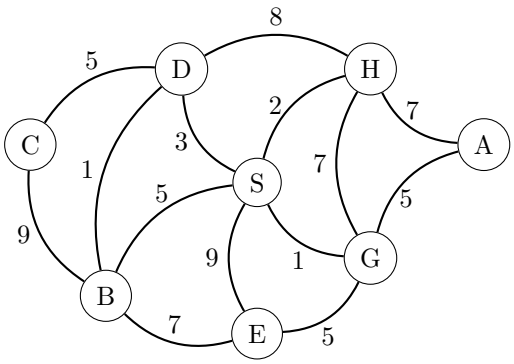
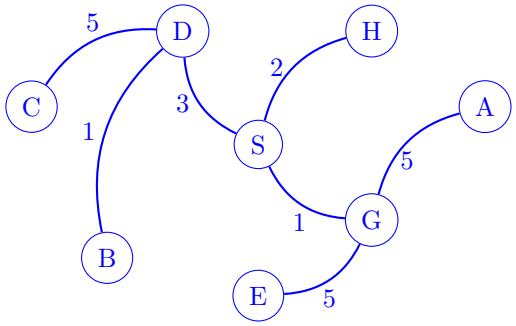
Week 10. Problem set (solutions by Danil Elgin)

1. Run Prim-Jarník algorithm [CLRS, §21.2] on the following graph $\mathcal{G}$, starting at vertex S, and answer the questions below.



(a) Draw a minimum spanning tree of the graph $\mathcal{G}$.
Answer: *(add edges in the template below)*



(b) Is the minimum spaning tree for graph $\mathcal{G}$ unique?
Answer: *Yes*

(c) Assuming that the algorithm is using Binary Heap implementation [CLRS, §6] of a priority queue, show the state of the priority queue 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 Binary heap. Each heap state must be represented as an array.

For example, a binary min-heap containing key-value pairs $\langle 3, a\rangle, \langle 2, b\rangle, \langle 1, c\rangle, \langle 4, d\rangle$ may be represented as follows:

$1, c$	$3, a$	$2, b$	$4, d$

Answer:

1)	$1, G$	$2, H$	$3, D$	$5, B$	$9, E$
2)	$2, H$	$5, B$	$3, D$	$5, E$	$5, A$
3)	$3, D$	$5, B$	$5, A$	$5, E$	
4)	$1, B$	$5, E$	$5, A$	$5, C$	
5)	$5, C$	$5, E$	$5, A$		
6)	$5, A$	$5, E$			
7)	$5, E$				
8)	$\emptyset$				

2. Suppose that T is the minimum spaning tree for graph $\mathcal{G}$. How difficult is it to update the minimum spanning tree after adding a new vertex and its incident edges to $\mathcal{G}$.

   (a) Describe your algorithm (write pseudocode).

   Answer:
   The pseudo-code for Kruskal's algorithm is in [CLRS, §21.2]

---

procedure MST_ADD_VERTEX(T, v)
    T.addVertex(v)
    for each $u \in v.adj$ **do**
        T.addEdge(u)
    end for
    Kruskal_algorithm(T)
end procedure

   (b) Provide the asymptotic time complexity for your algorithm.

   Answer: $O(|V| * log(|V|))$

   (c) Justify the asymptotic time complexity (1–3 sentences).

   Justification: *We can add a new vertex in O(1). Also, in the worst case, our vertex will be connected to all the others, and it will take $O(|V|)$ to add all the necessary edges. Further, since the graph was a tree, the number of edges in it was $|V|-1$, then if we apply Kruskal's algorithm (it is assumed that sorting is performed for $(2|V|-1) * log(|2|V|-1))$ it will be executed on the new graph in $O(|V|+1+(2|V|-1)*log(2|V|-1)+(|V|-1+|V|)*\alpha(|V|+1)$ where ($|V|$ is the number of vertices before adding a new one), and we can convert all procedures to $O(|V| * \alpha(|V|) + |V| * log(|V|))$. However, since the logarithm function dominates and majorizes the Ackerman function, we can convert to $O(|V| * log(|V|))$*

3. Suppose that all edge weights in a graph are *real numbers* uniformly distributed in the range from 0 to k.

   (a) Which of the MST algorithms, Kruskal's or Prim's (possibly with minor modifications) can run faster on such a graph?

   Answer: *Kruskal's algorithms*

   (b) Which modification is required to make the algorithm run faster? Justify your answer (2–3 sentences).

   Answer: *Apply "Bucket sort" when sorting edges in Kruskal's algorithm*

   Justification: *Since the weights of the edges are evenly distributed, this means that each bucket will contain approximately the same number of items. This will allow us to do the sorting for $|E|$, which in the end will give a time complexity less than in the Prime's algorithm.*

   (c) What is the *average* case time complexity of the modified algorithm? Justify your answer (1—2 sentences).

   Answer: *$O(|V| + E * \alpha(|V|)$*

   Justification: *On average, a "Bucket sort" takes O(n), where n is the number of elements, in our case the number of weights, i.e. edges - $|E|$. We also need to go through the vertices to create sets from the same vertex elements, for later unification. If we use union with pass compression, then each operation will be performed in $\alpha(|V|)$ (α - is the inverse Ackermann function), so we need to go through all edges at most - $|E|$. This means that the total time complexity will be $O(|V| + |E| + |E| * \alpha(|V|)$, that is the same as $O(|V| + |E| * \alpha(|V|))$.*

References

[CLRS] Cormen, T.H., Leiserson, C.E., Rivest, R.L. and Stein, C., 2022. *Introduction to algorithms, Fourth Edition*. MIT press.

[GTG] M. T. Goodrich, R. Tamassia, and M. H. Goldwasser. *Data Structures and Algorithms in Java*. WILEY 2014.