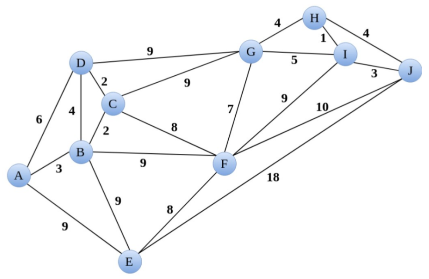
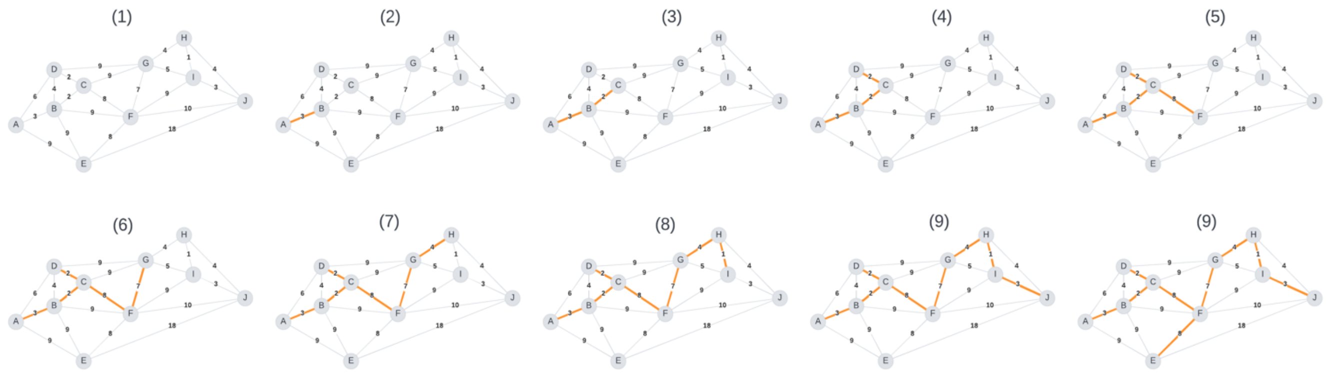
Saleh Alkhalifa CS 5800
10/3/2022 Module 7

Homework#7

Problem 1 Prim's (15 points)

- **Let G be a graph, where each edge has a weight.**
- **A spanning tree is a set of edges that connects all the vertices together, so that there exists a path between any pair of vertices in the graph.**
- **A minimum-weight spanning tree is a spanning tree whose sum of edge weights is as small as possible.**
- **In this question, you will apply Prim's Algorithm on the graph below.**
- **You must start with vertex A.**
- **There are nine edges in the spanning tree produced by Prim's Algorithm, including AB, BC, and IJ.**
- **Determine the exact order in which these nine edges are added to form the minimum-weight spanning tree. Explain each step**





- Using Prim's algorithm, we begin at node A. This node has three different paths, with the minimum at a value of 3 connecting to node B.
- From B, we see that the edge with the minimal value of 2 connects to node C.
- Similarly, the minimal value connects to node D. Node D however connects back to already visited nodes, or node G with an edge value of 9. Comparatively, Node C which we already visited connects to node F with an edge value of 8, which is less than 9.
- Following a similar pattern of pursuing the edge with the smallest value, we go from node F, we can move to node G, and then H, and then I, and finally J. In each of these, we visited the edge with the smallest value.
- Finally, we see that all nodes have been visited with the exception of E. The minimum edge that connects to E stems from F, we can now connect to E via that edge.
- The order can be seen in the visual diagram above.

Problem 2 Dense graph problem (10 points)

Let G be a graph with V vertices and E edges. One can implement Kruskal's Algorithm to run in O(E log V)time, and Prim's Algorithm to run in O(E + V log V) time.

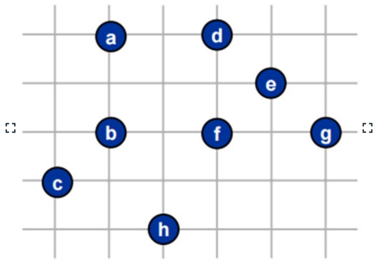
If G is a *dense* graph with an extremely large number of vertices, determine which algorithm would output the minimum-weight spanning tree more quickly. Clearly justify your answer.

- Let us recall that Prim's algorithm (similar to Dijkstra's algorithm), operates by starting at a given root node and traversing the graph via edges with the minimum values until a MST has been created connecting all the nodes.
- We can also recall that Kruskal's algorithm operates by starting with an empty tree, orders the edges in ascending order, and adds edges unless a cycle is created.
- It is important to note that Kruskal's algorithm runs in O(E log V), whereas Prim's runs in O(E + V log V). However, one big difference is that Kruskal's algorithm requires the edges be sorted
- Using a binary heap, we can achieve O(m log n), or O(E log V) with Prim's algorithm, whereas in Kruskal's algorithm we can achieve O(m log n) with sorting.
- Since this problem focuses on a dense graph, there will be many more vertices and edges within this graph, and therefore Prim's algorithm, which does not require a sorting step, will likely be faster.

