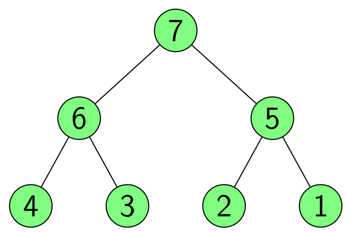
Algorytmy i struktury danych (Lista 4)

Zadanie 7 Metodą jak na wykładzie, udowodnij, że procedura `buildHeap` działa w czasie O(n).

Zadanie 8 Udowodnij, że wysokość kopca n-elementowego wynosi $\lfloor log2n \rfloor + 1$



i = ostatnie dziecko z dziećmi = $n/2 - 1$

$$i = 0$$

$$2(2(2 * 0 + 1) + 1) + 1 ... < n$$

Maksymalna ilość węzłów w kopcu o wysokości h:

$$n(h) = 2^h - 1$$

$$n(h-1) = 2^{h-1} - 1$$

Minimalna ilość węzłów w kopcu o wysokości h:

$$n(h-1) + 1 = 2^{h-1} - 1 + 1 = 2^{h-1}$$

Ilość węzłów w kopcu o wysokości h:

$$2^{h-1} \leq n < 2^h$$

$$h - 1 \leq \log_2 n < h$$

$$h \leq \log_2 n + 1 < h + 1$$

$$\lfloor \log_2 n \rfloor + 1$$

Zadanie 9 (2 pkt) Podaj ideę algorytmu, jak przy pomocy struktury kopca, złączyć k posortowanych list jednokierunkowych o łącznej ilości elementów n, w jedną posortowaną listę, za używając nie więcej niż $3n\ log_2 k$ porównań

k list po $\frac{n}{k}$ elementów

1. kopiec z pierwszych elementów list
2. korzeń wraca do listy i jest zastępowany kolejnym elementem z rodzimej listy dla korzenia
3. przesiewamy -> $2log_2 k$ porównań dla jednego elementu

4. n elementów więc w sumie $2n\ log_2 k$
5. wyciągamy elementy z kopca

Zadanie 10 W pliku spis.txt umieszczone są w przypadkowej kolejności wszystkie liczby całkowite od 1 do n za wyjątkiem jednej (n jest bardzo duże). Jak wyliczyć brakującą liczbę w czasie liniowym nie wykorzystując dodatkowej pamięci plikowej ani RAM za wyjątkiem kilku zmiennych typu int? Wskazówka: w C++ działania + - * na liczbach całkowitych obywają się modulo 2^{32}

Zadanie 11 Niech Fn oznacza ilość różnych kształtów drzew binarnych o n węzłach. Rysując drzewa, łatwo sprawdzić, że `F0 = 1, F1 = 1, F2 = 2, F3 = 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.
$F_0 = 1, F_1 = 1$
$F_2 = 2 = F_0F_1 + F_1F_0$
$F_3 = 5 = F_0F_2 + F_1F_1 + F_2F_0$
$F_4 = 14 = F_0F_3 + F_1F_2 + F_2F_1 + F_3F_0$
$F_n = \sum_{i=1}^{n} F_{i-1}F_{n-i}$
Wymnażamy prawą stronę danego węzła przez lewą stronę a potem na odwrót co po zsumowaniu daje na

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

```
F[0] = 1
F[1] = 1
for i = 2 to n:
    F[i] = 0
    for j = 1 to i:
        F[i] += F[j-1] * F[i-j]
```

(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?

$$\text{Ilośc mnożeń: } \sum_{i=1}^{n} i = \frac{n(1+n)}{2} = O(n^2)$$

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

$$\sum_{k=1}^{n} = \frac{k(1+k)}{2} = \frac{n(n+1)(n+2)}{6} = O(n^3)$$

Zadanie 12 Wykonaj zadanie 11 (a) (b) (c) dla drzew trynarnych (dzieci: left, down, right).