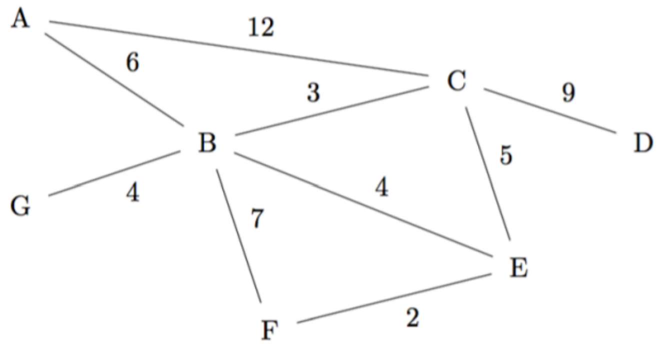
CS 61B Exam Prep 13: Graphs Spring 2020

1 Warmup with MSTs



(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:

Kruskal's algorithm order:

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

(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

(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.

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

2 Graph Conceptuals

Answer the following questions as either **True** or **False** and provide a brief explanation -

(a) If a graph with n vertices has $n-1$ edges, it **must** be a tree.

(b) The adjacency matrix representation is **typically** better than the adjacency list representation when the graph is very connected.

(c) Every edge is looked at exactly twice in **every** iteration of DFS on a connected, undirected graph.

(d) In BFS, let $d(v)$ be the minimum number of edges between a vertex v and the start vertex. For any two vertices u, v in the fringe, $|d(u) - d(v)|$ is **always less than** 2.

(e) Given a fully connected, directed graph (a directed edge exists between every pair of vertices), a topological sort can never exist.

3 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.

4 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)

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