Ćwiczenia: lista zadań nr 9 *29 listopada 2005*

Algorytmy i Struktury Danych

PODZIAŁ, SCALANIE, WYBÓR

Instytut Informatyki Uniwersytetu Wrocławskiego *Paweł Rzechonek*

1. [∗∗] Poniżej przedstawiono procedurę podziału tablicy liczb według pierwszego elementu, której autorem jest *N. Lomuto*. Ta wersja podziału tablicy tworzy dwa obszary: elementy mniejsze lub równe *pivotowi* $A[0 \ldots i-1]$ i elementy większe od *pivota* $A[i \ldots n-1]$.

```
(1)   function Lomuto-partition (number A[0...n-1]) ↦ integer
(2)   {
(3)       integer pivot ← A_0 ;
(4)       integer i ← 1 ;
(5)       for j = 1...n-1 do
(6)           if A_j ≤ pivot then
(7)           {
(8)               A_i ↔ A_j ;
(9)               i++ ;
(10)          }
(11)      return i ;
(12)  }
```

   (a) Uzasadnij poprawność procedury *Lomuto-partition*.

   (b) Oszacuj złożoność tej procedury (ile razy każdy element może być przesuwany w trakcie jej działania).

   (c) Jak poprawić procedurę *Lomuto-partition*, aby dokonywany przez nią podział zawsze był właściwy.

2. [∗∗∗∗] Rozważmy następujący pomysł na trójpodział: z n-elementowej tablicy losowo wybieramy dwa; mniejszy z nich x przenosimy na początek tablicy a większy y na koniec; następnie przesuwamy pozostałe elementy w taki sposób, aby najpierw występowały $\leq x$ (pierwszy blok), potem te które są $> x$ ale $< y$ (drugi blok), a na końcu $\geq y$ (trzeci blok). Napisz imlementację takiego algorytmu (powinna to być funkcja, która po przestawieniu wszystkich elemntów, zwróci parę liczb: pozycje, od których rozpoczynają się drugi i trzeci blok). Jakie jest prawdopodobieństwo, że po trójpodziale żaden z trzech bloków nie będzie

mniejszy niż $\frac{1}{6}n$, jeśli każda para pivotów mogła być wylosowana z jednakowym prawdopodobieństwem?
Uwaga: Możesz założyć, że n jest wielokrotnością 6, oraz że wszystkie elementy w tablicy są różne.

3. W dużej n-elemntowej tablicy T wyróżnione są dwa sąsiednie bloki $U = T[p \dots q-1]$ i $V = T[q \dots r-1]$, gdzie $0 \le p < q < r \le n$. Zadanie polega na tym, aby zamienić ze sobą te bloki nie zmieniając występującego w nich uporządkowania.

   (a) [$*$] Wymyśl algorytm, który będzie działać w miejscu wykonując $\le (|U|+|V|)$ zamian.
   Wskazówka: Zacznij od zamiany pierwszego elementu z u z ostatnim elementem z V.

   (b) [$**$] Wymyśl algorytm, który będzie działać w miejscu wykonując $< (|U| + |V|)$ ale $\ge \max(|U|, |V|)$ zamian.
   Wskazówka: Zacznij od zamiany miejscami mniejszego bloku z sąsiadującym fragmentam bloku drugiego.

4. [$****$] W dużej n-elemntowej tablicy T wyróżnione są dwa sąsiednie bloki $U = T[p \dots q-1]$ i $V = T[q \dots r-1]$, gdzie $0 \le p < q < r \le n$. Tak jak poprzednio, zadanie polega na tym, aby zamienić ze sobą te bloki nie zmieniając występującego w nich uporządkowania. Wymyśl algorytm, który będzie działać w miejscu wykonując tylko $|U|+|V|+\gcd(|U|, |V|)$ przesunięć. Uzasadnij, że $\gcd(|U|, |V|) \le \max(|U|, |V|)$.
Uwaga: Przesunięcie to nie to samo co zamiana — przesunięcie to przepisanie wartości z jednego miejsca do drugiego, zamiana składa się więc z 3 przesunięć.
Wskazówka: Wyciągnij jeden element z tablicy, a na jego miejsce wstaw właściwy, itd. Technikę z *latającą dziurą* opracowali w roku 1981 *K. Dudziński* i *A. Dydek*.

5. [$***$] Na wykładzie został przedstawiony uproszczony algorytm *M. A. Kronroda* z roku 1969 scalania *w miejscu* dwóch posortowanych ciągów z liczbami zapisanych w jednej tablicy. Posortowane ciągi o długościach odpowiednio p i $n-p$, gdzie $0 < p < n$, są umieszczone w n-elementowej tablicy $T[0 \dots n-1]$, w taki sposób że $T_0 \le T_1 \le \dots \le T_{p-1}$ oraz $T_p \le T_{p+1} \le \dots \le T_{n-1}$. Wylicz, ile przestawień wykona ten algorytm w najgorszym przypadku.
Uwaga: Zakładamy, że $\sqrt{n} \in \mathbf{N}$ oraz $\sqrt{n} \mid p$).

6. [$****$] Na wykładzie został przedstawiony uproszczony algorytm *Kronroda* scalania *w miejscu* dwóch posortowanych ciągów z liczbami zapisanych w jednej tablicy. Posortowane ciągi o długościach odpowiednio p i $n-p$, gdzie $0 < p < n$, są umieszczone w n-elementowej tablicy $T[0 \dots n-1]$, w taki sposób że $T_0 \le T_1 \le \dots \le T_{p-1}$ oraz $T_p \le T_{p+1} \le \dots \le T_{n-1}$. W wersji uproszczonej zakładaliśmy, że $\sqrt{n} \in \mathbf{N}$ oraz $\sqrt{n} \mid p$ Uogólnij ten algorytm na przypadek dowolnych wielkości n i p, aby nie pogorszyć liniowej złożoności algorytmu.

7. [$**$] Podaj *stabilne* algorytmy scalania *w miejscu* ciągów binarnych i trynarnych. Twoje algorytmy mają działać w liniowym czasie.
Uwaga: Algorytm operujący na tablicy jest *stabilny*, gdy nie zmienia wzajemnego położenia elementów o takich samych wartościach kluczowych.

8. [$***$] Mamy dostęp do algorytmu *czarna skrzynka*, który w liniowym czasie znajduje medianę nieuporządkowanego ciągu liczb. W jaki sposób, korzystając z *czarnej skrzynki*, wyznaczyć dowolny k-ty co do wielkości element w nieuporządkowanym ciągu liczb. Twój algorytm także powinien działać w czasie liniowym.