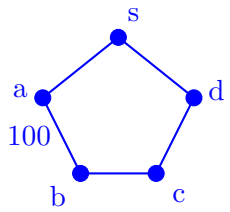
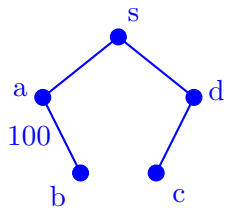
1 Q1 [5 marks]

Give an example of a connected, weighted, undirected graph G and a start vertex s such that neither the DFS spanning tree nor the BFS spanning tree of G from s is a minimum weight spanning tree of G, regardless of how the adjacency lists are ordered...

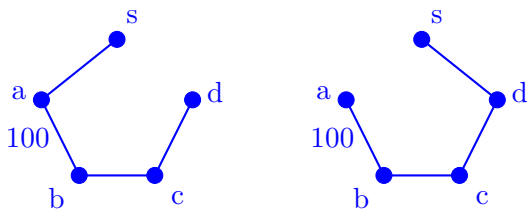
Solution. Consider the graph



with all edge weights 1 except for the edge (a, b) which has weight 100. The BFS tree starting at s yields the spanning tree



with weight 103 and the two possible spanning trees obtained from DFS started at s are



which both also have weight 103. The spanning tree obtained by removing edge (a, b), by contrast, has total weight 4.

2 Q2 [7 marks]

Solution. Solution idea is based on simple properties of directed graphs:
1. sink vertex will always have no elements in corresponding adjacency list;
2. source vertex in the original graph G will become a sink vertex in transposed graph G^T;
3. graph G^T can be constructed in time $\Theta(|V| + |E|)$ given adjacency list representation of G;
4. the sink vertex can be found in time $O(|V|)$...

3 Q3 [6 marks]

Prove that if the BFS and DFS spanning trees of a connected undirected graph G from the same start vertex s are equal to each other, then G is a tree.

Solution. Assume for contradiction that G is a c connected graph that is not a tree and that there is a vertex s such that the BFS tree T and the DFS tree T' are equal to each other. Since G is not a tree, there must be some cycle in G and some non-tree edge (u, v) connecting u to one of its ancestors v in the DFS tree T'. But then u cannot be a descendant of v in the BFS tree T unless it is its child (in which case (u, v) is a tree edge) so $T \neq T'$.

4 Q4 [8 marks]

The *distance* between two vertices in an undirected graph is the length of the shortest path between them. The *diameter* of an undirected graph is the maximum distance between vertices in the graph...

Solution. Idea of the solution is based on a simple observations:
1. if u and v are two vertices in given graph with the maximum shortest distance between them, than `BFS_Visit` that starts at u will find this distance;
2. we can starts `BFS_Visit` from every vertex of the graph and find maximum of computed shortest distances.

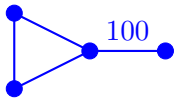
Running time is a cost of $|V|$ runs of `BFS_Visit` ($\Theta(|V| + |E|)$ each): $\Theta(|V|(|V| + |E|))$.

5 Q5 [14 marks]

Prove or disprove each of the following statements, where in each case $G = (V, E)$ is a connected undirected weighted graph with n vertices and m edges.

(a) If G has $m \geq n$ edges and a unique heaviest edge e, then e is not part of any minimum spanning tree of G.

Solution. **False.** Consider the graph



that has 4 vertices and 4 edges, three of which of weight 1 and one of which has weight 100. The unique heaviest edge of weight 100 must be included in any spanning tree of the graph, so it is also in the minimum spanning tree.

(b) If G has $m \geq n$ edges and a unique lightest edge e, then e is part of every minimum spanning tree of G.

Solution. **True.** The proof is very similar to that of the Cut Property in MST Lecture notes. Assume for contradiction that there is a MST T of G that does not include e. Then if we add the edge e to T, we create a cycle in the graph and we can remove any other edge $e' \neq e$ in the cycle to obtain another spanning tree T' of G. But the resulting tree T' has smaller weight than T, contradicting the fact that T was a MST of G.

(c) If e is a maximal weight edge of a cycle of G, then there is a minimum spanning tree of G that does not include e.

Solution. **True.** Let $T = (V, E')$ be a MST of G that includes e. Let $X = E' \setminus \{e\}$ be the set of edges in T *except* for e. Then X is part of a MST of G and there must be a set $S \subseteq V$ (that includes exactly one of the two end vertices of e) for which X has no edges that crosses the cut $(S, V \setminus S)$. The edge e crosses the cut, but since e is part of a cycle, there must be another edge $e' \neq e$ in the cycle that in the minimum-weight edge crossing the cut. (The edge e cannot be the unique minimal weight edge crossing the cut since it has maximal weight among the edges in the cycle.) By the Cut Property Lemma, the tree T' with edges $X \cup \{e'\}$ is a MST of G that does not contain the edge e.

(d) Prim's algorithm returns a minimum spanning tree of G even when the edge weights can be either positive or negative.

Solution. **True.** The analysis of Prim's algorithm in the lecture notes for MST does not assume anything about the edge weights, so it works with both positive and negative weights.

If we want to be extra careful and assume that this analysis only works for non-negative weights, then for any graph G with some negative edge weights, let $-\gamma$ denote the minimum weight of any edge of G. Let $G' = (V, E)$ be the weighted graph with weight function $w' : E \to \mathbb{R}$ defined by $w'(e) = w(e) + \gamma$. Then the graph G' has non-negative edge weights and Prim's algorithm computes a MST T' of G'. The same tree is also a MST of G since the for every tree T, $\text{cost}_w(T) = \text{cost}_{w'}(T) - (n-1)\gamma$.

6 Q6 [10 marks]

Given a directed graph with positive edge lengths and a specified vertex v in the graph, the "all-pairs v-constrained shortest path problem" is the problem of computing for each pair of vertices i and j the shortest path from i to j that goes through the vertex v. If no such path exists, the answer is ∞. Describe an algorithm ...

Solution. Let P be the shortest path from i to j that goes through vertex v. Then P consists of the shortest path from i to v and from shortest path from v to j. To prove this, assume it is not true, for example because there exists a shorter path from i to v than the one used in P. But then we could improve P by replacing the original path from i to v with the new shorter path.

Therefore we need to compute the shortest paths from every vertex to v and shortest paths from v to every other vertex. The second task is easy – we can use Dijkstra's algorithm. For the

first task we can simply use Dijkstra's algorithm on graph G^T (recall that G^T is G with directions of all edges reversed). Each path from i to v in G corresponds to a path from v to i in G^T.

Now assume that $d[i]$ is the length of the shortest path from v to i and $d^T[i]$ is the length of the shortest path from i to v. Then $M[i,j] = d^T[i] + d[j], i = 1, \ldots, n, j = 1, \ldots, n$.

We use Dijkstra's algorithm twice. If we use the simple implementation, it takes $\Theta(n^2)$ time. Then we need to fill in all n^2 entries of table M, each in $\Theta(1)$ time. The overall computation therefore takes $\Theta(n^2)$ time.

Since the table has n^2 entries that need to be filled in, no algorithm can possibly work faster than $\Omega(n^2)$. Thus the complexity of this problem is also $\Theta(n^2)$.