Algoryty i Struktury Danych 2013

Lista 1

1. Udowodnij, że:

   (a) Każdy wielomian stopnia n jest klasy $O(n)$.

   (b) Jeśli f, g są wielomianami stopnia n z dodatnim wiądącym współczynnikiem, to $f(n) = O(g(n))$ oraz $g(n) = O(f(n))$.

2. Udowodnij, że dla każdego $\varepsilon > 0$ zachodzi $\log n = O(n^{\varepsilon})$.

3. (a) Udowodnij, że ilość porównań niezbędna dla znalezienia maksimum z n elementów wynosi $n - 1$.

   (b) Ile porównań potrzeba, by znaleźć maksimum i minimum z n elementów?

4. Wykazaliśmy, że nie można posortować n liczb używając mniej niż $\lceil \log_2(n!) \rceil$ porównań. Dla jakiego największego n potrafisz wskazać algorytm (sposób postępowania) gwarantujący posortowanie n elementów przy użyciu nie więcej niż $\lceil \log_2(n!) \rceil$ porównań. Wskazówka: Szczególnie ciekawe są odpowiedzi większe niż 5.

5. Przyjmij następującą definicję węzła:

```
struct node
{   int key;
    node* next;
    node(int k0,node* n0=NULL):key(k0),next(n0){}
};
```

   Napisz procedury:

   (a) `void insert(node*& l,int x)` – umieszcza x na początku listy l

   (b) `int pop(node*& l)` – zwraca pierwszy element listy i usuwa go z listy.

   (c) `lnode* find (lnode* l, int x)` – zwraca wskaźnik na wezęł zawierający klucz x lub NULL gdy nie ma takiego węzła.

   (d) `int sum(node* l)` – zwraca sumę elementów listy l.

   (e) `void reverse(node*& l)` – odwraca kolejność elementów listy l.

6. Dana jest tablica n elementowa. `int t[n];` Napisz funkcję `void shift(int t[],int n, int k)` która przesuwa elementy tablicy `t` cyklicznie prawo o k pozycji. Jeśli $k < 0$ to oznacza przesunięcie w lewo. Jeśli $k > n$ to przesuwa o $k \bmod n$ pozycji. Ilość dodatkowej pamięci użytej przez procedurę musi być stała (niezależna od n oraz k).

7. Zakładając, że elementy tablicy t są posortowane napisz funkcję `int gdzie(int t[], int x)`, która zwraca pozycję w tablicy t na której występuje element x. Użyj metody bisekcji. Jaka jest złożoność czasowa twojego algorytmu? Jaka byłaby średnia złożoność (wartość oczekiwana $T(n)$ - faktycznie wykonanej liczby porównań) dla tablicy nieposortowanej w przypadku gdy: (a) element x jest w tablicy (b) elementu x nie ma w tablicy t. Oblicz również wariancję $T(n)$ w obu przypadkach.

Zadania z programowania prezentujemy w postaci wydruków z komputera, po upewnieniu się, że działają poprawnie.