Homework#7

Problem 3 (10 points total)

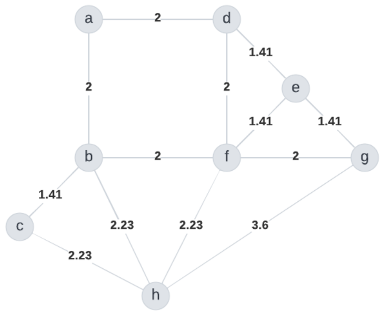
Consider eight points on the Cartesian two-dimensional x-y plane. For each pair of vertices u and v, the weight of edge uv is the Euclidean (Pythagorean) distance between those two points. For example, dist(a,h) = sqrt{4^2 + 1^2} = sqrt{17} and dist(a,b) = sqrt{2^2 + 0^2}=2



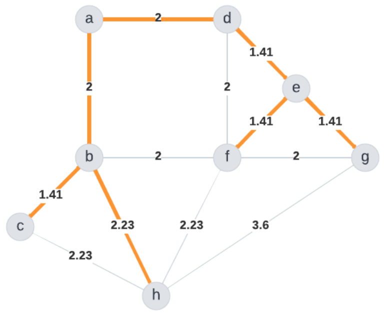
3.1 (5 points)

Using the algorithm of your choice, determine one possible minimum-weight spanning tree and compute its total distance, rounding your answer to one decimal place. Clearly show your steps.

- Since we have a number of vertices, but no specific edges mapped between them, what we can do is use Kruskal's algorithm, and determine the edges based on the Pythagorean theorem mentioned above.
- Recall that Kruskal's algorithm considers the edges of the graph in ascending order based on weight, which in this case is the distance between the points
- We can apply Kruskal's algorithm as illustrated in the following pseudo-code:
  - Start with an empty tree
  - Sort the edges of the original tree in ascending order, based on the distance between them as calculated by the Pythagorean theorem.
  - Select the smallest edge in the list, and add that to the graph unless a cycle is formed
  - Continue to add edges as specified above until all vertices are connected.



Original Graph



Minimum Spanning Tree

- We can calculate the total distance by summing the edges of interest:
  - 2+2+1.41+1.41+1.41+1.41+2.23 = 11.87 = 11.9 (one decimal)

Saleh Alkhalifa
10/3/2022

Homework#7

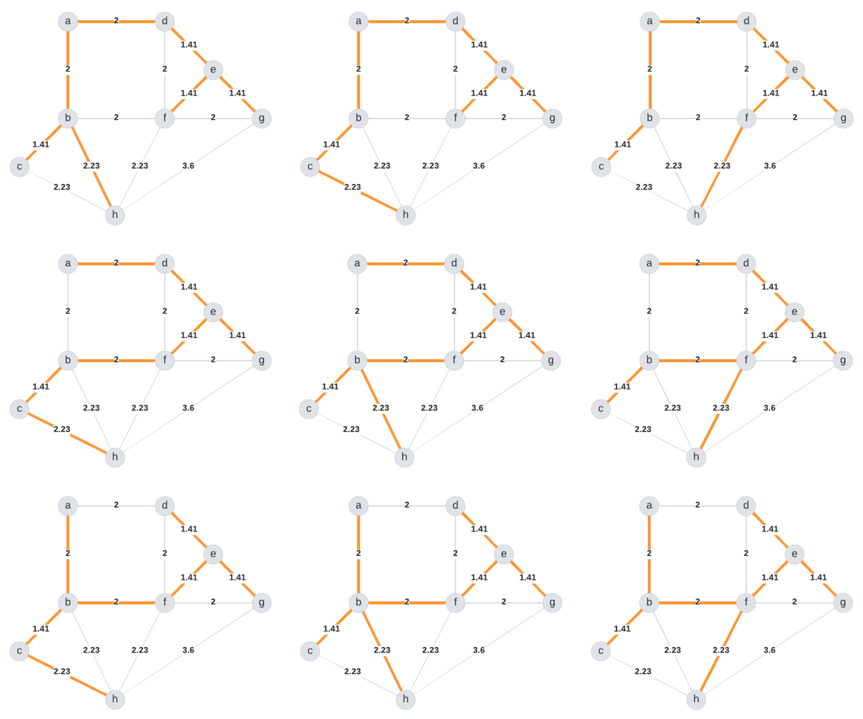
3.2 (5 points)

Because many pairs of points have identical distances (e.g. dist(h,c) = dist(h,b) = dist(h,f) = sqrt{5}), the above diagram has more than one minimum-weight spanning tree.

Determine the total number of minimum-weight spanning trees that exist in the above diagram. Clearly justify your answer.

