Ćwiczenia: lista zadań nr 10 *6 grudnia 2005*

Algorytmy i Struktury Danych

SORTOWANIE

Instytut Informatyki Uniwersytetu Wrocławskiego *Paweł Rzechonek*

1. [∗∗] Udowodnij, że w modelu drzew decyzyjnych scalenie dwóch ciągów uporządkowanych o długościach m i n, gdzie $m \leq n$, wymaga wykonania $\Omega\left(m(\log \frac{n}{m} + 1)\right)$ porównań.

2. [∗∗] Udowodnij, że w modelu drzew decyzyjnych wyznaczenie k-tego co do wielkości elementu w nieuporządkowanym ciągu n-elementowym wymaga wykonania $\Omega\left(\log k + \log \binom{n}{k}\right)$ porównań.

3. [∗∗] Jakie jest dokładne dolne ograniczenie na liczbę porównań dla algorytmu sortującego 5 elementów? Podaj algorytm, który posortuje 5-elementową tablicę tylko przy pomocy porównań i zamian, i który wykona nie więcej porównań niż to wynika z dolnej granicy.

4. [∗] Wykaż, że algorytm sortowania przez wstawianie *Insertion-Sort* wykona w średnim przypadku około $\frac{n^2}{4}$ porównań.

5. [∗] Jakie poprawki można wprowadzić do algorytmów *Bubble-Sort* i *Cocktail-Sort*, aby maksymalnie przyspieszyć ich działanie?

6. [∗] Wykaż, że algorytm sortowania bąbelkowego *Bubble-Sort* wymaga jedynie $O(n)$ porównań i zamian w przypadku sortowania ciągów, w których każdy element pozostaje w inwersji z co najwyżej stałą liczbą innych elementów.

7. [∗] Podaj przykład danych, w których całkowita liczba inwersji wynosi $O(n)$, a algorytm sortowania koktajlowego *Cocktail-Sort* będzie musiał wykonać $\omega(n)$ porównań i zamian zanim ciąg nie będzie posortowany.

8. [∗] Pokaż, jak zaimplementować algorytm *Quick-Sort*, aby w pesymistycznym przypadku działał on w czasie $O(n \log n)$ na danych rozmiaru n.

9. [∗∗∗] Pokaż, jak zaimplementować algorytm *Quick-Sort*, aby działał on *w miejscu* i nie korzystał z rekurencji.

10. [∗] Pokaż, jak zaimplementować algorytm *Merge-Sort*, aby działał on *w miejscu* i nie korzystał z rekurencji. Zakładamy, że mamy gotową procedurę scalającą *Merge*, która działa *w miejscu*.

11. [$*$] Jak zaimplementować *Counting-Sort*, aby działał on *stabilnie*?

12. [$**$] W tablicy n-elementowej zapisane są liczby z zakesu $0 \ldots n^2 - 1$. Skonstruuj algorytm, który posortuje te liczby w liniowym czasie.

13. [$**$] Udowodnij prze indukcję, że algorytm sortowania pozycyjnego *Radix-Sort* działa poprawnie.

14. [$**$] Zmodyfikuj podany na wykładzie algorytm sortowania leksykograficznego *Radix-Sort*, tak aby sortował ciągi o różnej długości. Twojemu algorytmowi musi wystarczyć pamięć rozmiaru $\sum_{i=0}^{n-1} l_i$, gdzie l_i dla $i = 0 \ldots n-1$ to długość i-tego ciągu. Oszacuj złożoność czasową zmodyfikowanego algorytmu.