ćwiczenia (licencjat dzienny): lista zadań nr 3 *24 października 2007 r.*

ALGORYTMY I STRUKTURY DANYCH

SORTOWANIE LINIOWE, SCALANIE

Instytut Informatyki Uniwersytetu Wrocławskiego *Paweł Rzechonek*

1. [**] W n–elementowej tablicy zapisane są liczby wymierne postaci $\frac{p}{2^q}$, gdzie $0 \leq p, q \leq 10$. W oparciu o algorytm *counting–sort* skonstuuj metodę, która posortuje te liczby w liniowym czasie.

2. [*] Jak należy zmodyfikować algorytm *bucket–sort*, aby w najgorszym przypadku działał w czasie $O(n \log n)$?

3. [***] Zmodyfikuj podany na wykładzie algorytm *radix–sort* tak, aby sortował ciągi o różnej długości. Jeśli danych jest n ciągów $A_1, \ldots, A_n$ o długościach odpowiednio $l_1, \ldots, l_n$ i ciągi te złożone są z liter alfabetu Σ, gdzie $|\Sigma| = k$, to twój algorytm powinien działać w czasie $O(k + \sum_{i=1}^{n} l_i)$.

4. [**] 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.

5. [**] Dane są dwa posortowane ciągi $A = (a_0, a_1, \ldots, a_{m-1})$ i $B = (b_0, b_1, \ldots, b_{m-1})$ zapisane w jednej $(m + n)$–elementowej tablicy (najpierw ciąg A a potem B). Pokaż, jak zaimplementować algorytm *scalania* takich ciągów, który będzie korzystał z pomocniczego bufora o rozmiarze $\min\{m, n\}$.

6. [***] Na wykładzie została przedstawiona uproszczona wersja *algorytmu Kronroda* scalania *w miejscu* dwóch posortowanych ciągów zapisanych w jednej tablicy $T[0 \ldots n-1]$. Posortowane ciągi mają długości odpowiednio p i $n - p$, gdzie $0 < p < n$ oraz $\sqrt{n} \in \mathbf{N}$ i $\sqrt{n} \mid p$ Uogólnij ten algorytm na przypadek dowolnych wielkości n i p, tak aby nie pogorszyć liniowej złożoności algorytmu.

7. [**] W przedstawionej na wykładzie rekurencyjnej wersji algorytmu *sortowania przez scalanie* dane wejściowe o rozmiarze n były dzielone na dwie równe z dokładnością do 1 części o rozmiarach odpowiednio $\lfloor \frac{n}{2} \rfloor$ i $\lceil \frac{n}{2} \rceil$. Jak zmieni się złożoność czasowa tego algorytmu, gdy dane wejściowe będziemy dzielić na dwie części o rozmiarach $\lfloor \frac{n}{3} \rfloor$ i $\lceil \frac{2n}{3} \rceil$?

8. [***] Zaimplementuj w pseudokodzie następujący pomysł na *sortowanie przez scalanie* bez używania dodatkowych buforów na dane. Najpierw dzielimy ciąg wejściowy na równe z dokładnością do 1 części i rekurencyjnie sortujemy pierwszą z nich; następnie drugą połowę dzielimy na dwie równe z dokładnością do 1 części i znowu rekurencyjnie sortujemy pierwszą z nich; potem scalamy pierwszą posortowaną połowę danych z następną posortowaną ćwiartką, wykorzystując jako wewnętrzny bufor pomocniczy przy scalaniu ostatnią ćwiartkę z danymi. Ostatnie dwa kroki powtarzamy do momentu, aż ostatnia nieuporządkowana część danych przeznaczona na bufor stanie się mała (2 lub 3 elementy); wtedy tych kila ostatnich elementów wkładamy do posortowanego ciągu tak jak to robił algorytm sortowania przez wstawianie. Oszacuj złożoność czasową i pamięciową tego algorytmu.