1. Insert letters of the word "uncopyrightable" in the given order into an initially empty B-tree, \( t = 2 \). Then delete letters of the word "cobra". Draw the resulting tree. The letters are sorted lexicographically: \( a < b < \ldots \).

2. How many heaps on keys \( \{1, 2, 3, \ldots , 12\} \) exist, where 4 is the right child of the root and 7 is left-left grandchild of the root? Draw at least one of these heaps.

3. Assume that a given array \( A \) contains elements \( \{1, 2, \ldots , n\} \) in some order. The order of these elements is represented by a permutation. Assume now, that \( A \) is going to be sorted by \textsc{QuickSort}.

   (a) How many permutations of the elements of \( A \) force each call recursive call on a subarray of size \( k \), for \( 1 < k \leq n \), to branch into only one recursive call to a single subarray of size \( k - 1 \)?

   (I.e., one of the possible cases is when \textsc{QuickSort}(A, p, r) yields the call \textsc{QuickSort}(A, p+1, r).)

   (b) How many permutations of the elements of \( A \) force each recursive call on a subarray of size \( k \), for \( 2 < k \leq n \), to branch into two recursive calls: one on an array of size 1, and the other on an array of size \( k - 2 \)?

   (Hint: consider first the case of \( n \) being odd.)

In other — less formal — words, in the case (a) calculate how many permutations cause that the recursion tree is a path. In (b) the recursion tree should be a caterpillar — a path with a pendant leaf on each internal node.

For both cases write the recurrence relations for the time complexity and resolve them.

4. Given an array of \( n \) items, all integers, describe an algorithm that finds the most frequent item of the array. Provide verbal description of the algorithm as well as its pseudocode. The running time should be \( O(n \log k) \), where \( k \) is the number of distinct items in the array. Prove the time complexity.

For extra points determine the space complexity of the algorithm. Do not encounter the memory allocated for the given array in the complexity.