

Algorytmy i Struktury Danych

Wojciech Typer

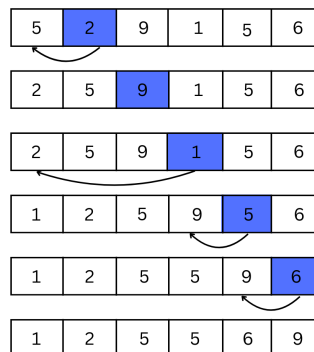
Algorithm 1 Insertion Sort

```
1: procedure INSERTIONSORT( $A, n$ )  
2:   for  $i = 1$  to  $n - 1$  do  
3:      $key = A[i]$   
4:      $j = i - 1$   
5:     while  $j \geq 0$  and  $A[j] > key$  do  
6:        $A[j + 1] = A[j]$   
7:        $j = j - 1$   
8:     end while  
9:      $A[j + 1] = key$   
10:  end for  
11: end procedure
```

Złożoność czasowa: $O(n^2)$

Best case: w najlepszym przypadku złożoność czasowa będzie wynosić $O(n)$

Złożoność pamięciowa: $O(1)$



Algorithm 2 Merge Sort

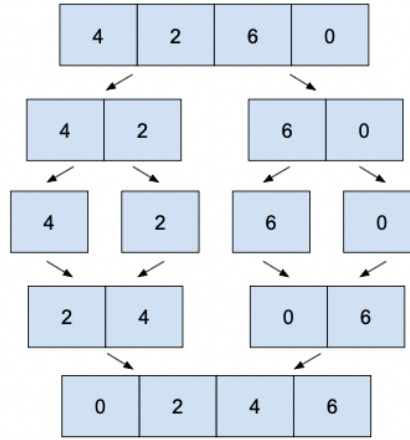
```
1: procedure MERGESORT( $A, 1, n$ )
2:   if  $|A[1..n]| == 1$  then
3:     return  $A[1..n]$ 
4:   else
5:      $B = \text{MergeSort}(A, 1, \lfloor n/2 \rfloor)$ 
6:      $C = \text{MergeSort}(A, \lfloor n/2 \rfloor, n)$ 
7:     return  $\text{Merge}(B, C)$ 
8:   end if
9: end procedure
```

Algorithm 3 Merge

```
1: procedure MERGE( $X[1..k], Y[1..n]$ )
2:   if  $X = \emptyset$  then
3:     return  $Y$ 
4:   else if  $Y = \emptyset$  then
5:     return  $X$ 
6:   else if  $X[1] \leq Y[1]$  then
7:     return  $[X[1]] \times \text{Merge}(X[2..k], Y[1..n])$ 
8:   else
9:     return  $[Y[1]] \times \text{Merge}(X[1..k], Y[2..n])$ 
10:  end if
11: end procedure
```

Złożoność czasowa Merge Sort: $O(n \log n)$

Złożoność pamięciowa Merge Sort: $O(n)$



Istnieje również iteracyjna wersja algorytmu Merge, sort, która została przedstawiona poniżej w postaci pseudokodu.

Algorithm 4 IterativeMergeSort

```

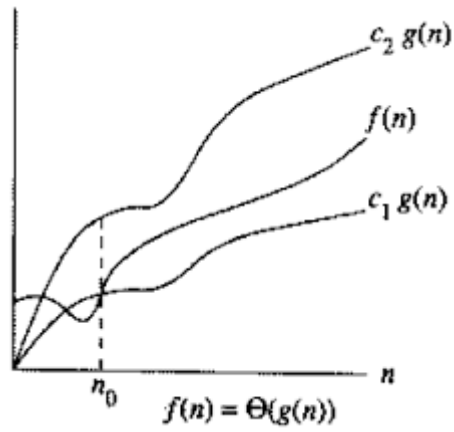
1: procedure ITERATIVEMERGESORT( $A[1..n]$ )
2:   for  $size = 1$  to  $n - 1$  by  $size \times 2$  do
3:     for  $left = 0$  to  $n - 1$  by  $2 \times size$  do
4:        $mid \leftarrow \min(left + size - 1, n - 1)$ 
5:        $right \leftarrow \min(left + 2 \times size - 1, n - 1)$ 
6:       MERGE( $A$ ,  $left$ ,  $mid$ ,  $right$ )
7:     end for
8:   end for
9: end procedure

```

Złożoność czasowa Iterative Merge Sort: $O(n \log n)$ - dzieje się tak, ponieważ $size$ jest podwajany o 2 w każdej iteracji, więc potrzebujemy około $\log_2 n$ iteracji, a w każdej z nich wykonujemy $O(n)$ operacji.

