1. (a) Give an algorithm for finding the smallest element in a max-heap. Assume the usual array implementation for the heap. The algorithm should examine as few elements in the heap as possible.

(b) Give an algorithm for printing all elements bigger than \( x \) from a max-heap. You are not allowed to change the contents of the heap. The time complexity of the algorithm should be linear in the number of elements printed.

2. How could an operation \texttt{heap-inc-all-keys}(k) that increases every key with the value \( k \) be defined such that it would only take time \( O(1) \) and the time complexity of the rest of the operations would stay unchanged?

3. Give an iterative (i.e. non-recursive) version of heapify in a min-heap. (Note: min-heap, not max-heap)

4. In a 3-ary heap, the internal nodes have three children instead of two as in the binary heap (as usual, the last node can have less children).

   How do you generalise the array representation of a binary heap to the case of a 3-ary heap? Write pseudocode for the max-3-ary-heap operations \texttt{3-heapify}, \texttt{3-heap-insert} and \texttt{3-heap-del-max}. What are the time complexities of the operations?

5. Consider an \( m \times n \) matrix \( A \), where each element is either a natural number or \( \infty \). We say that \( A \) is a Young tableau if each row is in increasing order from left to right and each column is in increasing order from top to bottom. The value \( \infty \) is interpreted as larger than any natural number. For example, the natural numbers 2, 5, 7, 11, 15, 16, 21 and 27 could be placed into a 3 \( \times \) 4 Young tableau as follows:

   \[
   \begin{array}{cccc}
   2 & 5 & 11 & 27 \\
   7 & 15 & 16 & \infty \\
   21 & \infty & \infty & \infty \\
   \end{array}
   \]

   Give an algorithm that removes the smallest element from a Young tableau, replaces it with \( \infty \) and then fixes the table so that it again becomes a Young tableau. Thus, one possible result starting from the previous tableau is the following:

   \[
   \begin{array}{cccc}
   5 & 11 & 16 & 27 \\
   7 & 15 & \infty & \infty \\
   21 & \infty & \infty & \infty \\
   \end{array}
   \]

   Your algorithm should run in time \( O(m + n) \).  

   \textit{Hint}: Think about heapify for min-heaps.

6. Consider again Young tableaus. We are given as input an \( m \times n \) Young tableau \( A \) and a natural number \( k \). The problem is to decide whether \( k \) is in the tableau \( A \) or not. Give an algorithm that runs in time \( O(m + n) \).
7. In the following problems the input of the algorithm is an array \( T \) that contains \( n \) integer values. Tell in which way each problem can be solved in time \( O(n \log n) \). You do not need to give any detailed algorithm in pseudocode; a description of the idea is enough.

(a) Does some value occur multiple times in the array?
(b) Which value occurs the most in the array?
(c) What is the smallest difference between two values in the array?

8. An algorithm gets as input an array \( T \) which contains \( n \) integers and an integer \( k \).

(a) Give an solution algorithm working in time \( O(n \log n) \), which checks if the array has two values, which produce sum \( k \). The element \( T[i] \) can be used only once. Each element \( T[i] \) can be used only once.
(b) Give an algorithm working in time \( O(n^2 \log n) \) which checks if the array has four values, which produce sum \( k \). Each element \( T[i] \) can be used only once.

9. We order an array of \( n \) numbers in increasing order using insertion sort. What is the running time of the algorithm in the following cases?

(a) The array contains \( n \) different numbers and they are given in increasing order.
(b) The array contains \( n \) different numbers and they are given in descending order.
(c) The array consists of the same number \( n \) times.

10. What are the answers if we change the sorting method in the previous exercise to heapsort?

11. What are the answers if we change the sorting method in the previous exercise to mergesort?

12. What are the answers if we change the sorting method in the previous exercise to quicksort using the partition function used in the lectures (the pivot is the first number in the array, not last as in Cormen p. 146)?

13. You can sort an array in time \( O(n \log n) \). But how fast can you shuffle an array? Define an efficient algorithm for shuffling an array. What is the time and space complexity of your algorithm? You can assume, that you have a function \( \text{random}(i,j) \), that returns an integer from the interval \([i, j]\) in constant time. Does your algorithm shuffle the array so that all the orders of the values are equally probable?

