Algorytmy i struktury danych

Lista zadań 4

Zadanie 1

Skorzystaj z metody rekurencji uniwersalnej i podaj dokładne asymptotyczne oszacowania dla następujących rekurencji:

(a) $n^{1/2} = \Theta(n^{log_4 2}) \implies T(n) = 2T(n/4) + \sqrt{n} = \Theta(n^{1/2} logn)$

(b) $n = \Omega(n^{log_4 3+\epsilon})$ dla $\epsilon < 0.2 \implies T(n) = 3T(n/4) + n = \Theta(n)$

(c) $n^{3/2} = \Theta(n^{log_4 8}) \implies T(n) = 8T(n/4) + n\sqrt{n} = \Theta(n^{3/2} logn)$

(d)

$$T(n) = 2T\left(n^{\frac{1}{2}}\right) + 1$$
$$m = \log n,\, U(m) = T\left(e^{m}\right) = T\left(e^{\log n}\right) = T(n)$$
$$U(m) = 2T\left(e^{\log n^{\frac{1}{2}}}\right) + 1 = 2T\left(e^{\frac{1}{2}\log n}\right) + 1 = 2T\left(e^{\frac{m}{2}}\right) + 1 = 2U\left(\frac{m}{2}\right) + 1$$
$$1 = O\left(m^{log_2 2-\epsilon}\right) \text{ dla } \epsilon \leq 1 \implies U(m) = \Theta(m)$$
$$T(n) = U(m) = U(\log n) = \Theta(m) = \Theta(\log n)$$

Zadanie 2

Czas działania algorytmu A opisany jest przez rekurencję $T(n) = 7T(n/2) + n^2$. Algorytm konkurencyjny A' ma czas działania $T'(n) = aT'(n/4) + n^2$. Jaka jest największa liczba całkowita a, przy której A' jest asymptotycznie szybszy niż A?

$$n^2 = O\left(n^{log_2 7-\epsilon}\right) \text{ dla } \epsilon \leq 0.80 \implies T(n) = 7T(n/2) + n^2 = \Theta\left(n^{log_2 7}\right)$$
$$n^2 = O\left(n^{log_4 a-\epsilon}\right) \implies T'(n) = aT'(n/4) + n^2 = \Theta\left(n^{log_4 a}\right)$$
$$n^{log_4 a} < n^{log_2 7}$$
$$n^{\frac{1}{2}log_2 a} < n^{log_2 7}$$

$$T'(n) = \begin{cases} \Theta\left(n^2\right) & \text{dla } a \in [1, 15] \\ \Theta\left(n^2 logn\right) & \text{dla } a = 16 \\ \Theta\left(n^{log_4 a}\right) & \text{dla } a \in [17, \infty) \\ \Theta\left(n^{log_4 49}\right) = \Theta\left(n^{log_2 7}\right) & \text{dla } a = 49 \end{cases}$$

Zadanie 3

Rozważmy warunek regularności $af(n/b) \leq cf(n)$ dla pewnej stałej $c \leq 1$, który jest częścią przypadku 3 twierdzenia o rekurencji uniwersalnej. Podaj przykład prostej funkcji $f(n)$, które spełnia wszystkie warunki twierdzenia o rekurencji uniwersalnej z wyjątkiem warunku regularności.

$$af(n/b) \leq cf(n),\ a \geq 1,\ b > 1,\ c \leq 1$$
$$T(n) = T(n/2) + \sin n$$
$$\sin\left(\frac{n}{2}\right) \leq \sin n$$
$$\text{dla } n = \frac{7\pi}{2},\ \sin\left(\frac{7\pi}{4}\right) \leq \sin\frac{7\pi}{2} \implies \leq -1$$

Zadanie 4

Zasymuluj działanie polifazowego mergesorta dla tablicy:

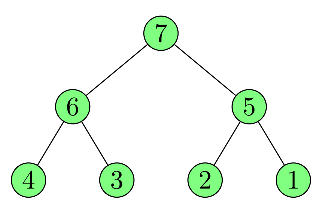
`{9,22,6,19,21,14,10,17,3,5,60,30,29,1,8,7,6,15,12}`.

W sortowaniu polifazowym na każdym etapie sortowania scala się sąsiadujące podciągi rosnące, to znaczy: w pierwszym przebiegu `{9,22}` z `{6,19,21}`, `{14}` z `{10,17}` itd..

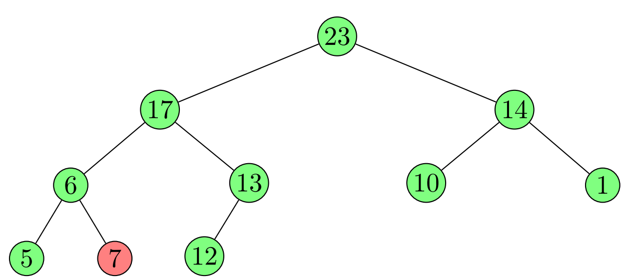
```
{9,22|6,19,21|14|10,17|3,5,60|30|29|1,8|7|6,15|12}
{6,9,19,21,22|10,14,17|3,5,30,60|1,8,29|6,7,15|12}
{6,9,10,14,17,19,21,22|1,3,5,8,29,30,60|6,7,12,15}
{1,3,5,6,8,9,10,14,17,19,21,22,29,30,60|6,7,12,15}
{1,3,5,6,6,7,8,9,10,12,14,15,17,19,21,22,29,30,60}
```

Zadanie 5

(a) Czy tablica posortowana malejąco jest kopcem?



