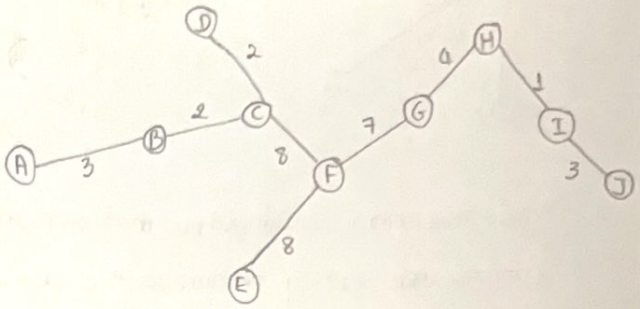
HW7

1. Determine the exact order in which these nine edges are added to form the MST

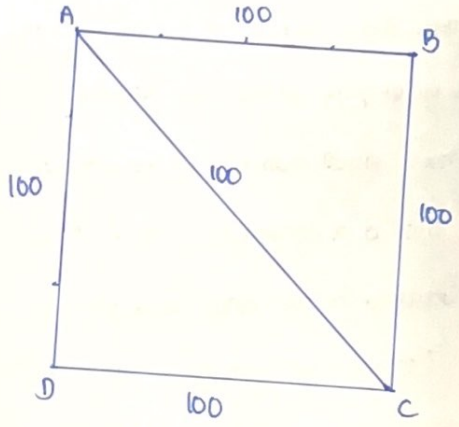
+ We are going to start with node A, ~~we are~~ from A, we are going to search for the minimum edge and that would be B cause (A, B) = 3 which is the smallest among A's neighbor.
+ from A, B, we are going to keep searching for the minimum weight edge and we find C cause (B, C) = 2
+ from all those 3 nodes so far (A, B, C), we will try to find which connected edge that are the smallest and we find (C, d) = 2
+ from (A, B, C, D), we are going to find another connected edge that are smallest, we find (D, B) but we will not pick it cause it will create a cycle, same idea for (A, D). Hence, the next smallest is (C, F) = 8
+ from (A, B, C, D, F), we are going to find the next smallest connected edge and that will not form a cycle and that will be (F, G) = 7
+ From (A, B, C, D, F, G), we are going to search for next smallest connected edge ~~and~~ that will not form a cycle and that will be (G, H) = 4
+ From (A, B, C, D, F, G, H), we will follow the above step and we find the minimum connected edge which is (H, I) = 1
+ From (A, B, C, D, F, G, H, I), we search for the ~~and~~ minimum connected edge that will not form a cycle and that will be (I, J) = 3
+ From (A, B, C, D, F, G, H, I, J), we will search for a minimum edge that will not form a cycle and that will be (F, E) = 8



⇒ cost = 3 + 2 + 2 + 8 + 7 + 4 + 1 + 3 + 8

cost = 38

4. Would the minimum-weight spanning tree have total weight less than 300?

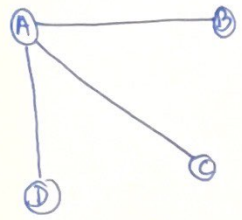


∴ No even if we add extra vertices inside the square, the total weight of the spanning tree will still be 300 cause:

Total = AB + BC + CD = 3(100) = 300

Or = AB + AC + AD = 3(100) = 300

1. Determine the total weight for the minimum-weight spanning tree of G



weight of minimum spanning tree of G is 3

2. Prim's algorithm would output the minimum-weight spanning tree quickly because in this case we are working with dense graph. for Kruskal's, we will have to sort first (sort the edges in descending order of weight) making the time complexity = $O(E \log V)$. For prim's algorithm, the time complexity is $O(E + V \log V)$ when we are using Fibonacci Heap. Hence, if we are dealing with sparse graph, we should use kruskal's cause we have small number of edge to sort. Otherwise if it is a dense graph, then Prim's algorithm would be better since we don't have to sort it. (if we use kruskal, number of edge could be as high as $E = O(V^2) \Rightarrow V^2 \log V$ for time complexity; therefore, no).

3. 3.1 using the algorithm of your choice, determine one possible minimum-weight spanning tree and compute its total distance

$dist(a,h) = \sqrt{4^2 + 1^2} = \sqrt{17} \simeq 4$

$dist(a,b) = \sqrt{4^2 + 0} = 4$ 2

similar calculating will apply to:

$dist(a,d) = \sqrt{2^2 + 0} = 2$

$dist(d,e) = \sqrt{1^2 + 1^2} = \sqrt{2} \simeq 1$

$dist(h,c) = \sqrt{2^2 + 1} = \sqrt{5} \simeq 2$

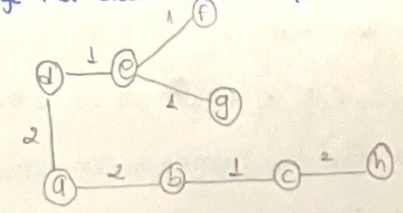
$dist(h,a) = \sqrt{1^2 + 4^2} = \sqrt{17} \simeq 4$

$dist(h,g) = \sqrt{3^2 + 2^2} = \sqrt{9+4} = \sqrt{13} \simeq 4$

all other edges will apply using the same calculation

I am going to apply Prim's algorithm here and the first node we are going to work with is a. 1

- From a, the smallest connected edge that does not form a cycle is b. $dist(a,b) = 2$
- From (a, b), the smallest connected edge that does not form a cycle is c. $dist(b,c) = \sqrt{1^2 + 1^2} \simeq 1$
- From (a, b, c), the smallest connected edge that will not form a cycle is d. $dist(a,d) = 2$
- From (a, b, c, d), the smallest connected edge that does not form a cycle is e. $dist(d,e) = \sqrt{1^2 + 1^2} \simeq 1$
- From (a, b, c, d, e), the smallest connected edge that does not form a cycle is f. $dist(e,f) = \sqrt{1^2 + 1^2} \simeq 1$
- From (a, b, c, d, e, f), the smallest connected edge that does not form a cycle is g. $dist(e,g) = \sqrt{1^2 + 1^2} \simeq 1$
- From (a, b, c, d, e, f, g), the smallest connected edge that does not form a cycle is h. $dist(c,h) \simeq 2$



$\Rightarrow$ total distance = 1 + 1 + 1 + 2 + 2 + 1 + 2 = 10

TOPIC | DATE

3.2 Determine the total number of minimum-weight spanning trees that exist in the above diagram

For total number of spanning tree, we can use Cayley's theorem.

⇒ Total spanning tree = n^{n-2}

$= 8^6 = 262144$

To find minimum spanning tree out of 262144 spanning tree, we could use this algorithm:

Pseudo code

```
def minimum_spanning_tree(spanning_tree) -> int:
    for i in spanning_tree:
        sum all spanning_tree's edge
        store the sum above in the list_of_spanning_tree_sum
    end loop
    for i in list_of_spanning_tree_sum:
        find the minimum spanning tree
        increment count for spanning tree with minimum weight
    end loop
    return count of spanning tree with minimum weight.
```