14. Give an algorithm that checks if a graph is bipartite. A graph is bipartite if its vertices can be divided into two disjoint sets \( U \) and \( V \) such that every edge connects a node in \( U \) to one in \( V \).

15. A breadth-first search finds the shortest path between two nodes, but it can take much space, as the space complexity is \( O(n) \), where \( n \) is the number of nodes in the graph. Give an algorithm for finding the shortest path, with space complexity only \( O(p) \), where \( p \) is the length of the shortest path.
16. You are given as input a directed non-weighted graph \( G = (V, E) \). In addition, a
given node is the starting node \( s \) and a given node the target node \( t \) and each edge
is coloured either red or blue. The task is to find a path from \( s \) to \( t \), so that all red
edges are before the blue edges: in other words, the path consists of a first part of
red edges (if any) and a second part of blue edges (if any). If there are several such
paths, the shortest one should be chosen (shortest in number of edges on the path).
Give an efficient algorithm for the problem and analyse its time complexity.
A slower algorithm suffices, but there is a solution to the problem with time complexi-
ty \( O(|V| + |E|) \).

17. We have a connected graph \( G = (V, E) \) and a vertex \( u \in V \). Prove that if we compute
a depth-first search tree rooted at \( u \) and a breadth-first search tree rooted and
these two trees are the same, then the graph \( G \) is exactly this tree, i.e., \( G \) does not
include any edges that do not belong to the search tree obtained.

18. Make a program, which counts how many rooms there are in a given house. As input
the algorithm gets a two-dimensionl character table representing the house. # means
the wall and . means the floor.
The input could be for example:

```
##########
#.###.#.###.
#.###.#.###.
#####.###.#.
#.###.#.###.
#.#####..#.####
#.###.#.###.
```

There are 4 rooms in the example above.

19. An Euler circuit for an undirected graph is a path which starts and ends at the same
vertex and uses each edge exactly once. A connected, undirected graph \( G \) has an
Euler circuit if and only if every vertex is of even degree. Give an \( O(e) \) algorithm to
find an Euler circuit in a graph with \( e \) edges, provided one exists.

20. A Hamiltonian path in an undirected graph is a simple path that visits each vertex
of the graph exactly once. Write an algorithm which checks whether the graph given
as input contains at least one hamiltonian path. What is the time complexity of your
algorithm?

21. Do a topological sort for the graph below using the algorithm given in Cormen.
Assume that nodes are in alphabetical order in the adjacency lists.
22. The program gets a set of dependencies as input. The input could be for example the dependencies between course passing order:

- ohpe ohja
- ohja tira
- tira lama
- tira tiralabra
- ohja javalabra
- javalabra tiralabra
- ohpe ohma
- tiralabra ohtuprojekti
- javalabra ohtuprojekti
- ohma ohtu
- ohtu ohtuprojekti
- lama ohtuprojekti

Each row declares one dependency, for example ohja tira defines that ohja must be passed before taking tira.

The program checks first, that the dependencies don’t form a cycle. If we added this line to the list above, ohtuprojekti tira

then a dependency cycle would be introduced:

- ohtuprojekti tira tiralabra ohtuprojekti

It is not necessary to print the dependency cycle. It is sufficient, if the program declares that a cycle is found. Printing out the cycle of course has style.

If there is no dependency cycle, the program prints one possible order, that doesn’t violate any dependency. For example:

- ohpe ohja javalabra tira tiralabra lama ohma ohtu ohtuprojekti

You can use any programming language.

23. Form the strongly connected components of the graph below using the algorithm in Cormen. Assume that nodes are in alphabetical order in the adjacency lists.

What is the smallest amount of edges needed to make the graph strongly connected?
24. We consider an undirected connected graph, with nodes representing computers and edges between them representing data communication links. The graph is vulnerable, if it has a node, which would make the graph disconnected were the node removed. In other words, the computer network is vulnerable, if removing a computer from the network would prevent the data communication between two other computers. For instance, the network to the left is vulnerable, as removal of \( b \) would make it disconnected. The network to the right is not vulnerable.

\[
\begin{array}{ccc}
\text{a} & \text{b} & \text{c} \\
\downarrow & \downarrow & \downarrow \\
\text{d} & \text{e} & \text{f} \\
\end{array}
\]

Give an algorithm that checks whether the graph is vulnerable. The graph is given in adjacency list mode. Analyse the time complexity of your algorithm.

