ćwiczenia (licencjat dzienny): lista zadań nr 11 *1 stycznia 2007 r.*

ALGORYTMY I STRUKTURY DANYCH

METODA „DZIEL I ZWYCIĘŻAJ"

Instytut Informatyki Uniwersytetu Wrocławskiego *Paweł Rzechonek*

1. [∗] Rozważmy pewną modyfikację *sortowania głupiego*, która polega na przepychaniu na koniec (połączonym z sortowaniem) $\frac{1}{4}n$ największych elementów (operację tą powtarzamy trzykrotnie):

```
procedure STOOGE-SORT-4 (key T[0...n−1])
{
    if n = 2 then { COMPARE-EXCHANGE (T[0], T[1]); return; }
    p1 ← 1/4 n; p2 ← 2/4 n; p3 ← 3/4 n;
    /* pierwszy etap przepychania z sortowaniem */
    STOOGE-SORT-4 (t[0...p2−1]);
    STOOGE-SORT-4 (t[p1...p3−1]);
    STOOGE-SORT-4 (t[p2...n−1]);
    /* drugi etap przepychania z sortowaniem */
    STOOGE-SORT-4 (t[0...p2−1]);
    STOOGE-SORT-4 (t[p1...p3−1]);
    /* trzeci etap przepychania z sortowaniem */
    STOOGE-SORT-4 (t[0...p2−1]);
}
```

   Uzasadnij poprawność tego algorytmu i oszacuj jego złożoność korzystając z *uniwersalnego twierdzenia o rekurencji*.

2. [∗∗] Dane są dwie posortowane n-elementowe tablice liczb oraz liczba całkowita k z zakresu $1 \leq k \leq 2n$. Opisz algorytm znajdowania k-tej co do wielkości liczby spośród liczb zapisanych w obu tablicach. Czas działania twojego algorytmu powinien być rzędu $O(\log n)$. Uzasadnij jego poprawność.

3. [∗∗∗] Niech dane będą dwie długie liczby w systemie d-arnym: jedna o długości m i druga o długości n. Załóżmy, że $m \leq n$. Gdy $m \simeq n$, to algorytm pisemny oblicza iloczyn takich liczb w czasie $O(n^2)$, a algorytm z wykładu oparty na technice „dziel i zwyciężaj" potrzebuje tylko $O(n^{\log 3})$ czasu. Ale gdy m jest znacznie mniejsze niż n, to również ten ostatni algorytm jest nie do zaakceptowania. Skonstruuj algorytm, który dla przypadku gdy $m \ll n$ będzie działał w czasie $O(n\, m^{\log \frac{3}{2}})$.

4. [***] Zmodyfikuj algorytm mnożenia długich liczb metodą „dziel i zwyciężaj” w ten sposób, aby liczba była dzielona na trzy części zamiast na dwie. Ile mnożeń krótszych liczb musisz wtedy wykonać? Jaka jest złożoność czasowa twojego algorytmu?
Uwaga! Wynik dla podziału na trzy części powinien być lepszy niż dla podziału na dwie części.

5. [*] Wyznacz wartość progową (*threshold value*) opłacalności stosowania algorytmu Strassena w stosunku do tradycyjnego algorytmu mnożenia macierzy.

6. [**] Jak zmodyfikować algorytm Strassena bez pogarszania jego asymptotycznej złożoności $O(n^{\log 7})$, aby mnożył macierze kwadratowe $n \times n$ metodą „dziel i zwyciężaj”, w których n nie jest potęgą 2 ? Jak uogólnić metodę Strassena, aby można nią było mnożyć macierze o rozmiarach $a \times b$ i $b \times c$ dla dowolnych $a, b, c \in \mathbf{N}$ i jaka jest wtedy złożoność tego algorytmu?
Uwaga! W pierwszej części zadania nie chodzi o to, by dopisać tyle zer aby rozmiar macierzy powiększył się do liczby będącej potęgą 2.
Wskazówka! W drugiej części zadania rozważ najpierw przypadek, gdy a, b i c są potęgami 2. Zastanów się także, co należy zrobić w sytuacji, gdy a, b lub c ma wartość 1.

7. [*] Niech A będzie macierzą o wymiarach $k\,n \times n$ a B macierzą o wymiarach $n \times k\,n$, gdzie $k \in \mathbf{N}$. Jak szybko można obliczyć iloczyn $A \cdot B$ korzystając z algorytmu Strassena? O ile lepszy jest ten wynik od tradycyjnego mnożenia macierzy? Odpowiedz na te same pytania dla iloczynu $B \cdot A$.

8. [***] Niech A będzie kwadratową macierzą trójkątną (górną lub dolną) o rozmiarach $n \times n$, gdzie n jest potęgą 2. Podzielmy macierz A na cztery podmacierze o wymiarach $\frac{n}{2} \times \frac{n}{2}$:
$$\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}$$
Zakładamy, że A_{11}^{-1} istnieje. Niech $\Delta = A_{22} - A_{21}A_{11}^{-1}A_{12}$. Zakładamy, że Δ^{-1} istnieje. Udowodnij, że można wówczas obliczyć macierz odwrotną A^{-1} w następujący sposób:
$$A^{-1} = \begin{bmatrix} A_{11}^{-1} + A_{11}^{-1}A_{12}\Delta^{-1}A_{21}A_{11}^{-1} & -A_{11}^{-1}A_{12}\Delta^{-1} \\ -\Delta^{-1}A_{21}A_{11}^{-1} & \Delta^{-1} \end{bmatrix}$$
Jaka jest złożoność czasowa algorytmu obliczającego macierz odwrotną A^{-1} opisaną powyżej metodą?

9. [**] Oszacuj złożoność czasową przedstawionego na wykładzie algorytmu znajdowania pary najbliżej położonych punktów na płaszczyźnie. W przedstawionym rozwiązaniu, po obliczeniu rozwiązań dla podproblemów i oznaczeniu punktów należących do tzw. *strefy niepewności* (strefa z kandydatami do poprawienia wyniku) sortowane były wszystkie punkty położone w tej strefie względem współrzędnej Y, a potem badane odległości każdego z tych punktów z siedmioma następnymi. Jak poprawić ten algorytm, aby obliczał tylko dwie odległości związane z każdym punktem w strefie niepewności zamiast siedmiu?