Algorytmy i Struktury danych

Lista zadań 3 (rekurencja, drzewa, sortowanie)

1. Korzystając z twierdzenia o rekurencji uniwersalnej rozwiąż następujące zależności:

   (a) $T(N) = 5T(n/3) + n$,
   (b) $T(N) = 4T(n/2) + n^2$,
   (c) $T(N) = 9T(n/3) + n^2$,
   (d) $T(N) = 6T(n/3) + n^2$,
   (e) $T(N) = 3T(n/3) + n$,
   (f) $T(N) = 5T(n/2) + n^2$,
   (g) $T(N) = T(n/2) + 1$.

2. Ile porównań wykona algorytm `insertion_sort` w wersji z wartownikiem (liczbą $-\infty$ zapisaną pod adresem `t[-1]`), jeśli dane $(a_1, ..., a_n)$ o rozmiarze `n` zawierają `k` inwersji. Liczba inwersji to liczba takich par (i, j), że $i < j$ i $a_i > a_j$. Jaka jest maksymalna możliwa liczba inwersji dla danych rozmiaru n?

3. Napisz procedurę `void insertion_sort(lnode*& L)` - sortowanie przez wstawianie działające na liście jednokierunkowej.

4. Napisz procedurę `void merge_sort(node*& n)` - sortowanie przez złączanie działające na liście jednokierunkowej, nie używaj rekurencji.
   Skorzystaj z procedury `lnode* merge(lnode* L1, lnode* L2))` z poprzedniej listy.

5. Napisz procedurę `merge` działającą na tablicy tak, aby nie wykorzystywać dodatkowego bufora, kosztem zwiększenia złożoności do $O(n \log n)$. Wskazówka: w czasie $O(n)$ można wykonać przejście $(1, 3, 5, 7, 9|2, 4, 6, 8, 10) \to (1, 2, 5|2, 4|7, 9|6, 8, 10)$ które powoduje, że każdy element lewej części jest mniejszy od każdego w prawej, a następnie rekurencyjnie wywołać `merge` dla każdej części. Jaka będzie wtedy złożoność `mergesort`?

6. W pliku jest $n = 10^6$ liczb całkowitych. Ile potrzeba pamięci i dodawań by sprawdzić, która z sum $k = 1000$ kolejnych liczb jest największa? Czy potrafisz zrobić tak, aby całkowity rozmiar utworzonych zmiennych był mniejszy niż 20-bajtów, niezależnie od wartości n i k?

7. Napisz nierekurencyjną procedurę `int poziom(BSTnode * t, int klucz)`, której wynikiem jest poziom w drzewie `t`, na którym występuje `klucz`. Wynik 0 oznacza brak klucza w drzewie, 1 - klucz w korzeniu, 2 - w dziecku korzenia itd.

8. Jaką dodatkową informację należy przechowywać w każdym węźle drzewa binarnego, by szybko znajdować i-ty co do wielkości z zawartych w nim elementów? Napisz implementację funkcji `BTSnode* ity(BSTnode *t, int i)`, korzystającą z tego dodatkowego pola, która będzie działała w czasie $O(\log n)$ dla drzew zrównoważonych. Jak należy zmienić rekurencyjne wersje procedur `insert` i `remove`, by informacja ta była automatycznie uaktualniana.

9. Niech $K(n)$ oznacza ilość różnych kształtów drzew binarnych o n węzłach.
   (a) Znajdź wzór rekurencyjny wyrażający $K(n)$ przez $\{K(i) : i < n\}$.
   (b) Napisz procedurę rekurencyjną, która używa tego wzoru.
   (c) Napisz procedurę nierekurencyjną, która oblicza po kolei wyrazy ciągu $K(n)$ i zapisuje je w tablicy. Przy obliczaniu kolejnych wyrazów, korzysta w poprzednio zapisanych wyników.
   (d) Uruchom programy (b) i (c) dla $n = 1000$ lub większego.
   (e) Oszacuj złożoność obu programów.