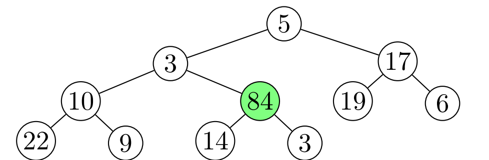
(b) Czy ciąg `{23,17,14,6,13,10,1,5,7,12}` jest kopcem?



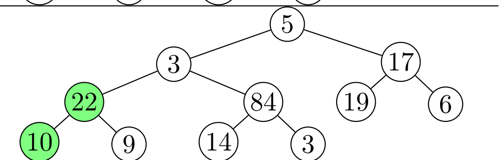
Zadanie 6

Zilustruj działanie procedury buildheap dla ciągu `{5,3,17,10,84,19,6,22,9,14,3}`. Narysuj na kartce wygląd tablicy i kopca po każdym wywołaniu procedury przesiej.

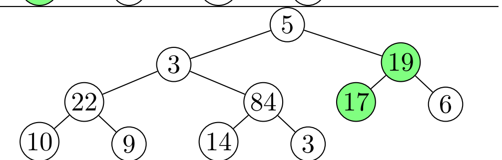
`{5,3,17,10,84,19,6,22,9,14,3}`



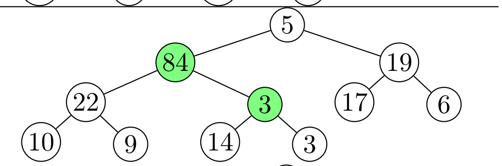
`{5,3,17,22,84,19,6,10,9,14,3}`
`{5,3,17,22,84,19,6,10,9,14,3}`

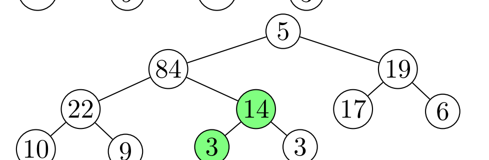


`{5,3,19,22,84,17,6,10,9,14,3}`
`{5,3,19,22,84,17,6,10,9,14,3}`

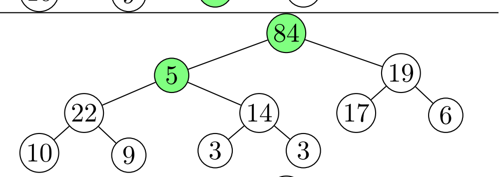


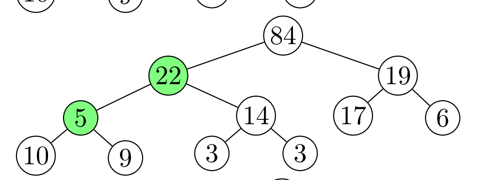
`{5,84,19,22,3,17,6,10,9,14,3}`
`{5,84,19,22,14,17,6,10,9,3,3}`
`{5,84,19,22,14,17,6,10,9,3,3}`

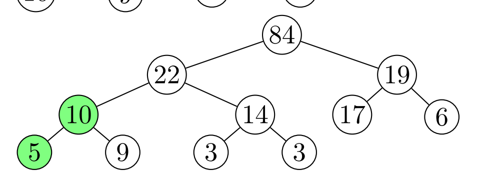




`{84,5,19,22,14,17,6,10,9,3,3}`
`{84,22,19,5,14,17,6,10,9,3,3}`
`{84,22,19,10,14,17,6,5,9,3,3}`
`{84,22,19,10,14,17,6,5,9,3,3}`







Zadanie 8

Udowodnij, że wysokość kopca n-elementowego wynosi $\lfloor \log_2 n \rfloor + 1$.

$$2^k \leq n \leq 2^{k+1} - 1 \implies k \leq \log_2 n \leq k + 1$$

Zadanie 11

Niech F_n oznacza ilość różnych kształtów drzew binarnych o n węzłach. Rysując drzewa, łatwo sprawdzić, że $F_0 = 1, F_1 = 1, F_2 = 2, F_3 = 5$, itd. Nie korzystając z internetu:

(a) Znajdź wzór wyrażający F_n przez $F_0, F_1, F_2, \ldots, F_{n-1}$ dla $n = 2, 3, 4$ a potem ogólnie.

(b) Zaprojektuj (na kartce) procedurę, która oblicza kolejne wyrazy ciągu F_n, zapisuje je w tablicy i korzysta z nich przy obliczaniu następnych wyrazów.

(c) Przeanalizuj ile mnożeń trzeba wykonać, by obliczyć wyrazy od F_1 do F_n. Czy da się ją zapisać w postaci $O(n^k)$ dla pewnego k?

(d) Jaka byłaby złożoność algorytmu rekurencyjnego, który nie korzysta z wartości zapisanych w tablicy, tylko oblicza je ponownie. Czy da się ją zapisać jako $O(n^k)$?