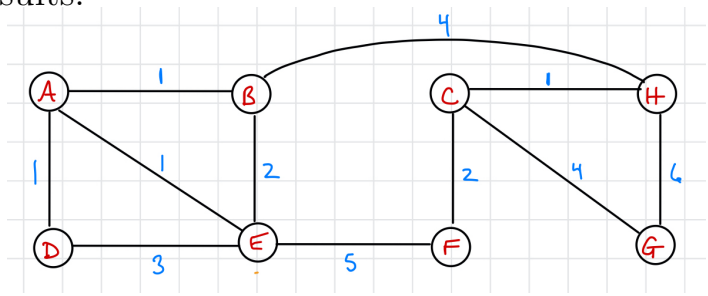
Practice Problem Set 13

. Problem 1

Run Dijkstras algorithm on the weighted graph, using vertex E as the source. (Recall that in an undirected graph, the edges go in both directions). Run Prim's algorithm on the same graph, again starting at vertex E. Explain the difference between the two results.

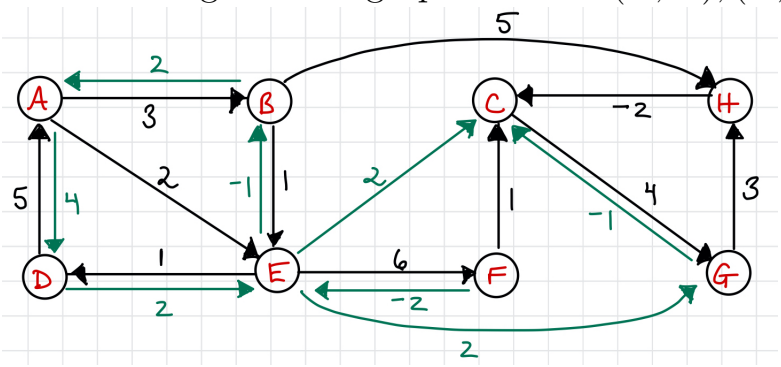


Problem 2

Run Kruskal's algorithm on the above graph to find the MST. Do you get the same result as with Prim's algorithm?

Problem 3

Run Bellman Ford on the weighted directed graph below, using vertex A as the source. Process the edges is lexographic order: $(A, B), (A, C)$, etc.



Problem 4

Show how to update the pseudo-code to Bellman ford so that no unnecessary iterations of the for-loop are performed in Step 2.

Problem 5

A connected graph G is such that the weight of every edge is an integer value in the range V to $2V$. Can you update Kruskal's algorithm so that it runs asymptotically faster?

Problem 6

Suppose a connected weighted graph G is such every edge has a distinct weight. Suppose that a graph G contains a cycle C. The edge with maximum weight on this cycle is edge e. Is it possible that a MST of G contains edge e?

Problem 7

Suppose a connected graph G has weighted edges such that each weight is greater than 1. For vertices x and y in the graph, we want to find the path from x to y that minimizes the *product* of the edge-weights. Explain how to modify Dijkstra's algorithm to solve this problem.

Problem 8

- Suppose G is a directed unweighted graph. Let s be a particular vertex of the graph. Supposed we assign a weight of 1 to every edge in the graph G. When Dijkstra's algorithm is executed from node s, what is the resulting tree? What tree from class is this equivalent to ? Justify your answer. Is there a faster way to have computed this tree?
- Suppose graph G is a weighted directed graph. Some edge-weights are negative, but no weights are less than -10. There are no negative-weight cycles in the graph. We would like to use Dijkstra's algorithm on this graph, so we decide to add 10 to each edge-weight so that there are no negative edge-weights. We apply Dijsktra's algorithm, and then simply subtract 10 from the shortest paths. Explain whether or not this approach works.

Problem 9

Consider a set of n cities which are connected by one-way trains. Not all pairs of cities have a direct route between them. However, for those that do, the time to travel on the train directly from city A to city B is given by $t(A, B)$. You live in one of those cities. Design an algorithm that determines the fastest route to **get home** from each of the n cities. Justify the runtime of your algorithm.

Problem 10

The pseudo-code for Prim's algorithm in class assigns $v.parent = u$ each time edge (u, v) is added to the MST. Update the pseudo-code for Prim's in such a way that each node also stores a list of its children. Use this updated version to write an algorithm that prints out all edges in the MST after the execution of Prim's algorithm.

Problem 11

Suppose G has distinct edge weights. Explain why the smallest-weight edge must be in the MST. Is it also true that the second-smallest edge must be in the MST? What about the third?

Problem 12

Consider a train map that has exactly n stations. Each direct route on the map has a specific travel time. The travel time between stations A and B is given by $t(A, B)$. You and your friend are at different stations. You can both travel using the train routes provided on the map. Your job is to design an algorithm that will determine for each station, which of you will arrive first. What is the runtime of your algorithm?

Problem 13

Given an undirected graph G that contains only one cycle. Design an algorithm that outputs the length of the cycle. Justify the runtime of $O(V + E)$.

Problem 14

A graph G consists of a set of vertices, V, and a set of undirected weighted edges E. Suppose we would like to connect the vertices of G in such a way that the set of edges has no cycle, but we would like to have a set with *maximum* total weight. Describe an algorithm that will solve this problem and determine the runtime.

Problem 15

At the end of your summer vacation in Europe, you find yourself in London, and need to get home to Rome. You have a map of the possible routes you can take to get home. On the map there are a set of n cities, including London and Rome. Pairs of cities are connected by toll roads, and/or train routes. If you drive on a road, you must pay the toll. If you take the train between two cities, you must pay the fare. Not all cities have both options between them. There are a total of $2n$ train routes and $3n$ toll roads.

In London, you currently have a car. You can drive it as long as you want, and in some city you may choose to drop it off (for free) at the rail station. If you drop your car off and start taking the train, you need to take the train all the way home since you can't rent another car. You would like to find the cheapest way home. Your job is to describe how to model this problem as a graph, and describe any attributes the edge and vertex objects would require. Next, design an algorithm that determines the cheapest way home. Justify the runtime of $O(nlogn)$.