AiSD, egzamin – 10.02.2014 (z poprawkami)

1. (15p) Niech $f \prec g$ oznacza, że $f \in O(g)$ i $g \notin O(f)$ a $f \approx g$ oznacza, że $f \in O(g)$ i $g \in O(f)$. Uporządkuj następujące funkcje: $n\sqrt{n}$, $2^{\log(n^2)}$, $n(\log(n))^2$, n^3.

2. (15p) Wyznacz f taką, że czas działania algorytmu znajduje się w $\Theta(f)$:
   BlaBla(int *t, int koniec){
   out:=1; for(i:=1 to i=koniec/2) out:=out + t[i];
   return out * BlaBla(t, koniec/2); }%BlaBla

3. (15p) Sortowanie tablicy $t[1 \ldots 7] = \{5, 2, 8, 3, 9, 1, 7\}$ algorytmem quicksort dzielacym wedlug ostatniego elementu tablicy, t[koniec]:
   QS(int *t, int poczatek, int koniec){
   if(poczatek>= koniec) return;
   if (poczatek+1==koniec and t[poczatek] > t[koniec]) {
   swap(t, poczatek, koniec); return; }
   i:=partition(t, poczatek, koniec); swap(t, i, koniec);
   QS(t,poczatek, i-1); QS(t, i+1, koniec); } %QS

   Wypisz drzewo wywołań rekurencyjnych w postaci QS(t, i, j) oraz zawartość tablicy w momencie każdego wywołania (korzeń drzewa to QS(t,1,7), [5,2,8,3,9,1,7]).

4. (15p) Wypisz kolejne drzewa BST powstałe przez wykonanie następujących operacji na pustym drzewie: i(3), i(8), i(6), i(1), i(4), i(7), i(10), d(3) (10p). Usuwając element, który nie jest liściem wpisujemy na jego miejsce odpowiedni element z jego prawego podrzewa. Następnie, stosując rotacje przesuń element 7 do korzenia (5p).

5. (15p) Wysokość pustego drzewa to długość jego najdłuższej ścieżki (czyli liczba krawędzi na tej ścieżce). Wysokość pustego drzewa (przez konwencję) to -1, wysokość drzewa jednolementowego to 0.

   Napisz algorytm `wysokosc(node *)` , który wywołany dla argumentu `tree` zwróci wysokość drzewa, na które wskazuje `tree`. Przyjmij, że `node` jest strukturą postaci `struct node{int key; node *left; node *right;}`

6. (15p) Wyszukujemy z 6 elementów o kluczach: a, b, c, d, e ,f. Prawdopodobieństwa wyszukiwania elementu o danym kluczu są podane w nawiasach: a(0.3), b(0.1), c(0.2), d(0.2), e(0.1), f(0.1). Zbuduj optymalne drzewo BST, które pozwoli zminimalizować koszt wyszukania jednego elementu (10p). Oblicz średni koszt wyszukania elementu (czyli średnią liczbę odwiedzonych elementów) (5p). Możesz założyć, że nigdy nie są wyszukiwane elementy spoza listy.