Minimum Spanning Trees

Problem Definition

Algorithms: Design and Analysis, Part II

Overview

Informal Goal: connect a bunch of points together as cheaply as possible.

Applications: clustering (more later), networking.

Blazingly Fast Greedy Algorithms:

- Prim's Algorithm [1957; also Dijkstra 1959, Jarnik 1930]
- Kruskal's Algorithm [1956]

# of edges # of vertices

$\Rightarrow O(m \log n)$ time (using suitable data structures)

Problem Definition

Input: undirected graph $G = (V, E)$ and a cost c_e for each edge $e \in E$.

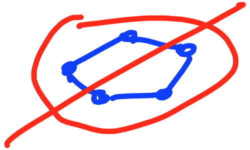
vertices   edges

- assume adjacency list representation (see Pt I for details)
- ok if edge costs are negative

Output: minimum cost tree $T \subseteq E$ that spans all vertices.

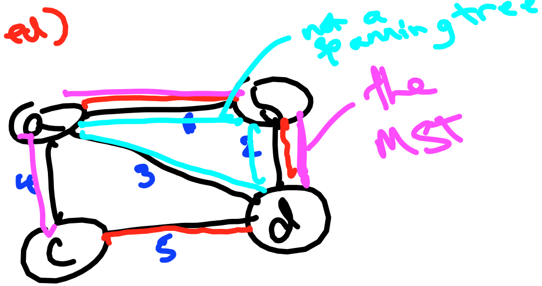
i.e., sum of edge costs

I.e.: (1) T has no cycles



(disallowed)

(2) the subgraph (V, T) is connected (i.e., contains path between each pair of vertices)



Standing Assumptions

Assumption #1: input graph G is connected.

- else no spanning trees
- easy to check in preprocessing (e.g., depth-first search)

Assumption #2: edge costs are distinct.

- Prim + Kruskal remain correct with ties (which can be broken arbitrarily)
- correctness proof a bit more annoying (will skip)