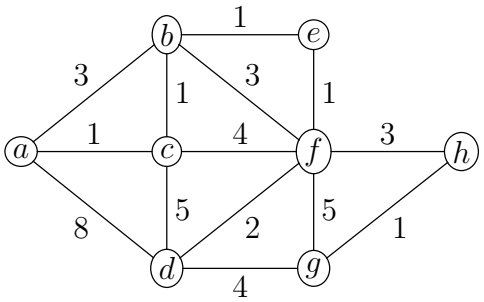
Solutions for Exercise Sheet 15

Handout: December 21st — Deadline: December 28th, 4pm

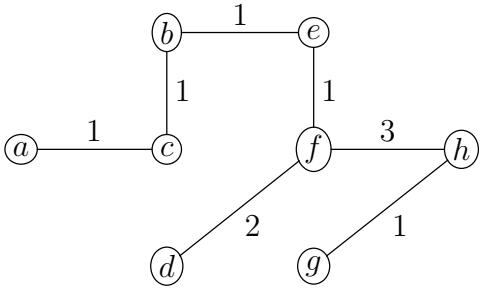
Question 15.1 (0.25 marks)

Compute the minimum spanning tree of the following weighted graph both with Prim's and Kruskal's algorithm. List the edges in the order considered, draw the tree and calculate its weight.



Solution.

With Prim's algorithm, starting from source a, the edges added are (a, c), (c, b), (b, e), (e, f), (f, d), (f, h), (h, g) in that order. The MST computed with this algorithm is

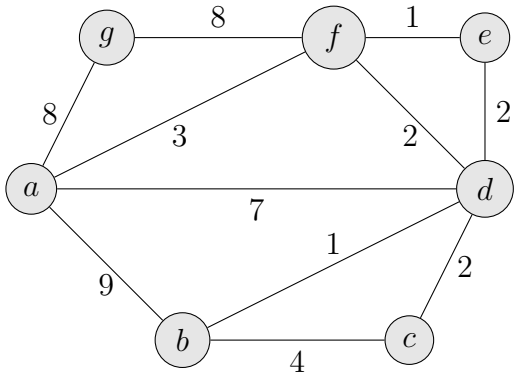


Its weight is 10.

Kruskal's algorithm leads to the same solution: add all edges of weight 1 (they are all feasible). Then add edge (d, f) of weight 2, which is feasible. There are 3 edges of weight 3: (a, b) and (b, f) are infeasible as they run within the same tree, so we only add (f, h) and get the above MST.

Question 15.2 (0.25 marks)

Execute Dijkstra's algorithm on the following weighted graph to find a shortest path from vertex a to c. Show for each iteration of the while loop which vertex is added to the set S and how the distance estimates of adjacent vertices are being refined.



Solution: The following table shows the progression of Dijkstra's algorithm.

	S	$a.d$	$b.d$	$c.d$	$d.d$	$e.d$	$f.d$	$g.d$
initial state ($a.\pi = \text{NIL}$)	$\{a\}$	0	9	∞	7	∞	3	8
adding f ($f.\pi = a$)	$\{a, f\}$	0	9	∞	5	4	3	8
adding e ($e.\pi = f$)	$\{a, e, f\}$	0	9	∞	5	4	3	8
adding d ($d.\pi = f$)	$\{a, d, e, f\}$	0	6	7	5	4	3	8
adding b ($b.\pi = d$)	$\{a, b, d, e, f\}$	0	6	7	5	4	3	8
adding c ($c.\pi = d$)	$\{a, b, c, d, e, f\}$	0	6	7	5	4	3	8
adding g ($g.\pi = a$)	$\{a, b, c, d, e, f, g\}$	0	6	7	5	4	3	8

The shortest path from a to c is via a, f, d, c and has length 7.

Question 15.3 (0.5 marks)

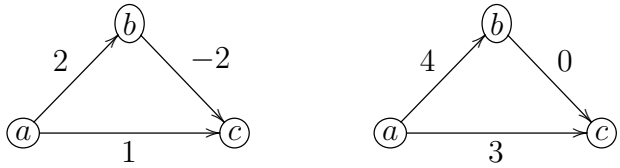
A precondition for Dijkstra's algorithm is that all edges of the directed graph under consideration have non-negative weight.

Somebody on the internet claims that the algorithm works for graphs with negative edge weights as well: just add an appropriate constant c to each edge weight to make all weights positive, then run Dijksta's algorithm, and finally remove the constants from the shortest paths computed.

Give a directed acyclic graph as a counterexample to falsify this claim. Explain in your own words what goes wrong.

Solution.

The problem is that the weight of a path of length n increases by $c \cdot n$ when c is added to the weight of each edge. Thus long paths gain significantly more weight than short ones, and it can happen that shortest paths are transformed into paths that are no longer shortest ones. In the following counterexample, weight 2 is added to all edges of the left-hand graph.



The shortest path from a to c in the left-hand graph is a, b, c, but the shortest path from a to c in the right-hand graph is a, c. So then running Dijkstra's algorithm on the right-hand graph it would return a, c as shortest path, which is an incorrect solution for the original graph on the left-hand side.

Question 15.4 (3 marks)

Implement DIJKSTRA(G, w, s) for a given directed graph $G(V, E)$ with weight vector w. The algorithm should return the shortest paths from the source s to all other nodes. The input will be:

- first line: N M (the number of vertices and edges).
- M lines each containing a pair $v_i v_j$ meaning there is an edge $v_i \rightarrow v_j$ followed by the weight of the edge.
- the final two lines contain one node each v_y and v_z to be given input to PRINT-PATH(G, s, v) which will print the shortest path from the source to the respective node.

You have to first implement a MIN-PRIORITY-QUEUE and the respective procedures (EXTRACT-MIN(Q), DECREASE-KEY(Q, x, k)) - You should NOT USE libraries for this.

Remember that you need a "handle" to keep track of where each node is in the heap - see Lecture Slides Week 8 to recap Priority Queues.

Then you have to build the adjacency list representing the graph.

Finally you can implement Dijkstra.