

ALGORYTMY I STRUKTURY DANYCH

SORTOWANIE

Instytut Informatyki Uniwersytetu Wrocławskiego

Paweł Rzechonek

1. [*] Czasami zdarza się sytuacja, gdy wielokrotnie powtarzają się dane o takim samym kluczu. Jak należy zmodyfikować procedurę `BINARY-SEARCH`, aby zwracała ona numer pierwszej pozycji, na której wartość x występuje w tablicy $T[0 \dots n-1]$? A jak wyznaczyć ostatnią pozycję? Modyfikacje te nie powinny wpływać na złożoność algorytmu.
2. [**] Przedstaw algorytm i jego implementację w pseudokodzie, który pozwoli zrealizować w miejscu dowolną zadaną permutację $P[0 \dots n-1]$ elementów tablicy $T[0 \dots n-1]$. Algorytm ma działać w liniowym czasie. Postaraj się, aby wykonywał on co najwyżej $\frac{3}{2}n$ przepisania elementów.
3. [*] Wykaż, że algorytm *sortowania bąbelkowego* ciągu n -elementowego wykona liniową liczbę zamian elementów (choćby liczba porównań zawsze będzie kwadratowa) w przypadku, gdy liczba inwersji związanych z każdym elementem jest rzędu $O(1)$.
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. [*] 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?
7. [***] 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 $T[0 \dots n-1]$, w taki sposób że $T_0 \leq T_1 \leq \dots \leq T_{p-1}$ oraz $T_p \leq T_{p+1} \leq \dots \leq T_{n-1}$. 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.