Announcements

- HW due next Tuesday (oral grading)
**DEFINITIONS:**

*any edge is a cut edge*

A graph is a maximal acyclic graph, always with no edge.  

\[\text{BFS/DFS tree}\]

A component of a graph is a maximal connected subset of \( G \).
Setting: a weighted graph

- A graph $G = (V, E)$ together with a function $w: E \to \mathbb{R}$ that gives a weight $w(e)$ to each edge $e \in E$

Goal: Find minimum set of edges which connects everything.

→ tree

(Obvious applications!)

MST
Note:

We'll assume edge weights are unique, so \( w(e) \neq w(e') \) for any \( e, e' \in E \).

Greedy algorithm
Strategy

We'll try to iteratively build the MST.

At each stage, some subgraph of the MST will exist (called a spanning forest).

Key question:

Which edge should I look at adding to $F$ next?
Key Lemma: Let $S$ be any subset of $V$ (besides $\emptyset$ or $V$ itself). Let $e$ be the edge of minimum weight with one endpoint in $S$ and one in $V - S$. Then $e$ is in any MST of $G$. 

[Diagram of a graph with labeled vertices and edges]
Proof: Consider a tree $T$ which doesn't contain $e$. We need to show $T$ is not minimum.

Let $e = \{u, v\}$

$T$ is connected so it contains a $u$ to $v$ path.

Take "first" edge going from $S$ to $V-S$ along this path $\Rightarrow e$.

Take $T - e' + e$
Any path in $T$ that used edge $e$ will now use:

path $u' \to u + e + v \to v'$. 

So $T - e' + e$ is still connected, has $n-1$ edges, so is a tree.

$T - e' + e$ has weight $\leq T$

$\Rightarrow T$ was not MST. $\blacksquare$
A bit further: Suppose we have a spanning forest, $F$.

A safe edge for a component is the minimum weight edge with only one endpoint in that component.

A useless edge is an edge not in $F$ but with both endpoints in the same component.
Note: Our lemma says safe edges should be in the MST!

So an algorithm:

[Add all the safe edges.]

[Recurse.]
This is Borůvka’s algorithm from 1926.
(Also others—often called Sollin’s algorithm.)

Pseudo code—first, find components:

```
Traverse(s):
  put s in bag
  while the bag is not empty
    take v from the bag
    if v is unmarked
      mark v
      for each edge vw
        put w into the bag

TraverseAll(s):
  for all vertices v
    if v is unmarked
      Traverse(v)
```

(see last lecture)
Now:

**Borůvka(V, E):**

\[ F = (V, \emptyset) \]

**TraverseAll(F) \{\text{count and label components}\}**

while \( F \) has more than one component

**AddAllSafeEdges(V, E)**

**TraverseAll(F)**

return \( F \)

**AddAllSafeEdges(V, E):**

for \( i \leftarrow 1 \) to \( V \)

\[ S[i] \leftarrow \text{NULL } \{\text{sentinel: } w(\text{NULL}) := \infty\} \]

for each edge \( uv \in E \)

if \( \text{label}(u) \neq \text{label}(v) \)

if \( w(uv) < w(S[\text{label}(u)]) \)

\[ S[\text{label}(u)] \leftarrow uv \]

if \( w(uv) < w(S[\text{label}(v)]) \)

\[ S[\text{label}(v)] \leftarrow uv \]

for \( i \leftarrow 1 \) to \( V \)

if \( S[i] \neq \text{NULL} \)

add \( S[i] \) to \( F \)

\( O(n+m) = O(m) \)
Runtime:

At each stage, # of components goes down by at least \( \frac{1}{2} \).

\[ T(n) = T\left(\frac{n}{2}\right) + O(m) \]

\[ = O(m \log n) \]

\[ O(E \log V) \] (in notes)
Prim's algorithm: add a safe edge one at a time

(really Jarnik's from 1929)
Code: Actually, similar to DFS, but keep edges in a heap. Take min edge, check if endpoints are both unmarked. If not, add to $T$.

Runtime: $O(m \log n)$

(Can improve with fancy data structures...)
Kruskal's algorithm (1953)

Idea: Scan edges in increasing order. If edge is safe, add it to F.

Since we'll go in sorted order, at each stage the smallest safe edge gets added, so results in MST.
How to implement?

Need a data structure to maintain a forest.

Allow:

- Lookups:
  - is this edge useless?

- Unions (to join 2 components)

Next time - the details!

Union-Find data structure