CSE 101
Algorithm Design and Analysis
Sanjoy Dasgupta, Russell Impagliazzo,
Ragesh Jaiswal
russell@cs.ucsd.edu
Lecture 17: Kruskal’s MST algorithm
Thanks: Miles Jones
A spanning tree of an undirected graph $G=(V,E)$ is a subgraph $G'=(V,E')$ such that $G'$ is a tree and all vertices in $V$ are connected.

An output tree of DFS or BFS is a spanning tree.
Suppose you have a network of computers that were linked pairwise

Suppose each link has a positive maintenance cost. Your job is to keep some links so that the cost of the network is minimized and the network stays connected.

Or each edge is a potential road, with the cost to build it, and you want to be able to drive to any location.
THE MINIMIZED GRAPH IS A TREE

When is a connected undirected graph NOT a tree?
When is a connected undirected graph NOT a tree?

When it has a cycle.

If the subgraph has a cycle, dropping any edge makes a cheaper connected subgraph.
So the minimum cost connected sub-graph is always a tree.
Greedy rule: which edge should we pick first?
Greedy rule: which edge should we pick first?

*Pick the smallest weight edge*
Greedy rule: which edge should we pick first?

*Pick the smallest weight edge*
Greedy rule: which edge should we pick first?

*Pick the smallest weight edge*
Greedy rule: which edge should we pick first? *Pick the smallest weight edge unless its vertices are already connected*
Greedy rule: which edge should we pick first?

*Pick the smallest weight edge unless its vertices are already connected*
Kruskal’s Algorithm for Finding the Minimum Spanning Tree

Start with a graph with only the vertices. Repeatedly add the next lightest edge that does not form a cycle.
Let $e$ be the smallest weight edge, $OT$ a spanning tree that does not contain $e$. Then there is another spanning tree $OT'$ that contains $e$, with $\text{Cost}(OT') \leq \text{Cost}(OT)$.
Let $e$ be the smallest weight edge, $OT$ a spanning tree that does not contain $e$. Then there is another spanning tree $OT'$ that contains $e$, with $\text{Cost}(OT') \leq \text{Cost}(OT)$

e=\{u,v\}, u must be connected to v in OT

Let $p$ be the path from $u$ to $v$ in $OT$, and let $e'$ be any edge in that path
Let e be the smallest weight edge, OT a spanning tree that does not contain e. Then there is another spanning tree OT’ that contains e, with \( \text{Cost}(OT’) \leq \text{Cost}(OT) \)

e = \{u, v\}, u must be connected to v in OT

Let OT’ = OT + e – e’
Let $OT' = OT + e - e'$

Let $e' = \{u, w\}$. Then the path $p$ is of the form $e', p'$ where $p'$ is a path from $w$ to $v$

To simulate $e'$, we can take the following path in $OT'$: $e$, $p'$ in reverse

So since we can simulate any path in $OT$ with a path in $OT'$, and $OT$ was connected, $OT'$ is still connected
Let e be the smallest weight edge, OT a spanning tree that does not contain e. Then there is another spanning tree OT’ that contains e, with Cost(OT’)\leq\text{Cost}(OT)

Let OT’ = OT + e - e'

\[ w(e) \leq w(e') \]

So \( \text{Cost}(OT') = \text{Cost}(OT) - w(e') + w(e) \leq \text{Cost}(OT) \)
If $G$ has at most two vertices, any solution is optimal
Assume Kruskal is optimal for any graph with $n-1$ vertices
Let $e$ be the smallest weight edge
$G'$: Contract the edge $e$ in $G$, treating its two vertices as one vertex
CONTRACTION
Contracted graph is not necessarily simple
If $G$ has at most two vertices, any solution is optimal.

Assume Kruskal is optimal for any graph with $n-1$ vertices.

Let $e$ be the smallest weight edge.

$G'$: Contract the edge $e$ in $G$, treating its two vertices as one vertex.

Kruskal($G$) = $e +$ Kruskal($G'$)

$OT'$ = $e +$ some spanning tree in $G'$

Therefore,

\[
\text{Cost (Kruskal}(G)) = \\
\text{Cost(Kruskal (G')) + w(e) ≤} \\
\text{Cost(spanning tree in G') + w(e) =} \\
\text{Cost(OT')} ≤ \text{Cost(OT)}
\]
Sort edges by weight, go through from smallest to largest, and add if it does not create cycle with previously added edges.

How do we tell if adding an edge will create a cycle?

Naive: DFS every time

  Need to test for every edge, m times

DFS on a forest: only edges added to MST searched

Thus, each DFS is $O(n)$.

Total time $O(nm)$
Next Time

How to implement Kruskal’s algorithm faster
Data structures for disjoint sets

Leading into: Amortized analysis.