25. (a) Prove that in every acyclic directed graph (DAG) there is a vertex to which there are no arcs from other vertices.

(b) On the basis of this, we can produce a topological sort as follows.

\[
\text{Topo}(V,E) \\
\quad \text{if } V \text{ is empty then stop} \\
\quad \text{search for a } v \text{ in } V \text{ with no incoming arcs} \\
\quad \text{print } v \\
\quad \text{Topo}(V-\{v\}, E - \{(v,u) \mid u \in V\})
\]

What is the time complexity of this method, if you implement the steps above in an efficient way? Is this worse or better than the one presented in the lectures (Cormen page 550)?

26. (a) Dijkstra’s algorithm presumes, that in the graph there are no edges, that would have a negative weight. Give an example of a situation, where Dijkstra’s algorithm gives a false solution, because a negatively weighted edge exists.

(b) You can avoid the problem by adding an appropriate constant value to each weight of the edge, so that every weight is positive. Thus we have a graph, which is suitable for the Dijkstra’s algorithm. Give an example, which demonstrates that this strategy, however, does not work.

27. Consider a communications network with \( n \) computers \( a_1, \ldots, a_n \).
If there is a direct communication link (say, cable) between computers $a_i$ and $a_j$, we denote by $p(i, j)$ the delay associated with the link. The delay $p(i, j)$ tells how long it takes for a (fixed-length) message sent from $a_i$ to reach $a_j$ using the link, and is always a positive real number. All existing links and their delays are known in advance. The links are not symmetrical, so it is possible that $p(i, j) \neq p(j, i)$.

Additionally, there is a delay $q(i)$ associated with each computer $a_i$. The delay $q(i)$ is the time that a message passing through $a_i$ must wait there before it can continue.

When a message is routed through several machines, the total time it requires to reach its destination is the sum of all the delays in the computers and links along its route. This includes the delays at the sending and receiving computer.

The task is to find for $j = 2, \ldots, n$ the fastest route from $a_1$ to $a_j$. Give an efficient algorithm for the task. Explain the basic idea of your solution briefly and analyze its time complexity.

You do not need to present the pseudocode for operations on any auxiliary data structures you wish to use (list, priority queue, balanced search tree, etc.); implementation of such data structures can be assumed to be available. Similarly, you can take the time complexities related to operations on such data structures as known.

28. Find the shortest paths starting from node $a$ in the graph below, using the Bellman-Ford algorithm. Assume that the arcs are stored in the adjacency lists in alphabetic order.

![Graph](image)

(a) Simulate the Floyd-Warshall algorithm in the case of the following graph:

![Graph](image)

(b) Extend the Floyd-Warshall algorithm so that besides finding out the shortest distances between nodes, the algorithm also finds the shortest paths between nodes. Demonstrate how you find the shortest paths in the graph above.

29. The longest increasing subsequence problem is as follows: Given numbers $a_1, a_2, \ldots, a_n$, find the maximum value $k$ such that $a_{i_1} < a_{i_2} < \cdots < a_{i_k}$ and $i_1 < i_2 < \cdots < i_k$.

As an example, if the input is $3, 1, 4, 1, 5, 9, 2, 6, 5$, the maximum increasing subsequence has length four ($1, 4, 5, 9$; among others). Give an $O(n^2)$ algorithm to solve the problem.
30. Write an algorithm, which given an \( n \times n \) matrix \( M \) of positive integers will find a sequence of adjacent entries starting from \( M[n, 1] \) and ending at \( M[1, n] \) such that the sum of the absolute values of differences between adjacent entries is minimized. Two entries \( M[i, j] \) and \( M[k, l] \) are adjacent if (a) \( i = k \pm 1 \) and \( j = l \), or (b) \( i = k \) and \( j = l \pm 1 \). For example, in the matrix below, the sequence 7, 5, 8, 7, 9, 6, 12 is a solution.

