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]

$\Rightarrow O(\underbrace{m}_{\#\text{ of edges}} \log \underbrace{n}_{\#\text{ of vertices}})$ time (using suitable data structures)

Problem Definition

vertices

edges

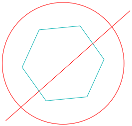
Input: Undirected graph $G = (V, E)$ and a cost c_e for each edge $e \in E$.

- Assume adjacency list representation (see Part I for details)
- OK if edge costs are negative

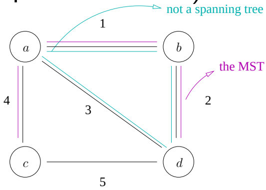
Output: minimum cost tree $T \subseteq E$ that spans all vertices.

i.e., sum of edge costs

I.e.: (1) T has no cycles, (2) the subgraph (V, T) is connected (i.e., contains path between each pair of vertices).



(disallowed)



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).