Answer:

- There are two areas within the graph in which similar values can be found leading to multiple MSTs.
- The first area is within ABFD in which AB, BF, FD, and DA contain weights of 2.
- The second area is within CF, BH, and FH in which all three contain values of $\sqrt{5}$ or 2.23
- Since there are three possible options for each of the the areas, we can calculate the number of possible options as 3*3 = 9, which we can see in the visualization below.



- We can reach this calculation by using Prim's algorithm in the sense that we apply the algorithm as specified before, however, when edges with similar values are observed within the process, the current graph is duplicated and both paths are traversed and the algorithm continues to run on them.
- Eventually a forest will be created comprising the many MSTs, and the cardinality of these trees should be the total number of MSTs.

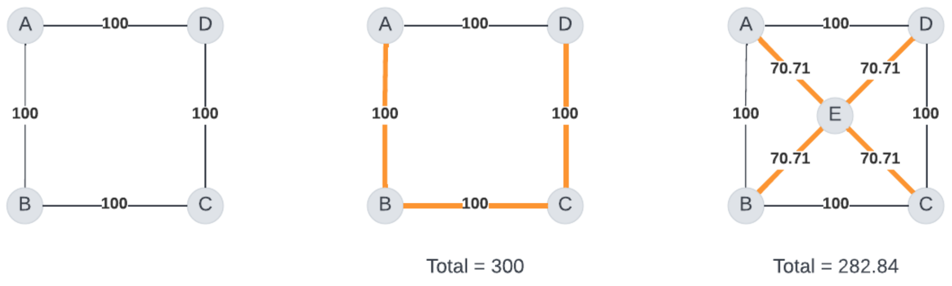
Homework#7

Problem 4: (5 points)

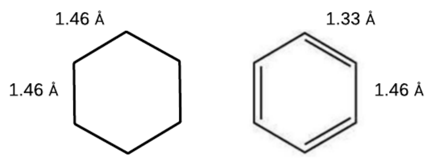
- Let A, B, C, D be the vertices of a square with side length 100.
- If we want to create a minimum-weight spanning tree to connect these four vertices, clearly this spanning tree would have total weight 300 (e.g. we can connect AB, BC, and CD).
- But what if we are able to add extra vertices inside the square, and use these additional vertices in constructing our spanning tree?
- Would the minimum-weight spanning tree have total weight less than 300? And if so, where should these additional vertices be placed to minimize the total weight?
- Let G be a graph with the vertices A, B, C, D, and possibly one or more additional vertices that can be placed anywhere you want on the (two-dimensional) plane containing the four vertices of the square.
- Determine the smallest total weight for the minimum-weight spanning tree of G. Round your answer to the nearest integer.
- **Note:** I encourage you to add n additional points (for n=1, 2, 3) to your graph and see if you can figure out where these point(s) need to be located to minimize the total weight of the spanning tree. You are welcome to use Excel or write a computer program to help you calculate the location of these additional point(s) that will minimize the total weight of the spanning tree.

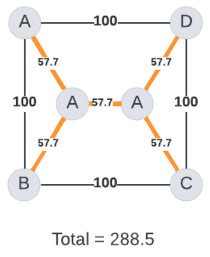
  Answer:

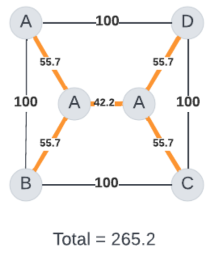
  - If we plot the graph described above, we can see that a total weight of 300 is achieved.
  - If we place another vertex in the center of the graph, and label the vertex as E, we can then connect that vertex to each of the other vertices as seen below



  - The addition of another vertex in the center did in fact yield a MST with a total weight of 282.84 which was less than the original value of 300.
  - Within the field of organic chemistry (my background), we explore the angles of molecular structures and alter them when necessary to adjust the potency of compounds, and the use of algorithms like MST is sometimes of value to determine the optimal angle. One of the most common structures is a benzene ring which takes of the form of a hexagon. If we take the structure of a hexagon and cut it in half, we can see a structure similar to that below, in which we add 2 vertices instead of one:
  - If we think of the box ABCD which now includes two other vertices as "half benzenes" in which the two vertices are centered equidistantly between the other vertices, we can calculate a total weight of 288.5 which is slightly higher than our last calculation.







- However, similarly to the Benzene ring above, its sometimes more favorable for some of the edges to be smaller than others. Within the field of chemistry, we attribute this to a concept known as steric hindrance, however within the field of computer science it is referred to the Steiner tree problem.
- The Steiner ratio is defined as $\frac{2}{\sqrt{3}}$, which is a ratio commonly see in natural phenomena, including within the field of organic chemistry. This ratio is obtained given the angles and lengths of the edges involved.
- Using this concept, we can achieve a weight of 265.2 for this graph, or 265 when rounded.

References:

[1] https://northeastern.instructure.com/courses/117409/pages/module-7-7-dot-1-mst-basics?module_item_id=7833155

[2] https://en.wikipedia.org/wiki/Prim%27s_algorithm

[3] Introduction to Algorithms, Cormen, Third Edition. (CLRS)

[4] LucidChart Visualization Tool