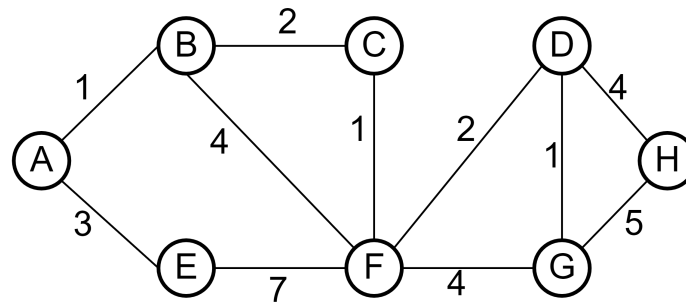


1 Dijkstra's, A*

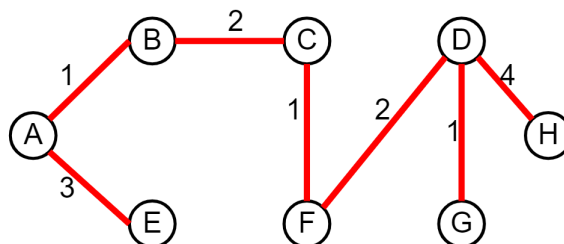


- (a) Run Dijkstra's Algorithm on the graph above starting from vertex A , breaking ties alphabetically. Fill in how the priority values change below. When you remove a node from the fringe, mark it with a check, and leave it blank for the subsequent rows. Stop when you remove G . Also sketch the resulting shortest paths tree in the end.

Solution:

Node	A	B	C	D	E	F	G	H
Start	0	∞	∞	∞	∞	∞	∞	∞
Iter 1	✓	1	∞	∞	3	∞	∞	∞
Iter 2		✓	3	∞	3	5	∞	∞
Iter 3			✓	∞	3	4	∞	∞
Iter 4				∞	✓	4	∞	∞
Iter 5				6		✓	8	∞
Iter 6				✓			7	10
Iter 7							✓	

The SPT of the whole graph is drawn below for your reference. You should be able to “read-off” the edgeTo pointers by locating the last time the priority value of a node changes, and search for where the “checkmark” is for that row - that row would be the edge you came from in the SPT!



- (b) The heuristic distance from all nodes to G is defined below. Run A*, starting from A and with G as a goal. For each entry in the table below, fill in the distance, followed by the priority value of each node,

separated by a comma. Is the heuristic admissible?

u	A	B	C	D	E	F	G	H
$h(u, G)$	9	7	4	1	10	3	0	5

Node	A	B	C	D	E	F	G	H
Start	0, 9	∞	∞	∞	∞	∞	∞	∞
Iter 1	✓							
Iter 2								
Iter 3								
Iter 4								
Iter 5								
Iter 6							✓	

Solution:

Node	A	B	C	D	E	F	G	H
Start	0, 9	∞	∞	∞	∞	∞	∞	∞
Iter 1	✓	1, 8	∞	∞	3, 13	∞	∞	∞
Iter 2		✓	3, 7	∞	3, 13	5, 8	∞	∞
Iter 3			✓	∞	3, 13	4, 7	∞	∞
Iter 4				6, 7	3, 13	✓	8, 8	∞
Iter 5				✓	3, 13		7, 7	10, 15
Iter 6							✓	

The shortest path is: $A \rightarrow B, B \rightarrow C, C \rightarrow F, F \rightarrow D, D \rightarrow G$.

This heuristic is not admissible ($h(A, G) > \text{dist}(A, G) = 7$). However, in this graph A* still returns the correct shortest path from A to G (recall in lecture, admissible and consistent heuristic is only a sufficient condition, so you can still have inadmissible heuristics and still correct results).

2 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.

Solution: 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.

Solution: 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.

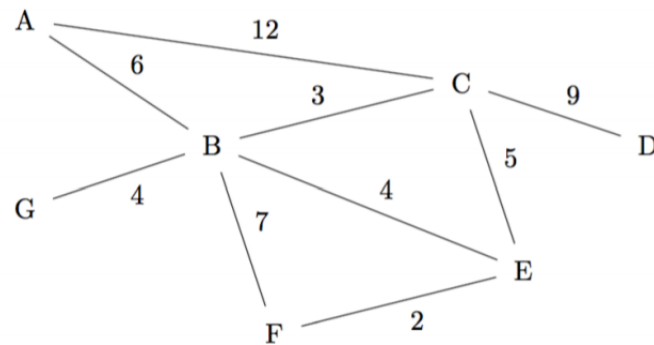
- (c) (T/F) **Adding** a constant positive integer k to all edge weights will not affect any shortest path between two vertices.

Solution: 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.

Solution: 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.

3 Introduction to 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:

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

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

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

Solution: 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, the total weight increases.

- (c) True/False: If all the weights in an MST are unique, there is only one possible MST.

Solution: True, the cut property states that the minimum weight edge in a cut must be in the MST. Since all weights are unique, the minimum weight edge is always unique, so there is only one possible MST.

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

Solution: False, consider vertices C and E in the graph above