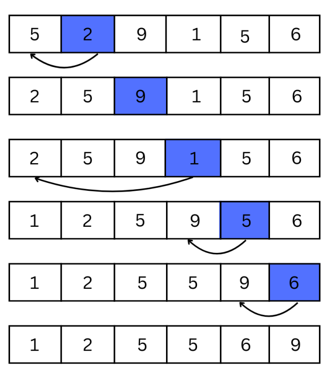
Algorytmy i Struktury Danych

Wojciech Typer

Algorithm 1 Insertion Sort

```
procedure InsertionSort(A, n)
    for i = 1 to n − 1 do
        key = A[i]
        j = i − 1
        while j ≥ 0 and A[j] > key do
            A[j + 1] = A[j]
            j = j − 1
        end while
        A[j + 1] = key
    end for
end procedure
```

Złożoność czasowa: $O(n^2)$
Best case: w najlepszym przypadku złożoność czasowa będzie wynosić $O(n)$
Złożoność pamięciowa: $O(1)$



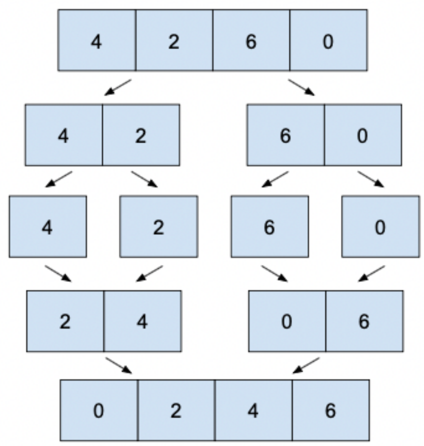
Algorithm 2 Merge Sort

procedure MergeSort(A, 1, n)
  if |A[1..n]| == 1 **then**
    return A[1..n]
  else
    $B = \text{MergeSort}(A, 1, \lfloor n/2 \rfloor)$
    $C = \text{MergeSort}(A, \lfloor n/2 \rfloor, n)$
    return Merge(B, C)
  end if
end procedure

Algorithm 3 Merge

procedure Merge(X[1..k], Y[1..n])
  if $X = \emptyset$ **then**
    return Y
  else if $Y = \emptyset$ **then**
    return X
  else if $X[1] \leq Y[1]$ **then**
    return $[X[1]] \times \text{Merge}(X[2..k], Y[1..n])$
  else
    return $[Y[1]] \times \text{Merge}(X[1..k], Y[2..n])$
  end if
end procedure

Złożoność czesowa 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.

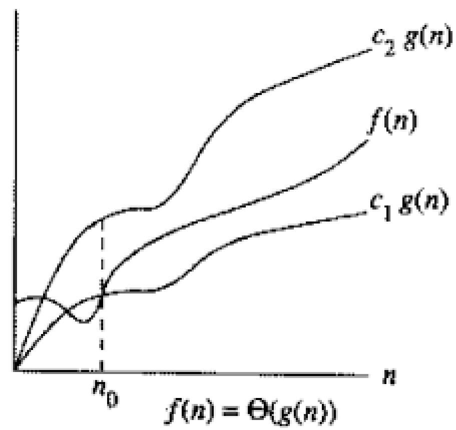
Algorithm 4 IterativeMergeSort

procedure ITERATIVEMERGESORT(A[1..n])
    for $size = 1$ **to** $n - 1$ **by** $size \times 2$ **do**
        for $left = 0$ **to** $n - 1$ **by** $2 \times size$ **do**
            $mid \leftarrow \min(left + size - 1, n - 1)$
            $right \leftarrow \min(left + 2 \times size - 1, n - 1)$
            MERGE(A, left, mid, right)
        end for
    end for
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)) $\rightarrow (\exists c > 0)(\exists n_0 \in N) : \forall n \geq n_0 \rightarrow 0 \leq f(n) \leq c \cdot g(n)$



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

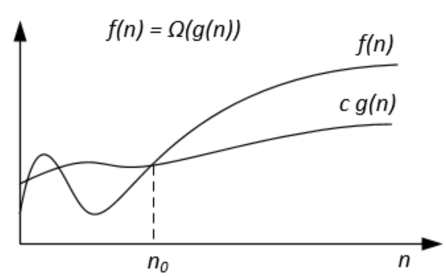
Notacja asymptotyczna - własności

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

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

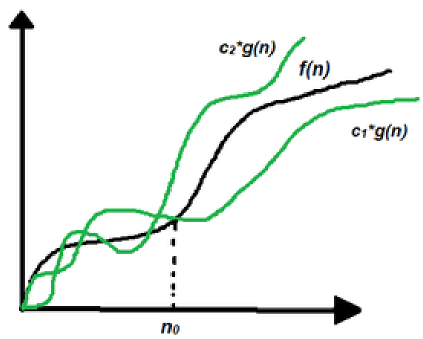
a) $n^3 = \Omega(2n^2)$

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

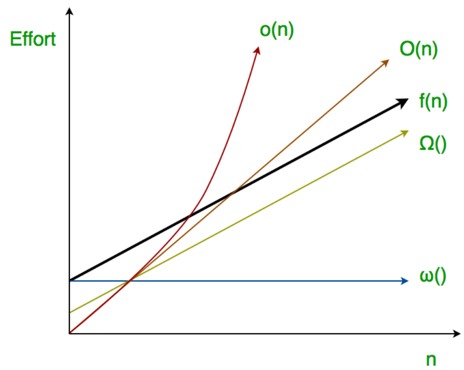
a) $117nlog(n) = o(n^2)$

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:

$T(n) = 4T(\frac{n}{2}) + n$
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, $(\forall_(k < n))(T(k) \leq c * k^3) =$
Dowód: $T(n) = 4T(\frac{n}{2}) + n \leq 4c * (\frac{n}{2})^3 + n = \frac{1}{2}cn^3 + n =$
$= cn^3 - \frac{1}{2}cn^3 + n = cn^3 - (\frac{1}{2}cn^3 - n) \leq cn^3$
Pokazaliśmy, że $T(n) = O(n^2)$

Spróbujmy wzmocnić zał. indukcyjne: $T(n) \leq c_1 n^2 - c_2 n$
$T(n) \leq 4T(\frac{n}{2}) + n \leq 4(c_1(\frac{n}{2})^2 - c_2\frac{n}{2}) + n =$
$= c_1 n^2 - 2c_2 n + n = c_1 n^2 - (2c_2 - 1)n \leq c_1 n^2 - c_2 n$
Musimy dobrać takie $c_1 i c_2$, aby $2c_1 \geq c_2$
Wówczas otrzymamy $T(1) = O(1) \leq c_1 1^2 - c_2 1$

Przykład 2:

$T(n) = 2T(\sqrt{n}) + log(n)$
Załóżmy, że n jest potęgą dwójki $n = 2^m \to m = log(n)$
$T(2^m) = 2T(2^{m/2}) + m$
oznaczmy $T(2^m) = S(m)$
$T(2^m) = 2T(2^{m/2}) + m \to 2S(m/2) + m$
$S(m) = O(mlog(m))$
$T(n) = O(log(n)log(log(n)))$ (formalnie powinniśmy to udowodnić)

Drzewo rekursji

Przykład : $T(n) = T(\frac{n}{2}) + T(\frac{n}{4}) + n^2$