Minimum spanning trees

Announcements

- Lab on Thursday

- Last HW will go up today
  (not to be graded)

  one question will be on final
Dfn: Given a weighted graph, find a tree \( T \) such that every vertex is in \( T \) and
\[
\sum_{\{u,v\} \in E} w(\{u,v\}) = w(T)
\]
is minimized.

Such a tree is a minimum spanning tree.
Question:

Why won't BFS/DFS work?

These don't pay attention to weights.

Why not shortest path tree?

Easy counterexample.
Key Fact

Let $G$ be a weighted, connected graph, and let $V_1 \cup V_2$ be a partition of $V$ into non-empty sets. Let $e$ be the minimum weight edge between $V_1 \cup V_2$. Then there is a MST containing $e$.

pf: Suppose $e$ is not in MST.

MST $T$ must have an $u-v$ path which has at least one edge $v \in V_1$ and $v \in V_2$. Delete that edge and replace it with $e$.

\[ \text{MST} \]
So how to use this fact?

1. Start w/ some vertex \( V \)
   put \( S = \emptyset \)
   At each stage, add min edge
   b/t \( S \) and \( V - S \)

2. Know min edge is in MST
Kruskal's algorithm

Build MST in "clusters":

- Initially, each edge is by itself.
- In a loop, take next smallest edge.
  - if e connects two different clusters, add it to MST
  - if e goes between 2 vertices of same cluster, discard it.

Implement: union-find data structure
Why does it work?
Another: Prim's algorithm

Grow MST starting from a vertex. (Similar to Dijkstra shortest path tree)

Keep a set of "reached" vertices. At each step, add lowest weight edge going from a vertex in the set to a vertex outside.

Why? Use fact:

Let \( V_1 = \text{set } S \)
\[ V_2 = V - S \]
• Take min edge b/t partition
Running time: (of Prim's)

Variant of Shortest path tree alg.

Use priority queue

\[ \sum_{v} \Delta(v) \cdot \log n + m \cdot \log n \]

= \( O((n+m) \log n) \)

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