Ćwiczenia: lista zadań nr 14 *10 stycznia 2006*

Algorytmy i Struktury Danych

drzewa BST

Instytut Informatyki Uniwersytetu Wrocławskiego *Paweł Rzechonek*

1. [∗] Znajdź wszystkie drzewa binarne, których węzły tworzą ten sam ciąg, gdy wypisze się je w porządkach:

   (a) *preorder* i *inorder*;

   (b) *preorder* i *postorder*;

   (c) *inorder* i *postorder*;

   Uwaga: Zakładamy, że żadne dwie wartości w takich drzewach nie powtarzają się.

2. [∗∗] Węzły pewnego drzewa binarnego zostały wypisane w porządku *inorder* i *postorder*. Pokaż, jak można odtworzyć strukturę drzewa, mając takie dwa ciągi danych.
   Uwaga: Zakładamy, że żadne dwie wartości w tym drzewie nie powtarzają się.

3. [∗∗∗] 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 $\sigma_1, \sigma_2, \ldots, \sigma_n$)?

4. [∗] Pokaż, jak za pomocą kolejki można wypisać zawartość drzewa binarnego poziomami (korzeń znajduje się na poziomie 0, jego synowie na poziomie 1, wnukowie korzenia na poziomie 2, itd).

5. [∗∗] Węzły drzewa binarnego można utożsamiać z ciągami zer i jedynek w następujący sposób:

   - korzeniowi (jeśli istnieje) odpowiada ciąg *1*;
   - korzeniom lewego i prawego poddrzewa węzła, któremu odpowiada ciąg α, odpowiadają odpowiednio ciągi α*0* i α*1*.

   Pokaż, jak za pomocą tej notacji można wygodnie zdefiniować porządki *preorder*, *inorder* i *postorder*.

6. [∗] Pokaż, że w drzewie BST o n węzłach jest zapamiętanych $n + 1$ wskaźników pustych.

7. [$**$] Dodajmy do węzła w drzewie BST jeszcze jedno pole *next*, które będzie kolejnym wskaźnikiem na węzeł. Pole to będziemy chcieli wykorzystać, aby w którymś momencie istnienia drzewa przeszyć go listą jednokierunkową (od elementu najmniejszego do największego). Przedstaw algorytm, który zadane drzwo BST o n węzłach przeszyje listą. Twój algowytm powinien działać w miejscu (nie wolno korzystać z rekurencji) i w liniowym czasie $O(n)$.

8. [$*$] Jak przeprojektować drzewo BST, aby można było łatwo znajdować k-ty co do wielkości element w drzewie? Napisz funkcję *index(int k)*, która będzie realizowała to zadanie w sposób rekurencyjny. Jak zmienią się wtedy metody modyfikujące drzewo BST: *insert()* i *delete()*?

9. [$**$] 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 $> x$.

10. [$*$] Napisz procedury, które znajdują element poprzedni i następny co do wielkości względem zadanej wartości klucza x w drzewie BST.

11. [$***$] Budujemy n-elementowe drzewo BST wstawiając do niego w losowej kolejności początkowe liczby naturalne $\{1, 2, \ldots, n\}$. Zakładamy, że każdy ciąg wstawień (permutacja $1, 2, \ldots, n$) jest jednakowo prawdopodobny, oraz że $n = 2^k - 1$. Jakie jest prawdopodobieństwo, że zbudowane drzewo będzie drzewem pełnym?

12. [$***$] Ile jest różnych drzew BST z n węzłami (chodzi o kształt drzewa) o wysokości $n-1$? Ile jest różnych sposobów (chodzi o liczbę permutacji) takiego wstawiania n różnych kluczy do początkowo pustego drzewa BST, które kończą się utworzeniem drzewa o wysokości $n-1$?

13. [$**$] Opisz drzewo BST zawierające n węzłów, dla którego średnia głębokość węzła wynosi $\Theta(\log n)$, ale wysokość drzewa jest $\omega(\log n)$. Jak bardzo wysokie może być takie drzewo BST?