V: u.key=∞, u.π=NIL
r.key = 0, Q = G.V
while Q not empty
    u = Extract-Min(Q)
    for each v in G.Adj[u]
        if v in Q and w(u,v) < v.key
            v.key=w(u,v), v.π=u
Prim

Runtime:
Extract-Min(V) is $O(lg |V|)$, run $|V|$ times is $O(|V| lg |V|)$

for loop runs over each edge twice, minimizing (i.e. Decrease-Key())...
$O( (|V|+|E|) lg |V| ) = O(|E| lg |V|) \ (Fibonacci \ heaps \ O(|E| + |V| lg |V|))$