ćwiczenia (licencjat wieczorowy): lista zadań nr 11 *18 grudnia 2006 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** $T[0 \ldots n-1]$)
   {
   if $n = 2$ **then** { COMPARE-EXCHANGE $(T[0], T[1])$; **return**; }
   $p_1 \leftarrow 1/4\,n$; $p_2 \leftarrow 2/4\,n$; $p_3 \leftarrow 3/4\,n$;
   STOOGE-SORT-4 $(t[0 \ldots p_2 - 1])$;
   STOOGE-SORT-4 $(t[p_1 \ldots p_3 - 1])$;
   STOOGE-SORT-4 $(t[p_2 \ldots n-1])$;
   STOOGE-SORT-4 $(t[0 \ldots p_2 - 1])$;
   STOOGE-SORT-4 $(t[p_1 \ldots p_3 - 1])$;
   STOOGE-SORT-4 $(t[0 \ldots p_2 - 1])$;
   }

   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 $0 \leq k < 2n$. Opisz algorytm znajdowania k-tej co do wielkości liczby spośród liczb zapisanych w zadanych tablicach. Czas działania twojego algorytmu powinien być rzędu $O(\log n)$. Uzasadnij jego poprawność.

3. [∗] Pomnóż algorytmem opartym na metodzie „dziel i zwyciężaj” dwie liczby dziesiętne 12481632 i 92781243. Przyjmujemy, że liczby jednocyfrowe mnożymy *ad hoc*.

4. [∗∗] Jak uogólnić algorytm mnożenia długich liczb metodą „dziel i zwyciężaj”, aby długości liczb nie musiały być potęgami 2, a także by ich długości mogły być różne.

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

6. [$*$] Użyj algorytmu Strassena do obliczenia następującego iloczynu:

$$\begin{pmatrix} 7 & 5 \\ 3 & 2 \end{pmatrix} \cdot \begin{pmatrix} 4 & 6 \\ 8 & 9 \end{pmatrix}$$

7. [$**$] 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 ?

8. [$**$] 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 $B \cdot A$.

9. [$***$] Macierz kwadratową M rozmiaru $n \times n$ nazywamy *macierzą Toeplitza*, jeśli jej elementy spełniają równanie $M[i, j] = M[i - 1, j - 1]$ dla $1 < i, j \leq n$.

   (a) Zaproponuj reprezentację macierzy Toeplitza, pozwalającą dodawać dwie takie macierze w czasie $O(n)$.

   (b) Podaj algorytm, oparty na metodzie "dziel i zwyciężaj", mnożenia macierzy Toeplitza przez wektor, działający poniżej czasu kwadratowego $o(n^2)$.

   Za operacje dominujące przyjmij operacje arytmetyczne (dodawanie i mnożenie) wykonywane na pojedynczych elementach macierzy/wektora.