ćwiczenia (licencjat dzienny): lista zadań nr 9 *5 grudnia 2007 r.*

ALGORYTMY I STRUKTURY DANYCH

SŁOWNIKI I DRZEWA BST

Instytut Informatyki Uniwersytetu Wrocławskiego *Paweł Rzechonek*

1. [**] Wylicz ile jest różnych drzew binarnych o n węzłach. Ułóż odpowiednie równanie rekurencyjne i przedstaw jego rozwiązanie (liczby *Catalana*). Udowodnij, że znalezione w podręczniku lub w internecie rozwiązanie tego równania jest prawdziwe?

2. [**] Niech dane będzie drzewo binarne o n węzłach. W porządku *preorder* węzły te tworzą ciąg $u_1, u_2, \ldots, u_n$, a w porządku *inorder* ciąg $u_{\sigma_1}, u_{\sigma_2}, \ldots, u_{\sigma_n}$. Pokaż, jak za pomocą stosu można zrealizować permutację σ (przekształcić ciąg $1, 2, \ldots, n$ w ciąg $\sigma_1, \sigma_2, \ldots, \sigma_n$)?

3. [**] Przedstaw algorytm, który podzieli drzewo BST na dwa odrębne drzewa BST względem zadanej wartości klucza x. W jednym z wynikowych drzew mają się znaleźć wszystkie węzły pierwotnego BST z kluczami $\leq x$, a w drugim z kluczami $> x$. Czas działania twojego algorytmu powinien być ograniczony przez głębokość węzła z wartością x w pierwotnym drzewie.

4. [***] Procedura szukająca klucza x w drzewie BST *search(x)* odwiedza wszystkie węzły od korzenia aż do węzła z wartością x (lub gdy x nie ma w drzewie, to do węzła z wartością x', najbliższą x z lewej albo z prawej strony). Wypisanie węzłów leżących na ścieżce od korzenia do x (albo do x') jest więc trywialnym zadaniem. Przedstaw algorytm, który pozwoli wypisać węzły w odwrotnej kolejności — od x (albo od x') do korzenia. Twój algorytm ma działać w miejscu (nie wolno więc korzystać z rekurencji) w czasie proporcjonalnym do głębokości węzła z wartością x (albo z wartością x'). Po wyjściu z procedury drzewo ma mieć taki sam kształt jak przed jej uruchomieniem (nie wolno zniszczyć pierwotnej struktury drzewa).

5. [**] Skonstruuj takie drzewo AVL o wysokości h, w którym wykonanie operacji *delete* będzie wymagało wykonania $\Theta(h)$ rotacji.

6. [***] Na wykładzie były omawiane drzewa AVL, które w każdym węźle pamiętały wysokość zaczepionego w nim poddrzewa (pole *height*). Zamiast wysokości można pamiętać tylko zrównoważenie poddrzewa w danym węźle (pole *balance* zamiast *height*), które będzie przyjmowało tylko trzy wartości: -1 gdy lewe poddrzewo jest wyższe od prawego, 0 gdy wysokości podrzew są takie same, +1 gdy prawe poddrzewo jest wyższe od lewego (różnica wysokości wynosi co najwyżej 1). Jak ta modyfikacja wpłynie na implementację operacji *insert* oraz *delete*?

7. [*] Wykaż, że najdłuższa prosta ścieżka z węzła x do czarnego liścia w drzewie RBT jest co najwyżej dwa razy dłuższa niż najkrótsza ścieżka z x do liścia.

8. [**] Jak wygląda drzewo RBT o n kluczach, w którym stosunek liczby czerwonych węzłów do czarnych węzłów wewnętrznych jest największy możliwych? Jaki jest ten stosunek? Dla jakiego drzewa ten stosunek jest możliwie największy, a dla jakiego najmniejszy?

9. [**] Wykaż, że każde drzewo BST o n węzłach można przekształcić w dowolne inne drzewo BST o n węzłach za pomocą $O(n)$ pojedynczych rotacji.