ćwiczenia (licencjat wieczorowy): lista zadań nr 4 *31 marca 2009 r.*

ALGORYTMY I STRUKTURY DANYCH

SCALANIE I PODZIAŁ

Instytut Informatyki Uniwersytetu Wrocławskiego *Paweł Rzechonek*

1. [1.5 pkt.] Dane są dwa posortowane ciągi $A = (a_0, a_1, \ldots, a_{m-1})$ i $B = (b_0, b_1, \ldots, b_{n-1})$ 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 $\min\{m, n\}$.

2. [3 pkt.] Dane są dwa posortowane ciągi $A = (a_0, a_1, \ldots, a_{m-1})$ i $B = (b_0, b_1, \ldots, b_{n-1})$ zapisane w jednej $(m + n)$–elementowej tablicy (najpierw ciąg A a potem B). Zaprojektuj rekurencyjny algorytm, który będzie scalał *stabilnie* takie ciągi i nie będzie używał dodatkowych buforów pomodniczych na dane. Czas działania twojego algorytmu powinien być nie gorszy niż $O((m + n) \log(m + n))$.

3. [2 pkt.] Pokaż, jak zaimplementować *sortowanie przez scalanie* dla danych zapisanych w *pliku*. Możesz użyć kilku plików pomocniczych i tylko stałej liczby komórek pamięci operacyjnej.

4. [1 pkt.] Czasem mamy do czynienia z danymi, których wartości często się powtarzają. Pokaż, jak zmodyfikować *algorytm podziału Lomuta*, aby podzielił dane zapisane w tablicy względem piwota p na 3 części: elementy $< p$, elementy $= p$ i elementy $> p$. Czas działania twojego algorytmu powinien być liniowy.

5. [1.5 pkt.] Jak zmodyfikować algorytm *quick–sort*, aby w każdym przypadku głębokość wywołań rekurencyjnych była nie większa niż $\log n$ dla danych rozmiaru n?

   Wskazówka. Jedno z wywołań rekurencyjnych rozwiń w miejscu.

6. [3 pkt.] Pokaż, jak zaimplementować algorytm *quick–sort*, aby działał on *w miejscu* (nie wolno korzystać z rekurencji).