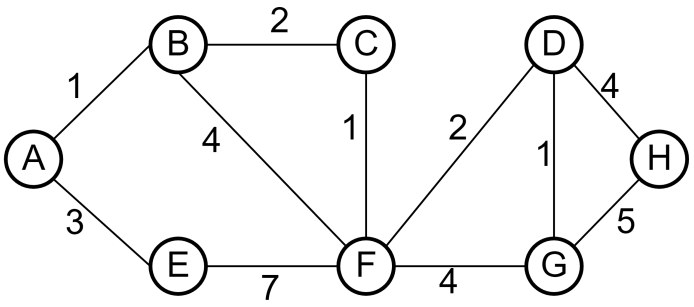
CS 61B Fall 2021 — Graphs

Exam Prep Discussion 13: November 15, 2021

1 DFS, BFS, Dijkstra's, A*

For the following questions, use the graph below and assume that we break ties by visiting lexicographically earlier nodes first.



(a) Give the depth first search preorder traversal starting from vertex A.

A, B, C, F, D, G, H, E

(b) Give the depth first search postorder traversal starting from vertex A.

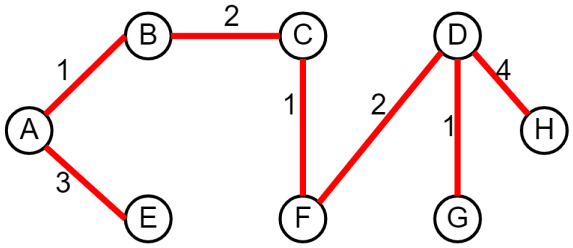
H, G, D, E, F, C, B, A

(c) Give the breadth first search traversal starting from vertex A.

A, B, E, C, F, D, G, H

(d) Give the order in which Dijkstra's Algorithm would visit each vertex, starting from vertex A. Sketch the resulting shortest paths tree.

A, B, C, E, F, D, G, H



(e) Give the path A* search would return, starting from A and with G as a goal. Let $h(u, v)$ be the valued returned by the heuristic for nodes u and v.

u	v	$h(u, v)$
A	G	9
B	G	7
C	G	4
D	G	1
E	G	10
F	G	3
H	G	5

$A \to B, B \to C, C \to F, F \to D, D \to G$

2 Graph Conceptuals

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

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

   False. The graph **must** be connected.

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

   True. The adjacency matrix representation is usually worse than the adjacency list representation with regards to space, scanning a vertex's neighbors, and full graph scans. However, when the graph is very connected, the adjacency matrix representation has roughly same asymptotic runtime in these operations, while "winning" in operations like hasEdge.

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

   True. The two vertices the edge is connecting will look at that edge when it's their turn.

4. 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.

   True. Suppose this was not the case. Then, we could have a vertex 2 edges away and a vertex 4 edges away in the fringe at the same time. But, the only way to have a vertex 4 edges away is if a vertex 3 edges away was removed from the fringe. We see this could never occur because the vertex 2 edges away would be removed before the vertex 3 edges away!

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

False. Consider the graph constructed as follows: for all vertices i, j such that $i < j$, draw a directed edge from i to j. A valid topological ordering of this graph is simply enumerating the vertices: $1, 2, 3, \ldots . N$.

3 Conceptual Shortest Paths

Answer the following questions regarding shortest path algorithms for a **weighted, undirected graph**. If 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, the 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 two vertices.

False. Consider a case of 3 nodes A, B, and C 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.

(d) (T/F) **Multiplying** a constant positive integer k to all edge weights will not affect any shortest path between two vertices.

True. Suppose we have arbitrary nodes u and v. Let's say the shortest path from u to v, before the multiplication by k, was of total weight w. This implies that every other path from u to v was of total weight greater than w. After multiplying each edge weight by k, the total weight of the shortest path becomes $w * k$ and the total weight of every other path becomes some number greater than $w * k$. Therefore, the original shortest path doesn't change.