PAL cv. 6

26 / 10 / 2022

4/10. Je dán certifikát stromu. Vysvětlete, jak určíme maximální stupeň uzlu tohoto stromu, aniž strom z certifikátu celý rekonstruujeme.

4/11. Strom typu T(1,3) obsahuje uzly pouze stupně 1 nebo 3. Popište neformálně jak bude vypadat certifikát takového stromu a navrhněte algoritmus, který pomocí certifikátu ověří, zda strom je skutečně typu T(1,3).

5/*1. Je dána množina malých písmen $P = \{'a', 'b', 'c', ..., 'z'\}$ bez diakritiky. Máme vygenerovat tabulku T s 8 sloupci, jejíž každý řádek bude obsahovat jednu 8 prvkovou podmnožinu množiny P tak, že každá buňka na řádku bude obsahovat právě jeden znak. Tabulka bude obsahovat všechny možné navzájem různé 8 prvkové podmnožiny P a žádná podmnožina se v T nebude opakovat. Určete, jak bude tabulka velká a zda ji váš počítač bude moci vyplnit během 1 sec.

5/*2. Napište pseudokód funkce, která vypíše všechny neprázdné podmnožiny množiny $\{0, 1, 2, ..., n-1\}$.

5/*3. Mějme permutace množiny $M = \{1, 2, 3, ..., n\}$, $n > 4$. Permutaci p této množiny prohlásíme za přívětivou, pokud platí: $p(3) \in \{3, n\}$, $p(n) \in \{3, n\}$, $p(1) = 1$, $p(2) = 2$, $p(i) \in \{4, ..., n-1\}$ pro $i = 4, ..., n-1$. Určete počet přívětivých permutací množiny M.

5/*5. Všechny permutace množiny M s 98 prvky očíslujeme pořadovými čísly od 0 do $98! - 1$. V programu pak nepracujeme s permutacemi ale jen s jejich pořadovými čísly. Víme, že budeme zkoumat pokaždé najednou 100 permutací, čili v paměti budeme muset mít uloženo právě 100 pořadových čísel různých permutací množiny M. Kolik minimálně bitů si musíme v paměti rezervovat, abychom si těchto 100 reprezentací mohli uložit?

5/9. Uvažujme všechny permutace množiny $M = \{1, 2, 3, ..., n\}$. Vyjděte z algoritmu transformujícího danou permutaci na permutaci bezprostředně následující v lexikografickém uspořádání. Navrhněte a popište algoritmus, který bude transformovat danou permutaci na permutaci bezprostředně předcházející v témže lexikografickém uspořádání. Bude mít stejnou asymptotickou složitost?

nextPerm

- najít první prvek od konce, který je větší než předchůdce (začátek poslední klesající sekvence), i.e. *while i > 0 and p[i-1] > p[i]: i −= 1*
- vzestupně uspořádat klesající část, i.e. *swap (p[i-j], p[n-1-j])* $j \in (0, \frac{(n-1-i)}{2})$
- najít nejmenší vyšší hodnotu hodnoty na pozici $i-1$ a tyto odvě prohodit

1 2 4 6 5 3

prevPerm

- najít začátek poslední rostoucí sekvence, i.e. *while i > 0 and p[i-1] < p[i]: i −= 1*
- sestupně uspořádat rostoucí sekvenci
- najít největší menší hodnotu hodnoty na pozici $i-1$ a tyto dvě prohodit

1 2 5 3 4 6

5/6. Uvažujme všechny k-prvkové podmožiny množiny $M = 1, 2, 3, ..., n$, $1 \leq k \leq n$. Vyjděte z algoritmu transformujícího seznam prvků jedné podmnožiny na seznam prvků podmožiny bezprostředně následující v lexikografickém uspořádání těchto podmnožin. Navrhněte a popište algoritmus, který bude transformovat seznam prvků jedné podmnožiny na seznam prvků podmožiny bezprostředně předcházející v témže lexokografickém uspořádání. Bude mít stejnou asymptotickou složitost?