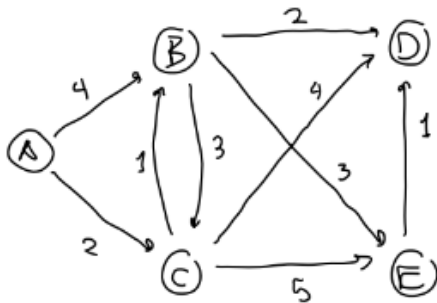

1: Basics

Fill in the definition of each of the following.

1. Bipartite graphs
2. Directed acyclic Graph (DAG):
3. Topological Ordering:
4. Dijkstra:

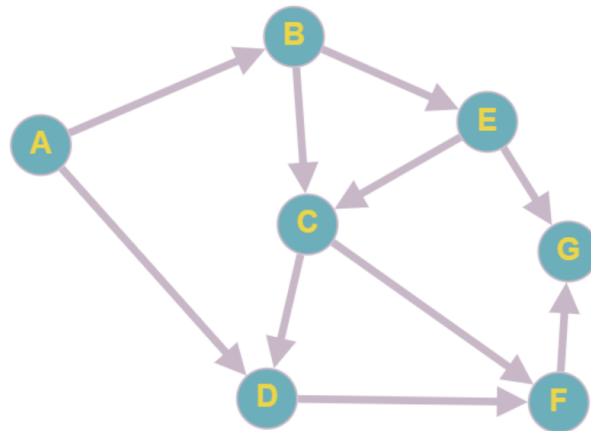
Find the shortest path for the following graph using:

1. Dijkstra

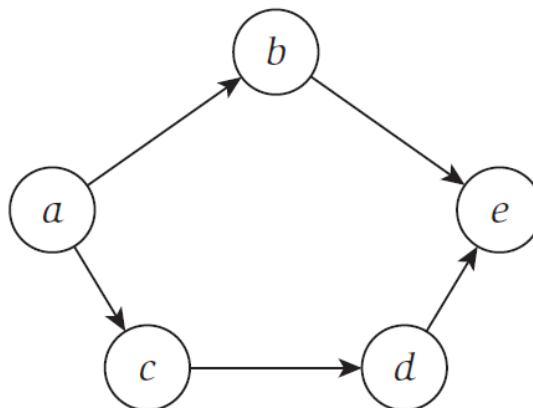


2: Practice Problems

1. Given a graph G with positive integer edges of cost c , how could you alter the graph in a way that would allow you to find the shortest path from s to t using BFS? More precisely, can you construct a graph G' such that you can use BFS on G' to find shortest paths in G ?
2. What is the order of the topologically sorted nodes for the DAG below?



3. How many topological orderings does the graph below have?



4. Show the following statement is false: For a simple directed graph with non-negative edge weights, if exactly one of the non-negative weights is changed to a negative weight, Dijkstra's algorithm will always return a $dist(t)$ equal to the shortest distance value between two nodes s and t .

Helpful Videos:

- Dijkstra's algorithm in 3 minutes, Micheal Sambol, https://youtu.be/_lHSawdgXpI?si=aoGcZiufBrnkb_Ab

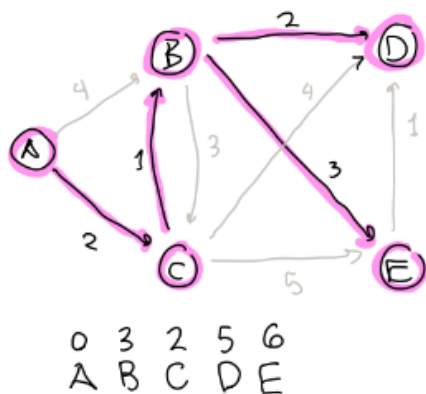
3: Solutions

Fill in the definition of each path finding algorithm.

1. Bipartite Graph: An undirected graphs that you can color with at most 2 colors such that every edge is incident to vertices of opposite color.
2. Directed acyclic Graph (DAG): A graph with directed edges that contain no cycles.
3. Topological Ordering: an ordering of the nodes of G , such that all edges point "forward". Usually is an array, where earlier entries can only point towards later entries, but not vice versa.
4. Dijkstra: The goal is to find the shortest distance from our starting node s to every other node. For each edge (u, v) , if the current known distance to v can be improved by taking the edge (u, v) , then update the distance to v . If $d(v) > d(u) + (u, v)$, then update $d(v) = d(u) + (u, v)$. Each node is only visited once. Does not work on graphs with negative edges. Runtime: $O(E \log(V))$

Find the shortest path for the corresponding graph :

1. Dijkstra: To see a walk through of the answer, please watch the Dijkstra video linked in the recitation packet: https://youtu.be/_lHSawdgXpI?si=aoGcZiufBrnkb_Ab



4: Practice Problems

1. For every edge in G of weight k , split it into k edges of weight 1 in G' with additional intermediate waypoint nodes. Then, G' is an undirected unweighted graph, so BFS works to find shortest paths, and those shortest paths in G' clearly correspond to shortest paths in G by just ignoring the waypoints.
2. $A > B > E > C > D > F > G$
3. The first node in a topological ordering must have no edge leaving it. So a must be first in all the sortings. The last node must be a node that has no edges leaving it. e is the only node with this property. So all

sorts will start with a and end with e . Then, c leads to d , so c must be before d . Now we can order however we want as long as these rules are followed. It leads to 3 possible topological orderings:

a, b, c, d, e (1)

a, c, b, d, e (2)

a, c, d, b, e (3)

4. A simple counterexample to this statement would be to have a graph like the one below, where the $B > C$ edge is changed to a negative number, with the goal being finding the shortest distance from A to C . Because of the way Dijkstra's Algorithm works, traversing the $A > C$ and $A > B$ edges will mark all three vertices as visited. Following this, Dijkstra's will terminate due to having processed all vertices and return 2 as the shortest path from A to C , despite the $A > B > C$ path being much shorter at -95.

