# Exercise IX, Algorithms 2024-2025

These exercises are for your own benefit. Feel free to collaborate and share your answers with other students. There are many problems on this set, solve as many as you can and ask for help if you get stuck for too long. Problems marked * are more difficult but also more fun :).

## More on maxflows

**1** ($\star$, Radar evasion) The People's Republic of Isolatia is a perfectly rectangular country. It shares its entire west border with the Republic of Westilia and its entire east border with the Republic of Eastilia. Westilia would like to send aircraft to provide emergency aid to Eastilia. The only way of reaching Eastilia is by flying through Isolatia's airspace (dangerous uncharted territory borders Isolatia on the south and north!), but Isolatia prohibits other countries from flying in its airspace. Therefore, Isolatia has deployed radar installations throughout the country. Each of the $n$ radar sites is specified by its coordinates and the radius of its range. Westilia plans to send saboteurs to Isolatia to disable some radar stations in order to allow it to fly planes from the west border to the east border of Isolatia without being detected by radar.

A flight path is a continuous curve entirely contained in the airspace of Isolatia that connects a point on the west border to a point on the east border. You can use the fact that there exists a flight path if and only if there is no sequence of radars $r_1, r_2, \ldots, r_k$ such that the range of $r_1$ overlaps the north border, the range of $r_i$ overlaps the range of $r_{i+1}$ for all $i = 1 \ldots k-1$ and the range of $r_k$ overlaps the south border.

Give an efficient algorithm for determining the smallest number of radar sites to be disabled to establish a flight path from Westilia to Eastilia that is outside the range of any radar station.

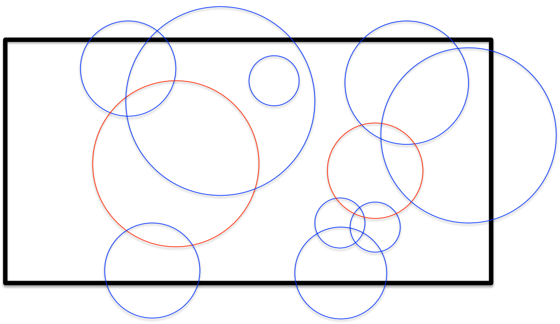
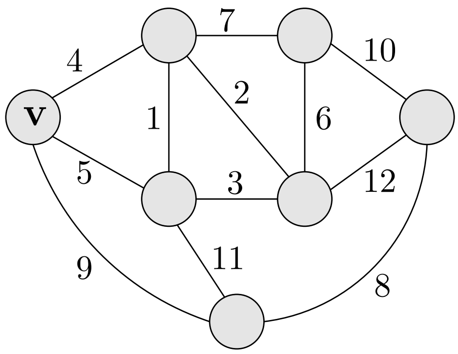


**Figure 1.** An example of radars in a rectangular region. Removal of the two radars indicated in red provides a valid flight path between the east and west boundaries.

## Minimum spanning trees

**2** (Exam question 2012 worth 20 pts) Consider the following minimum spanning tree instance, i.e., an undirected connected graph with weights on the edges.




**2a** (5 pts) Write the weights of the edges of the minimum spanning tree in the order they are added by Prim's algorithm starting from vertex $v$.

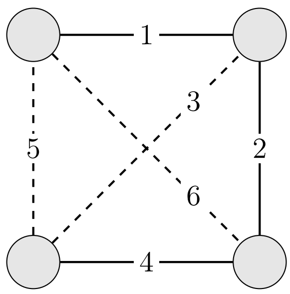
**2b** (5 pts) Write the weights of the edges of the minimum spanning tree in the order they are added by Kruskal's algorithm.

**2c** (*,10 pts) A bottleneck edge is an edge of highest weight in a spanning tree. A spanning tree is a minimum bottleneck spanning tree if the graph does not contain a spanning tree with a smaller bottleneck edge weight. *Show that a minimum spanning tree is also a minimum bottleneck spanning tree.*

**3** *(*, Exam question 2013 worth 10 pts)* Consider three undirected edge-weighted connected graphs $G_1 = (V, E_1), G_2 = (V, E_2)$, and $H = (V, E_1 \cup E_2)$ with non-negative weights $w : E_1 \cup E_2 \to \mathbb{R}_+$ on the edges. Note that they are all graphs on the same vertex set but their edges differ: $G_1$ has only the edges in $E_1$, $G_2$ has only the edges in $E_2$, and $H$ has all the edges $(E_1 \cup E_2)$.

Let $T, T_1, T_2$ be minimum spanning trees of $H, G_1$, and $G_2$, respectively. Assuming that the weights of the edges are unique, i.e., no two edges have the same weight, *prove that* $T \subseteq T_1 \cup T_2$.

For an example of the statement see the figure below. The solid edges are $E_1$ and the dashed edges are $E_2$. Note that the minimum spanning tree of $G_1$ is $T_1 = \{1, 2, 4\}$, the minimum spanning tree of $G_2$ is $T_2 = \{3, 5, 6\}$, and the minimum spanning tree of $H$ is $T = \{1, 2, 3\}$. We have thus that $T \subseteq T_1 \cup T_2$ in this case. You should prove that it holds in general.



**4** (Exercise 23.2-1) Let $(u, v)$ be a minimum-weight edge in a connected graph $G$. Show that $(u, v)$ belongs to some minimum spanning tree of $G$.

**5** (Exercise 23.2-8) Professor Borden proposes a new divide-and-conquer algorithm for computing minimum spanning trees, which goes as follows. Given a graph $G = (V, E)$, partition the set $V$ of vertices $V_1$ and $V_2$ such that $|V_1|$ and $|V_2|$ differ by at most 1. Let $E_1$ be the set of edges that are incident only on vertices in $V_1$, and let $E_2$ be the set of edges that are incident only on vertices $V_2$. Recursively solve a minimum-spanning-tree problem on each of the two subgraphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$. Finally, select the minimum-weight edge in $E$ that crosses the cut $(V_1, V_2)$, and use this edge to unite the resulting two minimum spanning trees into a single spanning tree.

Either argue that the algorithm correctly computes a minimum spanning tree of $G$, or provide an example for which the algorithm fails.

**6** (part of Problem 23-3) A **bottleneck spanning tree** $T$ of an undirected graph $G$ is a spanning tree of $G$ whose largest edge weight is minimum over all spanning trees of $G$. We say that the value of the bottleneck spanning tree is the weight of the maximum-weight edge in $T$.

Give a linear-time algorithm that given a graph $G$ and an integer $b$, determines whether the value of the bottleneck spanning tree is at most $b$.