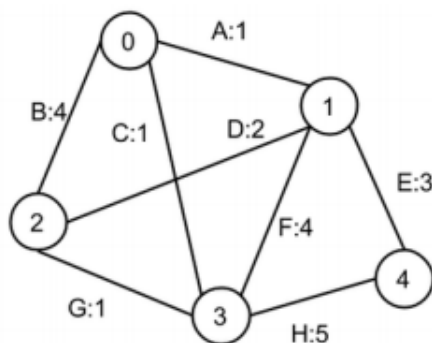


1 Warmup with MST and SP

- (a) For the graph below, use Kruskals algorithm to find the MST. The number on each edge is the weight, and the letter is a unique label you should use in your answer to specify that edge. **Provide the edges in the order theyd be inserted into the MST by Kruskals algorithm.** Break any ties using the alphabetical label. Use the blanks below. You may not need all blanks. **Give your answer in terms of the alphabetical labels**, e.g. if Kruskals starts with the edge between vertices 3 and 4, you would write H in the first blank.



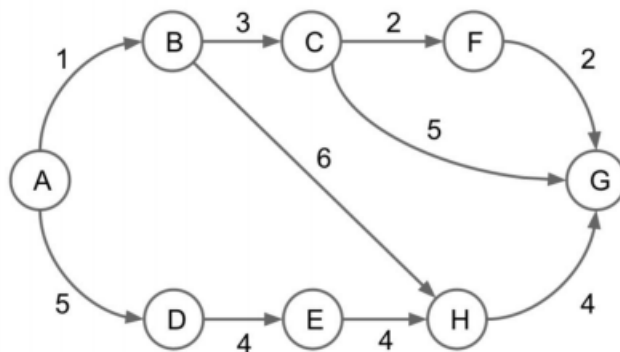
A C E G _ _ _ _ _

- (b) Repeat part a, but using Prim's algorithm, starting from vertex #3. **As before, give your answer in the order added to the MST and in terms of the alphabetical labels.** You may not need all of the blanks.

A C G H E _ _ _ _ _

never forget in Prim's distTo means the distance to the MST under construction!

- (c) For the graph below, give the order in which Dijkstras Algorithm would visit each vertex, starting from vertex A.



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

2 Conceptual MST and SP

Answer the following questions regarding MSTs and 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.

T: because all the edge weights are identical, the total number of edges can represent total weights

- (b) (T/F) If all edges have distinct weights, the shortest path between any two vertices is unique.

F: there are two paths - weight 1 and weight 2 + (-1)

 even applicable to positive edge weights: $1 + 4 = 2 + 3$

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

F: there are two paths: 2 and $1 + 1$; now $2 + k$ and $2 + 2k \rightarrow$ now the number of edges matters

- (d) Draw a weighted graph (could be directed or undirected) where Dijkstra's would incorrectly give the shortest paths from some vertex.


as long as there are negative edge weights

- (e) (T/F) Adding a constant positive integer k to all edge weights will not affect any MST of the graph.

T: because there is one and only one edge between any two vertices

- (f) (T/F) Prim's algorithm works even when there are negative edge weights in the graph.

F: go through the negative-weight edge multiple times can decrease total weights

 True: because to achieve "minimum", you must ensure the total edges is only $V-1$

(g) Why are disjoint sets used in Kruskal's algorithm?

To make sure that there are no cycles.

(h) (T/F) The last edge added to the MST by Prim's algorithm is always the highest weight edge of the MST.

F: it may be just too far away from the initial vertex

3 Shortest Path Algorithm Design


Design an efficient algorithm for the following problem: Given a weighted, undirected, and connected graph G where the weights of every edge in G are all integers between 1 and 10, and a starting vertex s in G , find the distance from s to every other vertex in the graph (where the distance between two vertices is defined as the weight of the shortest path connecting them).

Your algorithm must run asymptotically faster than Dijkstra's.

Hint: What other shortest path algorithms have we learned? Could we possibly modify the graph to apply other SP algorithms?

Dijkstra: $O(E \log V)$

Kruskals: $O(E \log V)$ for sorted edges \rightarrow so we can sort all the edges based on their values from 1 to 10 the sorting can take $O(E)$ time (10 scans)



as for the shortest path problem, it is natural for us to think of BFS, but how can we convert this problem into BFS? Well, we can extend vertices according to their edge weights, for example, a chain of 4 vertices to replace the original edge weight 4. Now we have a time complexity of $O(10E + 10V) = O(E + V)$