Spis treści

Rozdział 1

poprawka 2010

1.1. zadanie 1

Rozwiąż poniższe równanie:

$$T(n) = \begin{cases} c & \text{jeśli n} = 1 \\ 2 \cdot T(n-1) + 1 & \text{jeśli } n > 1 \end{cases}$$

Czy funkcja $T(n)$ jest $O(n^{log_2 n})$?

Rozwiązanie równania rekurencyjnego:

$$\begin{aligned} T(n) &= 2 \cdot T(n-1) + 1 \\ &= 2 \cdot (2 \cdot T(n-2) + 1) + 1 \\ &= 2^2 \cdot T(n-2) + 2 + 1 \\ &= 2^3 \cdot T(n-3) + 2^2 + 2 + 1 \\ &... \\ &= 2^n \cdot T(1) + 2^{n-1} + ... + 2^2 + 2 + 1 \\ &= 2^n \cdot c + 2^n - 1 \end{aligned}$$

Pokażemy że $T(n) \notin O(n^{log_2 n})$:

Z powyżej rozwiązanej rekurencji widać że: $T(n) \in \Omega(2^n)$,

$$\exists_{n_0 \in N} \forall_{n > n_0} n^{log_2 n} = \left(2^{log_2 n}\right)^{log_2 n} = 2^{(log_2 n)^2} < 2^n \tag{1.1}$$

Wynika z tego, że $n^{log_2 n}$ nie jest ograniczeniem górnym.

1.2. zadanie 2

Wyznacz z dokładność do Θ (przy jednorotnym kryterium) poniższego fragmentu algorytmu:

suma $\leftarrow 0$
for $i \leftarrow 1$ *to* n **do**

```
    k ← 1
    while k ⩽ i do
      read(x)
      suma ← suma + x · k
      k ← k + k
    end while
  end for
```

$$\sum_{i=0}^{n} log_2 i = log_2 1 + ... + log_2 n = log_2(1 \cdot ... \cdot n) = log_2(n!)$$

Algorytm jest $\Theta(log_2(n!))$

1.3. zadanie 3

Załóżmy, że w definicji drzewa czerwno-czarnego zmienimy warunek mówiący iż dzieci czerwonego ojca są czarne na warunek:
dzieci czerwonego ojca, którego ojciec też jest czerwony są czarne
Określ jak zmieni się (z dokładnością) do stałego czynnika maksymalna wysokość tak zdefiniowanych drzew?

Przy orginalnym założeniu:

$$h \leqslant 2 \cdot log(n-1)$$

Po zamianie założeniam, minimalna liczba czarnych wierzchołków w każdej scieżce od korzenia do liścia, zmieniła się z $\frac{h}{2}$ na $\frac{h}{3}$. Warunek na h przyjmuje teraz postać:

$$h \leqslant 3 \cdot log(n-1)$$

1.4. zadanie 4

Z ilu drzew może składać sie kopiec dwumianowy (wersja eager) zawierający 49 elementów?

Ponieważ jest to kopiec typu eager nie może miec dwóch drzew dwumianowych tego samego stopnia. Aby przekonać się z ilu drzew dwumianowych się on składa, zamieńmy liczbę jego elementów na liczbe o binarną.

$$49_{10} = 32_{10} + 16_{10} + 1_{10} = 2^5_{10} + 2^4_{10} + 2^0_{10} = 110001_2 \tag{1.2}$$

Widźimy, że kopiec ten składa się z trzech drzew: B_5 , B_4 i B_1.

1.5. zadanie 5

Dla której z poniżej podanych struktur danych koszt (najgorszego przypadku) wykonania operacji $find(i)$ sprawdzającej czy klucz i jest pamiętany w struktrze jest $O(logn)$, gdzie n jest rozmiarem struktury?

(a) drzewo binarnych przeszukiwań,

(b) drzewo AVL

(c) kopiec

(d) kopiec dwumianowy

(e) kopiec Fibonacciego

(f) drzewo czerwono-czarne

Dla struktur b, f koszt wykonania operacji $find(i)$ jest $O(logn)$. Dla pozostałych jest on $O(n)$.

1.6. zadanie 6

Podaj przykłady nietrywialnych zastosowań poniższych algorytmow i struktur danych: kopiec, kopiec Fibonacciego, kopiec dwumianowy, sortowanie leksykograficzne ciągów róznej dlugości

	Przykład zastosowania
kopiec	algorytm sortowanie przez kopcowanie
kopiec Fibonacciego	algorytm Dijskry
kopiec dwumianowy	kolejka priorytetowa
sortowanie leksykograficzne ciągów różnej długości	algorytm rozpoznający czy dwa drzewa są izomorficzne

1.7. zadanie 14

W problemie LCS stosowaliśmy redukcję problemu porównując ostatnie litery X i Y. Czy jakies znaczenie ma fakt, że są to ostatnie litery a nie pierwsze?

Nie jest to istotne.

Weźmy sobie ciągi X, Y i niech algorytm $LCS(X, Y)$, obliczający długość najdłuższego podciągu, wykorzystując redukcję problemu porównując ostatnie litery ciagów. Zaóważmy że algorytm $LCS'(X, Y) = LCS(X^R, Y^R)$ również poprawnie oblicza długośc najdłuższego podciągu ciagów X, Y, ale można powiedzieć że wykorzystuje on redukcję problemu porównując pierwsze litery X, Y.

Bibliografia

[1] test reference