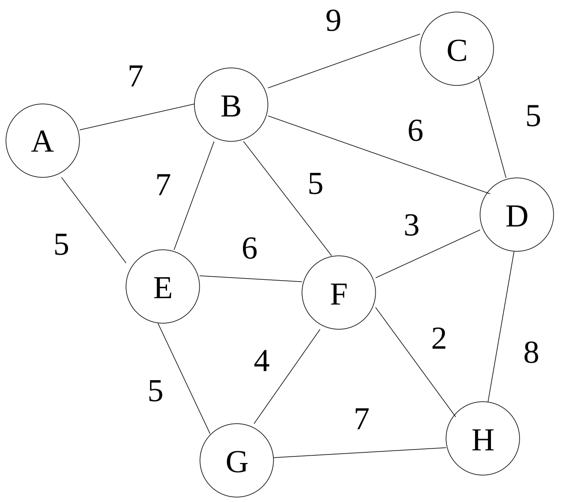
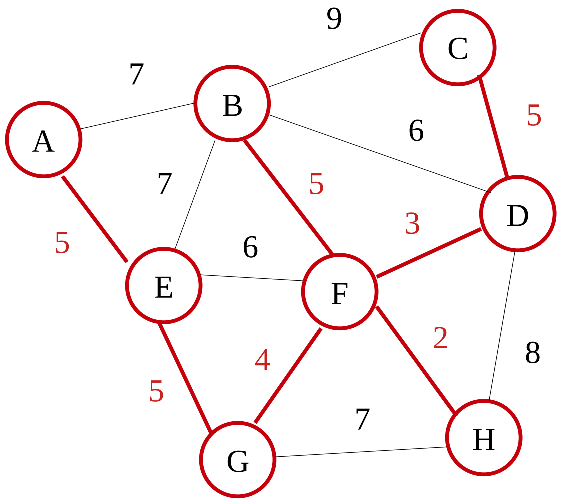
Jahongir Hayitov – CS-01

Problem-Set 12, Theoretical Part

1. Run Prim-Jarn´ık algorithm [Cormen, Section 21.2] on the following graph, starting at vertex C. Assuming that the algorithm is using Fibonacci heap implementation of a priority queue, show the state of the Fibonacci heap after each iteration of the algorithm (i.e. after adding each new vertex to the MST). The graph contains 8 vertices, which means that your solution must provide 8 states of the Fibonacci heap. No justification required.

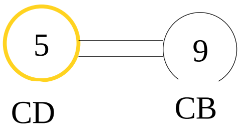


Solution: MST

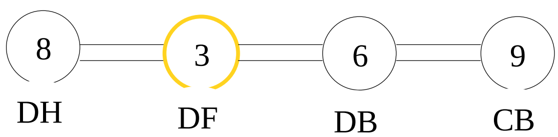


{C-D: 5, D-F: 3, F-H: 2, F-G: 4, F-B: 5, G-E: 5, E-A: 5}

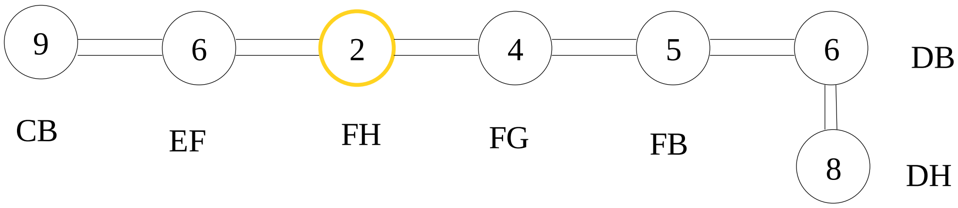
1)



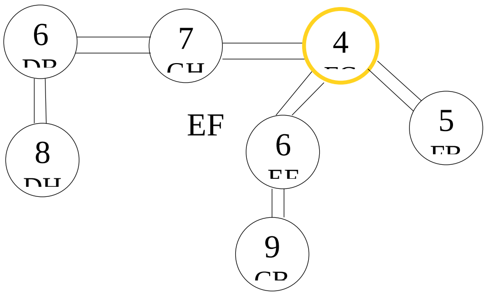
2)



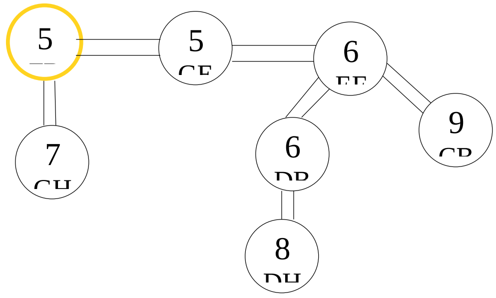
3)



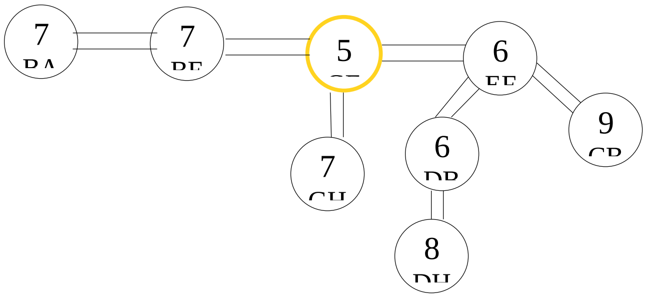
4)



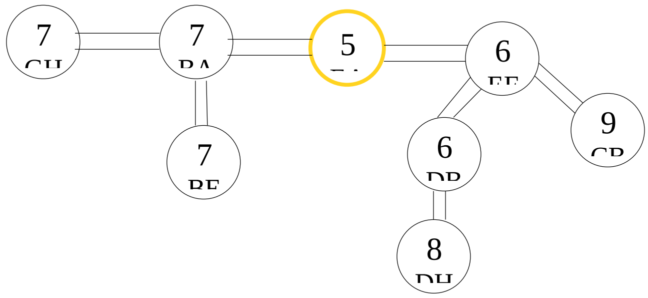
5)



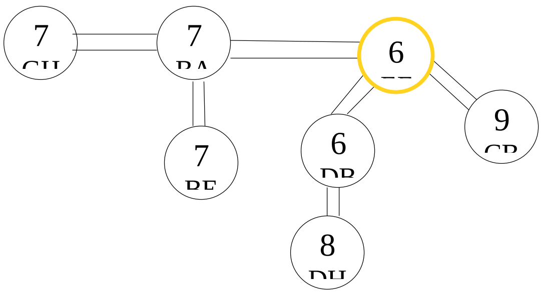
6)



7)



8)



2) Suppose that all edge weights in a graph are integers in the range from 1 to |V |. How fast can you make Prim-Jarn´ık algorithm run? What if the edge weights are integers in the range from 1 to W for some constant W? Justify your answer in at most two paragraphs.

<u>Solution:</u>

V = number of Vertices, E= number of Edges

O(V(initialization)) + O(V * T(extract-min)) + O(E * T(decrease-key)).

If the edges are in the range 1, . . . , |V| by using efficient priority queue we can make DECREASE-KEY and EXTRACT- MIN running O(log(log(V)), and a total running time of O(E log(log V))

If the edges are in the range from 1 to W, we can use array [1...W+1] where the array[i] holds a doubly linked list of the edges with weight i. The array[W+1] contains ∞. EXTRACT-MIN runs in O(W) = O(1) time since we can scan it in constant time. DECREASE-KEY runs in O(1) time as well and it additionally swap the elements.

I got extra information from this reference: https://github.com/gzc/CLRS/blob/master/C23-Minimum-Spanning-Trees/23.2.md