Złożoność pamięciowa Iterative Merge Sort: $O(n)$

Notacja asymptotyczna $O: f(n) = O(g(n)) \rightarrow (\exists c > 0)(\exists n_0 \in N) : \forall n \geq n_0 \rightarrow 0 \leq f(n) \leq c \cdot g(n)$



$$f(n) = O(g(n)) \rightarrow \lim_{n \rightarrow \infty} \frac{|f(n)|}{|g(n)|} < \infty$$

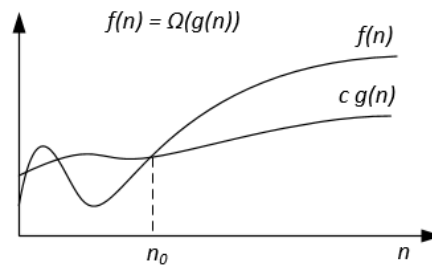
Notacja asymptotyczna - własności

$$\text{a) } f(n) = n^3 + O(n^2) \rightarrow (\exists h(n) = O(n^2))(f(n) = n^3 + h(n))$$

$$\text{b) } n^2 + O(n) = O(n^2) \rightarrow (\forall f(n) = O(n))(\exists h(n) = O(n^2))(n^2 + f(n) - h(n))$$

Notacja Ω

$$f(n) = \Omega(g(n)) \rightarrow (\exists c > 0)(\exists n_0 \in N)(\forall n \geq n_0)(c * g(n) \leq |f(n)|)$$



Notacja Ω - własności

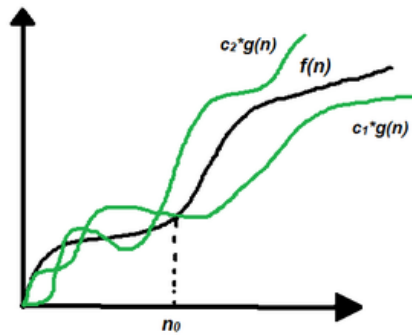
a) $n^3 = \Omega(2n^2)$

b) $n = \Omega(\log(n))$

c) $2n^2 = \Omega(n^2)$

Notacja Θ

$$f(n) = \Theta(g(n)) \rightarrow (\exists c_1, c_2 > 0)(\exists n_0 \in N)(\forall n \geq n_0)(c_1 g(n) \leq |f(n)| \leq c_2 g(n))$$



$$\Theta(f(n)) = O(f(n)) \cap \Omega(f(n))$$

Notacja o - małe

$$f(n) = o(g(n)) \rightarrow (\forall c > 0)(\exists n_0 \in N)(\forall n \geq n_0)(|f(n)| < c * |g(n)|)$$

Notacja o - małe - przykłady

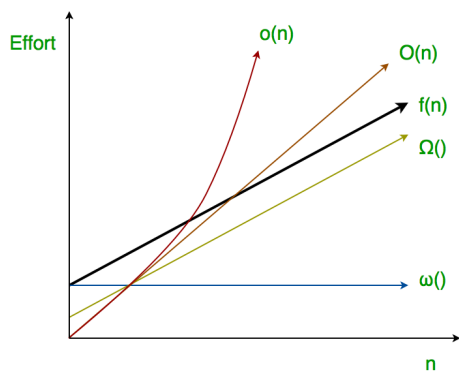
a) $117n \log(n) = o(n^2)$

b) $n^2 = o(n^3)$

Notacja ω

$$f(n) = \omega(g(n)) \rightarrow (\forall c > 0)(\exists n_0 \in N)(\forall n \geq n_0)(|f(n)| > c * |g(n)|)$$

$$\lim_{n \rightarrow \infty} \frac{|f(n)|}{|g(n)|} = \infty$$



Rekurencje

a) Metoda podstawiania (metoda dowodzenia indukcyjnego)

1. Zgadnij odpowiedź (bez stałych)
2. Sprawdź przez indukcję, czy dobrze zgadliśmy
3. Znajdź stałe

Przykład 1:

$$T(n) = 4T\left(\frac{n}{2}\right) + n$$

Pierwszy strzał: $T(n) = O(n^3)$

Cel: pokazać, że $(\exists c > 0) T(n) \leq c * n^3$

Krok początkowy: $T(1) = \Theta(1) = c * 1^3 = c$

Krok indukcyjny: zał. że, $(\forall (k < n))(T(k) \leq c * k^3) =$

$$\text{Dowód: } T(n) = 4T\left(\frac{n}{2}\right) + n \leq 4c * \left(\frac{n}{2}\right)^3 + n = \frac{1}{2}cn^3 + n =$$

$$= cn^3 - \frac{1}{2}cn^3 + n = cn^3 - \left(\frac{1}{2}cn^3 - n\right) \leq cn^3$$

Pokazaliśmy, że $T(n) = O(n^2)$

Spróbujmy wzmocnić zał. indukcyjne: $T(n) \leq c_1n^2 - c_2n$

$$T(n) \leq 4T\left(\frac{n}{2}\right) + n \leq 4\left(c_1\left(\frac{n}{2}\right)^2 - c_2\frac{n}{2}\right) + n =$$

$$= c_1n^2 - 2c_2n + n = c_1n^2 - (2c_2 - 1)n \leq c_1n^2 - c_2n$$

Musimy dobrać takie c_1 i c_2 , aby $2c_1 \geq c_2$

Wówczas otrzymamy $T(1) = O(1) \leq c_11^2 - c_21$

Przykład 2:

$$T(n) = 2T(\sqrt{n}) + \log(n)$$

Założmy, że n jest potęgą dwójki $n = 2^m \rightarrow m = \log(n)$

$$T(2^m) = 2T(2^{m/2}) + m$$

oznaczmy $T(2^m) = S(m)$

$$T(2^m) = 2T(2^{m/2}) + m \rightarrow 2S(m/2) + m$$

$$S(m) = O(m \log(m))$$

$$T(n) = O(\log(n) \log(\log(n))) \text{ (formalnie powinniśmy to udowodnić)}$$