PAL labs 6

26 / 10 / 2022

Given a certificate of a tree, describe how you can derive the maximal degree of a vertex in the tree without reconstructing the whole tree from the certificate.

Tree of type T(1,3) contains vertices of degree 1 or 3. Describe, informally, how the certificate of T(1,3)-tree looks like. Design an algorithm which verify if a certificate is of type T(1,3)-tree.

There is a set of small letters $P = \{'a', 'b', 'c', ..., 'z'\}$. We are asked to generate a table T with 8 columns and rows, where each row will contain one 8-elements subset of P; no subset-repetitions are allowed in the table T. Decide the size of the table and whether your PC is able to fill it in within 1 second.

Write down a pseudocode of a method printing out all unempty subsets of the set $\{0, 1, 2, ..., n-1\}$.

Let's have a permutation of the set $M = \{1, 2, 3, ..., n\}$, $n > 4$. We call a permutation p of M as *common* if the following hold:
$p(3) \in \{3, n\}$, $p(n) \in \{3, n\}$, $p(1) = 1$, $p(2) = 2$,
$p(i) \in \{4, ..., n-1\}$ for $i = 4, ..., n-1$. Compute the number of *common* permutations of M.

We rank all permutations of the set M with 98 elements by numbers from 0 to $98! - 1$. In a process, we are working with ranks of permutations; each time, we need to work with exactly 100 ranks of permutations. How many bits do we have to allocate in the memory in order to run the process?

Let's assume all permutations of the set $M = \{1, 2, 3, ..., n\}$. Follow the algorithm *nextPermutation* (in lexicographical order) to design an algorithm working in the opposite direction, i.e. *prevPermutation*. Will both the algorithms have the same asymptotic complexity?

nextPerm

- find the last element such that it is bigger than its predecessor , i.e. *while i > 0 and p[i-1] > p[i]: i − = 1*
- reorder the decreasing suffix into an increasing one, i.e. *swap (p[i-j], p[n-1-j]) j* $\in (0, \frac{(n-1-i)}{2})$
- find the smallest bigger value than the one at position $i - 1$ and swap those

1 2 4 6 5 3

prevPerm

- find the beginning of increasing suffix, i.e. *while i > 0 and p[i-1] < p[i]: i − = 1*
- reorder the increasing suffix into a decreasing one
- find the biggest value which is smaller than the one at position $i - 1$ and swap those

1 2 5 3 4 6

Let's assume all size$-k$ subsets of the set $M = 1, 2, 3, ..., n$, $1 \leq k \leq n$. Follow the algorithm *nextLexicographicalSubset* to design an algorithm working in the opposite direction, e.g. *previouLexicographicalSubset*. Will both of them be in the same asymptotic complexity class?

Let's assume a permutation of the set $M = \{1, 2, 3, ..., n\}$. We define a *cycle* of length k in the permutation p as follows: let A be a subset of M, i.e. $A = \{a_1, a_2, ..., a_k\} \in M$, for which it holds that $1 \leq a_1 < a_2 < ... < a_k \leq n$, $p(a_j) = a_{j+1}$ pro $1 \leq j < k$, $p(a_k) = a_1$.
Compute the number of permutations of the set $\{1, 2, 3, ..., n\}$, which contains exactly two cycles, one having the length of 4 and the second one having the length of $n - 4$.