\[
\begin{array}{cccc}
1 & 9 & 6 & 12 \\
8 & 7 & 4 & 5 \\
5 & 9 & 11 & 4 \\
7 & 3 & 2 & 6 \\
\end{array}
\]

31. You are given a graph whose vertices present all airports of the world. The edges present all direct flights between two airports. The weight of each edge is the probability that the luggage of a passenger gets lost during the corresponding flight (this is calculated based on the statistics gathered by IATA). Your task is to plan a route from Helsinki to Addis Abeba where the probability of your luggage getting lost is as low as possible. How can you use Dijkstra’s algorithm to solve this problem?

**Hint:** If the probability to lose luggage between Helsinki and London is \( p_1 \), between London-Mumbai \( p_2 \) and between Mumbai-Chennai \( p_3 \), then the probability that the luggage does not get lost between Helsinki-London-Mumbai-Chennai is \( (1 - p_1)(1 - p_2)(1 - p_3) \) and the probability to luggage getting lost is \( 1 - (1 - p_1)(1 - p_2)(1 - p_3) \).

32. Consider the following game: you have a directed graph \( G \). Each vertex of the graph contains a coin worth one Euro. You may move your piece along the edges of the graph starting from a starting vertex \( s \) (which you may choose) until the piece reaches a given end vertex \( t \). However, you may only move your pieces in the direction of the edge: you cannot move the piece from node \( q \) to node \( p \) along the edge \((p, q)\). You cannot continue the game after you have reached the end vertex.

When the piece arrives at the node which contains a coin you may pick up this coin. You may visit a certain node more than once (if you can reach it again by using the edges), but you cannot pick up any coins at the later visits.

How can you calculate the vertex you should choose to start the game at, so that you can gather the maximum number of coins? You do not need to give an algorithm, an explanation how to solve the problem is enough.

33. Form the smallest spanning tree to the following graph by simulating the Kruskal algorithm.
34. Transports with very heavy trucks can be challenging, if there are weight-restricted bridges along the road. Let us study a situation in which an entrepreneur wants to know which is the maximum weight for a truck, when he wants that the truck can be used for transporting goods between any two company stores. We know for each pair of stores \((u,v)\) what is the maximum weight allowed \(w(u,v)\). How do you solve the problem? You do not need to give an algorithm, an explanation how to solve the problem is enough.

Note: It is not possible to simply search the arc with the least weight, as this could be circumvent by driving another route.

35. We have a set of variables \(x_1, \ldots, x_n\), and for them a set of equality constraints of the form \(x_i = x_j\) and inequality constraints of the form \(x_i \neq x_j\). Give an efficient algorithm do decide, whether all the constraints can be satisfied simultaneously.

36. We consider the union-find data structure, where union is implemented by choosing the higher tree of the two as the root (if both trees are of equal height, there is no difference which one to choose). Prove that trees constructed in this way has height \(O(\log n)\), where \(n\) is the number of nodes. Use induction.

37. Suppose we have \(n\) currencies \(c_1, \ldots, c_n\) among which we can make trades. For example, \(c_1\) is euro, \(c_2\) pound sterling, \(c_3\) US dollar etc. For each pair \((c_i, c_j)\) there is an exchange rate \(r_{ij}\) for buying currency \(c_j\) using currency \(c_i\). That is, \(x\) units of \(c_i\) buys you \(r_{ij} \cdot x\) units of \(c_j\). We assume that for all \(i\) and \(j\) the rate \(r_{ij}\) is positive and finite.

Usually there are costs associated with currency trading, so we would expect that for example \(r_{ij}r_{ji} < 1\) for all \(i, j\). However, occasionally for a short period of time a situation may exist, where some currencies \(c_{i_1}, \ldots, c_{i_k}\) satisfy

\[ r_{i_1i_2}r_{i_2i_3} \ldots r_{i_{k-1}i_k}r_{i_1i_k} > 1. \]

This means that funds we have in currency \(c_{i_1}\) can be increased for free by rotating it via currencies \(c_{i_2}, \ldots, c_{i_k}\). Give an algorithm for detecting such situations, when the exchange rates \(r_{ij}\) are given.