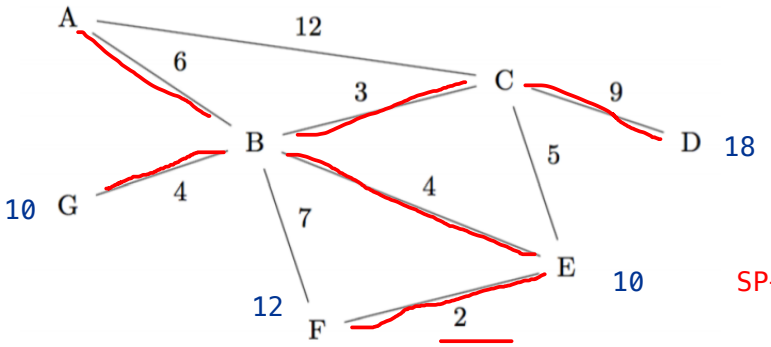
CS 61B Spring 2019

Graphs

Exam Prep 11: April 8, 2019

1 Warmup with MSTs



SP–PQ: A B C G E F D

(a) For the graph above, list the edges in the order they're added to the MST by Kruskal's and Prim's algorithm. Assume Prim's algorithm starts at vertex A. Assume ties are broken in alphabetical order. Denote each edge as a pair of vertices (e.g. AB is the edge from A to B)

Prim's algorithm order: AB/BC/BE/EF/BG/CD

Kruskal's algorithm order: EF/BC/BE/BG/AB/CD

(b) Is there any vertex for which the shortest paths tree from that vertex is the same as your Prim MST?

A

(c) True/False: Adding 1 to the smallest edge of a graph G with unique edge weights must change the total weight of its MST

True. 1. included 2. not included: larger will replace

(d) True/False: The shortest path from vertex A to vertex B in a graph G is the same as the shortest path from A to B using only edges in T, where T is the MST of G.

False. The vertex A or B does not refer to the ones in the graph. Consider: C, E

(e) True/False: Given any cut, the maximum-weight crossing edge is in the maximum spanning tree.

True. Just replace "smallest" with "largest"

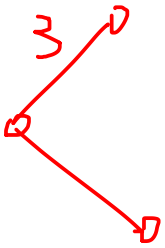
2 Oracle Dijkstra's

In some graph G, we are given a sorted list of nodes, sorted by their distances from some start vertex A. Design an algorithm to find the shortest paths tree starting from A in linear O(V+E) time.

1. Pop the smallest node and set it as the source s.

2. Pop the smallest node x with distance D.
3. Add an edge with the weight as D from s to x.

4. Pop the smallest node x with distance D
5. Check every vertex v so far, iterate adj(v) and find the vertex with the weight w such that distTo[v] + w == D, then add an edge from v to x.
// until the list is empty.



I misunderstood the problem. The G is already there and we just need to pick the tree out.

We just need to rerun the Dijkstra's, but this time the list acts like our PQ.
- Pop the smallest v -> Set the source
- Pop the smallest v -> Iterate previous popped vertices p:
    - If distTo[p] + weight(p, v) = distTo[v] -> Connect them
- Until the list is empty.

1 + 2 + 3 + 4 + … + V
= V^2 NOT OKAY. It is O(V + E)
UNLESS the vertice v can access who point it
OR we know edgeTo in advance

3 Graph Algorithm Design

For each of the following scenarios, write a brief description for an algorithm for finding the MST in an undirected, connected graph G.

(a) If all edges have edge weight 1. Hint: Runtime is O(V+E)

BFS. The shortest path is the MST

Any tree that connects two nodes is MST. So we can use DFS or BFS.

(b) If all edges have edge weight 1 or 2. Hint: Use your algorithm from part (a)

1. O(E): Remove 2-weight edges from the graph so only weight 1 edges remain.
2. O(E + V): Run DFS / BFS. Count the edge we add and union them (logV).
3. If the count == V - 1: It's done. (It won't exceed V - 1)
4. O(E): Otherwise, for each 2-weight edge with condition count < V - 1, check two ending vertices of this edge if they are connected. (remember to union when connecting)

Totally: O(E + (E + V) + (V - 1)logV) = O(E + V + VlogV) = O(E + VlogV)