1. A sequence of items can be sorted by first inserting its keys into a 2–3–4 tree, and then picking them during an appropriate traversal.
   (a) Sketch the algorithm; Assume, that a sub-routine for inserting a key into a 2–3–4 tree is available.
   (b) Demonstrate the working of the algorithm by using it to sort the keys 3, 1, 4, 5, 11, 2, 6, 10, 7, and 9.
   (c) Analyze algorithm’s time complexity and need for space (of the tree). (Assume that records of a fixed size are used for the nodes of the tree, and that each key and pointer takes one memory word each.)

2. (a) Present a procedure \texttt{shiftUp(A, i)} that lifts the key \texttt{A[i]} at its proper level in 2-heap \texttt{A[1..i]}.
   (b) Simulate the transformation of the array \texttt{A[1..8] = [3, 1, 4, 1, 5, 9, 2, 6]} into a 2-heap by lifting each key \texttt{A[2], A[3], ..., A[8]} at its proper level.
   (c) Simulate the transformation of the same array into a 2-heap by shifting \texttt{down} the keys \texttt{A[4], A[3], A[2] ja A[1]}.

3. We can find the \( k \) largest items of an array \( A[1..n] \)
   (a) in time \( O(kn) \) using ”brute-force”,
   (b) in time \( O(n \log n) \) applying sorting, and
   (c) in time \( O(n + k \log n) \) by building a heap.

   Sketch the algorithms and justify their complexity.
   (d) Could we find the \( k \) largest items in time \( O(n) \), or \( O(k) \)? How, or why not?

4. Prim’s algorithm can be modified to run in time \( O(e \log e) \), where \( e \) is the number of edges in the graph. Present the algorithm, and analyze its complexity. Assume that the graph is given as adjacency lists, which give for each node a list of weighted edges emanating from the node. (Hint: maintain the edges that emanate from nodes covered by the tree in a priority queue.)

5. (a) At the lecture we considered as an example a dynamic programming solution for computing the number of shortest rectangular paths from square \((0, 0)\) to square \((n, n)\). Simulate how the algorithm computes the value \( L(5, 5) \).
   (b) One can observe a regularity that allows \( L(n, n) \) to be computed more efficiently. Based on this, present a more efficient algorithm for the task.