Algorithms and Data Structures II.
Test2 (example exercises)

23 Minimum Spanning Trees

23.1 Kruskal's algorithm

23.1-1 As you have seen in the classroom, illustrate the run of Kruskal's algorithm on the following graph[1]. Finally, draw the MST calculated by the algorithm.

1 – 2, 2; 5, 1.	2 – 1, 2; 6, 0.
3 – 4, 4; 6, 1; 7, 1.	4 – 3, 4; 7, 3; 8, 2.
5 – 1, 1; 6, 1.	6 – 2, 0; 3, 1; 5, 1; 7, 2.
7 – 3, 1; 4, 3; 6, 2; 8, 1.	8 – 4, 2; 7, 1.

23.1-2 Given class Edge{ $+u, v : \mathbb{N}_+$; $+w : \mathbb{R}$ }. The objects of this class can represent edges of weighted graphs where $w(u, v) = w$. Arrays G and T have the element type Edge. Array G represents a weighted undirected connected graph. It consists of the edges of the graph. It is non-decreasing according to the weights of the edges. It has m elements. Array T has $n1$ elements where $n1 = n - 1$ and n is the number of vertices of the graph.

Write the structogram of the procedure Kruskal(G, T) which calculates an MST of graph G and puts the edges of this MST into the array T in $O(m * \log n)$ time.

23.2 Prim's algorithm

23.2-1 As you have seen in the classroom, illustrate the run of Prim's algorithm on the following graph. Draw the minimum spanning tree represented by the final π and d values.

1 – 2, 2; 5, 1.	2 – 1, 2; 6, 0.
3 – 4, 4; 6, 1; 7, 1.	4 – 3, 4; 7, 3; 8, 2.
5 – 1, 1; 6, 1.	6 – 2, 0; 3, 1; 5, 1; 7, 2.
7 – 3, 1; 4, 3; 6, 2; 8, 1.	8 – 4, 2; 7, 1.

23.2-2a Suppose that we represent the weighted undirected connected graph $G = (V, E)$ as an adjacency matrix. Give a simple implementation of Prim's algorithm for this case that runs in $O(|V|^2)$ time.

[1] u – $v_1, w_1; \dots v_n, w_n$. means that the graph has the undirected edges $(u, v_1), \dots (u, v_n)$ with weights $w_1, \dots w_n$.

23.2-2b Suppose that we represent the weighted undirected connected graph $G = (V, E)$ with adjacency lists. Give a simple implementation of Prim's algorithm for this case that runs in $O(|V|^2)$ time.

23.2-2c* Suppose that we represent the graph $G = (V, E)$ with adjacency lists. Give a sophisticated implementation of Prim's algorithm for this case that runs in $O(|E| * \log |V|)$ time.

Hint: Use a binary minimum heap to represent the priority queue of the vertexes (organized according to the d values of the vertexes). When we decrease $d(v)$ for a vertex v, it must be compared with its parent in the heap (concerning their d attributes), and they possibly must be swapped, recursively. Therefore, we need an indexing array to know the place of each vertex in the heap.

24 Single-Source Shortest Paths

24.1 Queue-based Bellman-Ford algorithm

(The **Queue-based Bellman-Ford algorithm** is also known as **Tarjan's breadth-first scanning algorithm**, and **Shortest Path Faster Algorithm (SPFA)**.)

24.1-1 As you have seen in the classroom, illustrate the run of the *Queue-based Bellman-Ford* algorithm on the directed graph below[2], using vertex z as the source. Draw the shortest-paths tree represented by the final π and d values.

Now, change the weight of edge (z, x) to 4 and run the algorithm again, using s as the source.

$s \to t, 6; y, 7.$ $t \to x, 5; y, 8; z, -4.$
$x \to t, -2.$ $y \to x, -3; z, 9.$
$z \to s, 2; x, 7.$

24.1-2a Suppose that we represent the graph $G = (V, E)$ as an adjacency matrix. Give a simple implementation of the *Queue-based Bellman-Ford* algorithm for this case that runs in $O(|V|^3)$ time.

