ćwiczenia (licencjat dzienny): lista zadań nr 10 *12 grudnia 2007 r.*

ALGORYTMY I STRUKTURY DANYCH

SŁOWNIKI ZEWNĘTRZNE I DRZEWA POZYCYJNE

Instytut Informatyki Uniwersytetu Wrocławskiego *Paweł Rzechonek*

1. [∗] Niech T będzie n–elementowym drzewem SPLAY, w którym zapamiętane są kolejne liczby naturalne $1, 2, \ldots, n$. Przedstaw taki ciąg operacji *splay()*, w wyniku którego otrzymamy drzewo o wysokości $n-1$.

2. [∗∗] Niech T będzie drzewem SPLAY, w którym na głębokości h znajduje się węzeł z kluczem x. Po wykonaniu operacji *splay(x)* na drzewie T otrzymamy drzewo T' z wartością x w korzeniu. Udowodnij, że wszystkie węzły leżące na ścieżce z korzenia do x w drzewie T po wykonaniu operacji *splay(x)* zmiejszą swoją głębokość o połowę w nowym drzewie T'. Ściślej, jeśli węzeł y leżał w drzewie T na ścieżce z korzenia do x na głębokości h_y, to po wykonaniu operacji *splay(x)* znajdzie się w nowym drzewie T' na głębokości $h'_y \leq \lceil \frac{h_y}{2} \rceil + 2$.

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

4. [∗∗] 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 maksymalna liczba kluczy pamiętanych w tak zmodyfikowanym 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?

5. [∗∗] Zmodyfikuj strukturę B–drzewa tak, aby oprócz efektywnie wykonywanych operacji słownikowych można było też wykonywać dodatkowe instrukcje:

   (a) *min()* — zwraca element minimalny w słowniku;

   (b) *max()* — zwraca element maksymalny w słowniku;

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

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

6. [∗∗] Przedstaw implementację operacji słownikowych w drzewie RST. Złożoność twoich procedur powinna być ograniczona tylko przez wysokość drzewa.

7. [∗∗∗] Załóżmy, że elementy zbioru S pamiętane w drzewie RST to $(maxb+1)$–bitowe klucze. Zbiór S zawiera n losowych kluczy, czyli dla klucza $v = \langle v_{maxb} \ldots v_1 v_0 \rangle \in S$ na każdej pozycji wartość 0 lub 1 występuje z prawdopodobieństwem $\frac{1}{2}$. Niech $G(n)$ oznacza sumę długości ścieżek od korzenia do wszystkich wierzchołków w losowym RST o n wierzchołkach:

$$\begin{cases} G(0) = G(1) = 0 \\ G(n) = (n-1) + \sum_{j=0}^{n-1} p_j (G(j) + G(n-1-j)) \quad : \quad n > 1 \end{cases}$$

   Prawdopodobieństwo p_j oznacza, że w $n-1$ losowaniach j razy padło 0 oraz $n-1-j$ razy padło 1 (gdy zostało wylosowane 0, wstawiamy klucz do lewego poddrzewa, a w przeciwnym przypadku do prawego). Udowodnij, że $G(n) \in n \log n$. Jak ten wynik przekłada się na oczekiwaną złożoność czasową operacji słownikowych w losowym RST?

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.