Algorytmy i Struktury Danych, 9. ćwiczenia

2019-11-27

Spis treści

1 Rozgłaszanie komunikatów **1**

2 Klasówka 2019 (1), zadanie 3 **2**

3 Klasówka 2019 (1), zadanie 2 **2**

4 Klasówka 2019 (1), zadanie 1 **3**

1 Rozgłaszanie komunikatów

Dane drzewo T, należy obliczyć czas potrzebny na przesłanie komunikatów do wszystkich węzłów drzewa. Przesłanie komunikatu po jednej krawędzi zajmuje 1 jednostkę czasu.
Algorytm $O(n \log n)$:

- jeśli wierzchołek jest liściem to $czas = 0$,
- wpp. rekurencyjnie oblicz czas potrzebny na rozgłoszenie w poddrzewach,
- posortuj malejąco otrzymane czasy: $t_1, \ldots, t_k$
- $czas = max\{i + t_i : 1 \leq i \leq k\}$

Aby otrzymać algorytm $O(n)$ trzeba sprytnie obliczać wartości atrybutu $czas$.

- $Q = \{$ liście $T \}$,
- while $root \notin Q$ do
  - $x = Q.extractMin()$
  - dodaj $x.czas$ do kolejki $parent(x)$,
  - jeśli $parent(x)$ ma już pełną listę poddrzew, to policz $parent(x).czas$ i dodaj $parent(x)$ do kolejki.

Kolejkę Q można zaimplementować w tablicy (i–ty element tablicy zawiera listę wierzchołków o wartości $x.czas = i$). Sumarycznie operacje $extractMin$ zajmą czas $O(n)$. Dodawanie do kolejki zajmuje czas $O(1)$.

2 Klasówka 2019 (1), zadanie 3

Dana jest tablica $a[1..4n]$, w której znajduje się po n rekordów etykietowanych każdą z liczb ze zbioru $\{0, 1, 2, 3\}$.

a) (4 punkty) Zaprojektuj liniowy algorytm sortujący tablicę a względem etykiet rekordów w miejscu.

b) (4 punkty) Podaj algorytm działający w czasie $O(n \log n)$, który sortuje tablicę a względem etykiet rekordów w miejscu i stabilnie (tzn. kolejność rekordów o tej samej etykiecie przed i po sortowaniu jest taka sama).

Rozwiązanie:

Algorytm liniowy (3 krotne rozwiązanie problemu flagi polskiej):

```
foreach x ∈ {0,1,2} do
    j:=1
    for i:=1 to 4n do
        if a[i] <= x then
            if j < i then swap(i, j)
            j:=j+1;
```

Algorytm $O(n \log n)$ (3 krotne rozwiązanie problemu stabilnego sortowania ciągów 0/1)

Algorithm 1: Sort

Data: $a[i..j]$, x
Result: $a[i..j]$ uporządkowana w ten sposób, że wszystkie wartości $\leq x$ występują przed wartościami $> x$

```
if i < j then
    m := ⌊(i + j)/2⌋
    Sort(a[i..m], x)
    Sort(a[m + 1..j], x)
    niech L blok a[i..m] zawierający wartości > x
    niech R blok a[m + 1..j] zawierający wartości ≤ x
    if |L| > 0 and |R| > 0 then
        zamień ze sobą bloki L i R
```

```
foreach x ∈ {0,1,2} do
    Sort(a, x)
```

Powyższy algorytm używa pamięci dodatkowej rzędu $O(\log n)$ ale łatwo zamienić rekurencję na iterację i osiągnąć pamięć $O(1)$.

3 Klasówka 2019 (1), zadanie 2

Rozważamy klasyczne sortowanie przez wstawianie tablicy a[1..n], w porządku niemalejącym, w której zapisano pewną permutację liczb od 1 do n. Zaprojektuj wydajny algorytm, który dla danej zawartości tablicy a wyznaczy najmniejsze

takie i, że podczas sortowania przez wstawianie a[i] jest porównywane najczęściej.

Rozwiązanie:

Element $a[i]$ jest porównany:

- niech $k = |\{j : a[j] > a[i] \text{ oraz} j < i\}|$ z k elementami podczas i–tej fazy sortowania, które zakończyły się zamianą elementów
- niech $m = |\{j : a[j] < a[i] \text{ oraz} j > i\}|$ z m elementami podczas kolejnych faz sortowania, które zakończyły się zamianą elementów
- niech p to liczba elementów z którymi porównywany jest $a[i]$ ale porównanie nie zakończyło się zamianą.

Jak policzyć wartości p:

---

foreach $x \in A$ **do**
  $p_x = 0$
$S := \emptyset$
for $i = 1..n$ **do**
  $x = A[i]$
  $y = \max\{y \; in S : y < x\}$
  if *y jest zdefiniowane* **then**
    $p_x := p_x + 1$
    $p_y := p_y + 1$
  $S = S + \{A[i]\}$

---

4 Klasówka 2019 (1), zadanie 1

a) (2 punkty) Ile jest permutacji liczb od 1 do n o co najwyżej 2 inwersjach?

b) (3 punkty) Zaproponuj optymalny (ze względu na liczbę porównań) algorytm sortujący przez porównania ciągi różnowartościowe o co najwyżej 2 inwersjach.