24.1-2b Suppose that we represent the graph $G = (V, E)$ with adjacency lists. Give a simple implementation of the *Queue-based Bellman-Ford* algorithm for this case that runs in $O(|V| * |E|)$ time.

24.2 Single-source shortest paths in directed acyclic graphs (DAGs)

24.2-1 As you have seen in the classroom, illustrate the run of the *DAG single source shortest paths* algorithm on the directed graph below, using vertex s as the source.

$r \to s, 5; t, 3.$ $s \to t, 2; x, 6.$ $t \to x, 7; y, 4; z, 2.$
$x \to y, -1; z, 1.$ $y \to z, -2.$ $z.$

24.2-2 Replace edge "$y \to z, -2$" with the edge "$y \to s, -2$" in the graph above. As you have seen in the classroom, illustrate the run of the *DAG single source shortest paths* algorithm on the resulting graph, using vertex r as the source.

---

[2] $u \to v_1, w_1; \ldots v_n, w_n.$ means that the graph has the directed edges $(u, v_1), \ldots (u, v_n)$ with weights $w_1, \ldots w_n$.

24.1-3a Suppose that we represent the graph $G = (V, E)$ as an adjacency matrix. Give a simple implementation of the *DAG single source shortest paths* algorithm for this case that runs in $O(|V|^2)$ time.

24.1-3b Suppose that we represent the graph $G = (V, E)$ with adjacency lists. Give a simple implementation of the *DAG single source shortest paths* algorithm for this case that runs in $O(|V| + |E|)$ time.

24.3 Dijkstra's algorithm

24.3-1 As you have seen in the classroom, illustrate the run of Dijkstra's algorithm on the directed graph below, first using vertex s as the source and then using vertex z as the source. Draw the shortest-paths tree represented by the final π and d values.

$s \to t, 3; y, 6.$ $t \to x, 8; y, 2.$ $x \to z, 2.$
$y \to t, 1; x, 4; z, 6.$ $z \to s, 3; x, 7.$

24.3-2 Give a simple example of a directed graph with some positive-weight edges and a negative-weight edge for which Dijkstra's algorithm produces an inconsistent answer.

24.3.3a Suppose that we represent the graph $G = (V, E)$ as an adjacency matrix. Give a simple implementation of Dijkstra's algorithm for this case that runs in $O(|V|^2)$ time.

24.3.3b Suppose that we represent the graph $G = (V, E)$ with adjacency lists. Give a simple implementation of Dijkstra's algorithm for this case that runs in $O(|V|^2)$ time.

24.3.3c* Suppose that we represent the graph $G = (V, E)$ with adjacency lists. Give a sophisticated implementation of Dijkstra's algorithm for this case that runs in $O((|V| + |E|) * \log |V|)$ time.

Hint: Use a binary minimum heap to represent the priority queue of the vertexes (organized according to the d values of the vertexes). When we decrease $d(v)$ for a vertex v, it must be compared with its parent in the heap (concerning their d attributes), and they possibly must be swapped, recursively. Therefore, we need an indexing array to know the place of each vertex in the heap.

25 All-Pairs Shortest Paths

25.2 The Floyd-Warshall algorithm

25.2-1 As you have seen in the classroom, illustrate the run of the Floyd-Warshall algorithm on the weighted graph below. Show the matrix pairs $(D^{(0)}, \Pi^{(0)}), \ldots, (D^{(4)}, \Pi^{(4)})$. Finally, draw the shortest path trees represented by the rows of the last pair of matrices.

	1	2	3	4
1	0	5	3	1
2	5	0	1	∞
3	3	1	0	1
4	1	∞	1	0

25.2.2 As you have seen in the classroom, illustrate the run of Warshall's transitive-closure algorithm on the unweighted, directed graph below.
Show the matrices $T^{(0)}, \ldots, T^{(4)}$.

	1	2	3	4
1	0	1	0	0
2	0	0	1	0
3	0	0	0	1
4	1	0	0	0