ćwiczenia (licencjat dzienny): lista zadań nr 2 *16 października 2006 r.*

ALGORYTMY I STRUKTURY DANYCH

SORTOWANIE

Instytut Informatyki Uniwersytetu Wrocawskiego *Paweł Rzechonek*

1. [$*$] Jaki jest oczekiwany czas działania procedury BINARY–SEARCH przy założeniu, że pytamy tylko o elementy występujące w tablicy, oraz że pytamy z jednakowym prawdopodobieństwem o każdy element.

2. [$**$] Przedstaw algorytm i jego implementację w pseudokodzie, który pozwoli zrealizować w miejscu dowolną zadaną permutację $P[0 \ldots n-1]$ elementów tablicy $T[0 \ldots n-1]$. Algorytm ma działać w liniowym czasie. Postaraj się, aby wykonywał on co najwyżej $\frac{3}{2}n$ przepisań elementów.

3. [$*$] Wykaż, że algorytm *sortowania przez wstawianie* na ciągu n-elementowym wykona liniową liczbę porównań i zamian (będzie działał w liniowym czasie), gdy całkowita liczba inwersji w sortowanym ciągu jest rzędu $O(n)$.

4. [$**$] Ile porównań elementów w średnim przypadku wykona algorytm *sortowania przez wstawianie*?

5. [$**$] Dana jest prostokątna tablica elementów, które możemy porównywać relacją $\leq$. W tablicy tej najpierw sortujemy wszystkie wiersze, a potem wszystkie kolumny. Wykaż, że po posortowaniu kolumn wiersze nadal pozostaną posortowane.

6. [$***$] Dany jest n-elementowy ciąg elementów, które możemy porównywać relacją $\leq$, oraz dwie liczby naturalne g i h, takie że $1 < g < h < n$. Na ciągu tym wykonujemy najpierw h-sortowanie, a potem g-sortowanie. Wykaż, że po wykonaniu g-sortowania ciąg ten będzie nadal h-posortowany.

7. [$*$] Pokaż, jak zaimplementować algorytm *scalania* dwóch ciągów n-elementowych, który będzie korzystał z pomocniczego bufora tylko o długości n. Wynik scalania należy umieścić w tablicach wejściowych. Jak duży bufor będzie potrzebny w twoim algorytmie w przypadku, gdy scalane ciągi będą różnych długości?

8. [$***$] Na wykładzie został przedstawiony uproszczony algorytm *Kronroda* scalania *w miejscu* dwóch posortowanych ciągów z liczbami zapisanych w jednej tablicy. Posortowane ciągi o długościach odpowiednio p i $n-p$, gdzie $0 < p < n$, są umieszczone w n-elementowej tablicy $T[0 \ldots n-1]$, w taki sposób że $T_0 \leq T_1 \leq \ldots \leq T_{p-1}$ oraz $T_p \leq T_{p+1} \leq \ldots \leq T_{n-1}$. W wersji uproszczonej zakładaliśmy, że $\sqrt{n} \in \mathbf{N}$ oraz $\sqrt{n} \mid p$ Uogólnij ten algorytm na przypadek dowolnych wielkości n i p, aby nie pogorszyć liniowej złożoności algorytmu.