CS 61B Fall 2020 — Graphs

Exam Prep Discussion 11: November 2, 2020

1 Conceptual Shortest Paths

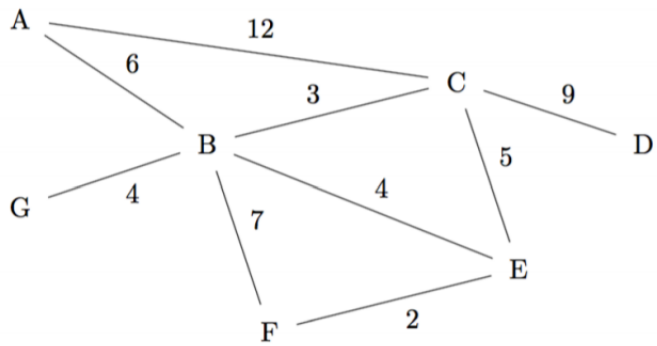
Answer the following questions regarding shortest path algorithms for a **weighted, undirected graph**. If the question is T/F and the statement is true, provide an explanation. If the statement is false, provide a counterexample.

(a) (T/F) If all edge weights are equal and positive, breadth-first search starting from node A will return the shortest path from a node A to a target node B.
True. If all edges are equal in weight, then the shortest path from A to each node is proportional to the number of nodes on the path, so breadth first search will return the shortest path.

(b) (T/F) If all edges have distinct weights, the shortest path between any two vertices is unique.
False. Consider a case of 3 nodes where AB is 3, AC is 5, and BC is 2. Here, the two possible paths from A to C both are of length 5. In general, paths with greater number of edges end up getting penalized more than paths with fewer edges.

(c) (T/F) Adding a constant positive integer k to all edge weights will not affect any shortest path between vertices.
False. Consider a case of 3 nodes where AB is 1, AC is 2.5 and BC is 1. Clearly, the best path from A to C is through B, with weight 2. However, if we add 1 to each edge weight, suddenly the path going through B will have weight 4, while the direct path is only 3.5.

2 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: 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?
Vertex B, A, or G

(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, either this smallest edge (now with weight +1) is included, or this smallest edge is not included and some larger edge takes its place since there was no other edge of equal weight. Either way, total weight increases

(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.
No, consider vertices C and E in the grpah above

(e) True/False: Given any cut, the maximum-weight crossing edge is in the maximum spanning tree.
True, we can use the cut-property proof as seen in class, but replace "smallest" with "largest"

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)

The key idea here is that any tree which connects all nodes is an MST. We can run DFS and take the DFS tree. You could also take a BFS tree, or run Prim's algorithm with a queue or stack instead of a priority queue (this would be equivalent to BFS/DFS). Unfortunately, a modified Kruskal's will be slightly slower, because even if we don't need to sort edges, the union-find operations will take additional time.

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

Remove weight 2 edges from the graph so only weight 1 edges remain. Now run an algorithm from part (a) as far as possible (e.g. find a DFS forest). We will have some number of connected components. Use these connected components as nodes in a new graph G*. Look at the weight 2 edges in G. For each edge, if the nodes containing the two endpoints are not already connected in G*, add an edge between the two containing nodes in G*. Now we can run our algorithm from part (a) again to complete the MST.