ćwiczenia (licencjat dzienny): lista zadań nr 8 *4 grudnia 2006 r.*

ALGORYTMY I STRUKTURY DANYCH

DRZEWA ZRÓWNOWAŻONE

Instytut Informatyki Uniwersytetu Wrocławskiego *Paweł Rzechonek*

1. [$*$] Przedstaw implementację w pseudokodzie pojedynczych rotacji w lewo i w prawo w drzewach AVL. Jak wykorzystać te operacje do implementacji podwójnych rotacji?

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

3. [$***$] W podanej na wykładzie implementacji drzewa AVL w każdym węźle pamiętaliśmy wysokość zaczepionego w nim poddrzewa (pole *height*). Zamiast wysokości można pamiętać tylko zrównoważenie poddrzewa w danym węźle (pole *balance*), 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*?

4. [$**$] Do początkowo pustego 2-3-4–drzewa wstawiamy klucze: 1, 2, ..., n. Jaką wysokość ma końcowe drzewo? Odpowiedź uzasadnij.

5. [$**$] Załóżmy, że wskaźnik do następnego węzła w B–drzewie można zapisać w słowie o takiej samej długości co klucz. Aby zaoszczędzić część pamięci, w liściach zamiast pustych wskaźników pamiętane są klucze. Jaka jest minimalna i maksymalna liczba kluczy pamiętanych w takim B–drzewie o wysokości h? O ile więcej kluczy można zapamiętać w B–drzewie o wysokości h z taką modyfikacją w stosunku do standardowego rozwiązania?

6. [$***$] Zaproponuj strukturę danych, której zadaniem jest wykonywanie operacji słownikowych dla dynamicznego słownika zewnętrznego, w którym każde dwa klucze są różne, oraz dodatkowo realizacja operacji:

   a. *min()* — zwracającej element minimalny w słowniku;

   b. *max()* — zwracającej element maksymalny w słowniku;

   c. *med()* — zwracającej element środkowy w słowniku (klucz który jest medianą spośród wszystkich pamiętanych aktualnie kluczy).

   Czas wykonania tych operacji powinien być stały $O(1)$. Podaj algorytmy realizujące te operacje oraz ewentualne zmiany w procedurach słownikowych *search*, *insert* i *delete*.

7. [$*$] W jaki sposób zmodyfikować drzewo *RST*, aby można było w nim przechowywać elementy indeksowane kluczami o różnej długości? Jak zmienią się wtedy operacje *insert* i *delete*?

8. [$**$] Jak obliczyć wysokość drzewa RST, zbudowanego z zadanego zestawu n kluczy r-bitowych, bez budowania tego drzewa? Twój algorytm ma działać w liniowym czasie.

9. [$**$] Drzewa *TRIE* doskonale nadają się do pamiętania elementów indeksowanych łańcuchami (bitów, cyfr, liter, itp.) o wspólnym przedrostku. Żaden łańcuch nie może być jednak prefiksem innego łańcucha. Jak zmodyfikować drzewo TRIE, aby można było w nim zapamiętywać elementy indeksowane dowolnymi łańcuchami (bez ograniczenia prefiksowego)?