

Algorytmy i struktury danych - Złożoność algorytmów

Marcin Żurowski

19 października 2025

Problem algorytmiczny

- **Wejście:** $a \in \mathbb{R}, n \in \mathbb{N}$
- **Pytanie:** Jaka jest najmniejsza liczba $b \in \mathbb{R}$ spełniająca równość $a^n = b$

Problem algorytmiczny

- **Wejście:** $a \in \mathbb{R}, n \in \mathbb{N}$
- **Pytanie:** Jaka jest najmniejsza liczba $b \in \mathbb{R}$ spełniająca równość $a^n = b$

Potęga iteracyjnie

```
POTEGA(a, n)
```

```
  a ∈ ℝ
```

```
  n ∈ ℕ
```

```
  pot = 1
```

```
  for i = 1 to n
```

```
    pot = pot * a
```

```
  return pot
```

Potęga binarnie

```
POTEGA(a, n)
```

```
  a ∈ ℝ
```

```
  n ∈ ℕ
```

```
  i = n
```

```
  pot = 1
```

```
  pod = a
```

```
  while i ≠ 0
```

```
    if i MOD 2 ≠ 0
```

```
      pot = pot * pod
```

```
      pod = pod * pod
```

```
      i = i DIV 2
```

```
  return pot
```

Potęga rekurencyjnie

```
POTEGA(a, n)
```

```
  a ∈ ℝ
```

```
  n ∈ ℕ
```

```
  if n = 0
```

```
    return 1
```

```
  else
```

```
    return a * POTEGA(a, n - 1)
```

Cel obliczania złożoności algorytmu

- Który algorytm jest najlepszy?
- **Złożoność algorytmu** ilość zasobów (czasu, pamięci) potrzebnych do rozwiązania problemu obliczeniowego za pomocą tego algorytmu

Cel obliczania złożoności algorytmu

- Który algorytm jest najlepszy?
- **Złożoność algorytmu** ilość zasobów (czasu, pamięci) potrzebnych do rozwiązania problemu obliczeniowego za pomocą tego algorytmu

Złożoność czasowa

- **Złożoność czasowa algorytmu** określa czas jego działania.
- Jednostką złożoności czasowej jest wykonanie jednej operacji dominującej.
- Operacje dominujące należy wybrać w taki sposób, żeby ich liczba faktycznie decydowała o czasie działania algorytmu. Można przyjąć, że wszystkie operacje są dominujące.

Złożoność czasowa

- **Złożoność czasowa algorytmu** określa czas jego działania.
- **Jednostką złożoności czasowej** jest wykonanie jednej operacji dominującej.
- Operacje dominujące należy wybrać w taki sposób, żeby ich liczba faktycznie decydowała o czasie działania algorytmu. Można przyjąć, że wszystkie operacje są dominujące.

Złożoność czasowa

- **Złożoność czasowa algorytmu** określa czas jego działania.
- **Jednostką złożoności czasowej** jest wykonanie jednej operacji dominującej.
- Operacje dominujące należy wybrać w taki sposób, żeby ich liczba faktycznie decydowała o czasie działania algorytmu. Można przyjąć, że wszystkie operacje są dominujące.

Złożoność pamięciowa

- **Złożoność pamięciowa algorytmu** określa rozmiar pamięci zużywanej przez ten algorytm.
- **Jednostką złożoności pamięciowej** jest słowo pamięci komputera.

Złożoność pamięciowa

- **Złożoność pamięciowa algorytmu** określa rozmiar pamięci zużywanej przez ten algorytm.
- **Jednostką złożoności pamięciowej** jest słowo pamięci komputera.

Metody obliczania złożoności czasowej

- stoper
- liczba taktów procesora
- narzędzia matematyczne

Metody obliczania złożoności czasowej

- stoper
- liczba taktów procesora
- narzędzia matematyczne

Metody obliczania złożoności czasowej

- stoper
- liczba taktów procesora
- narzędzia matematyczne

Metody obliczania złożoności pamięciowej

- wielkość pamięci operacyjnej
- narzędzia matematyczne

Metody obliczania złożoności pamięciowej

- wielkość pamięci operacyjnej
- narzędzia matematyczne

Narzędzia matematyczne

- Złożoność algorytmu przedstawiamy za pomocą funkcji zależnej od rozmiaru danych wejściowych określającej ilość potrzebnego zasobu
- **Notacja asymptotyczna** narzędzie matematyczne służące do opisu złożoności asymptotycznej

Narzędzia matematyczne

- Złożoność algorytmu przedstawiamy za pomocą funkcji zależnej od rozmiaru danych wejściowych określającej ilość potrzebnego zasobu
- **Notacja asymptotyczna** narzędzie matematyczne służące do opisu złożoności asymptotycznej

Notacja O

$$O(g(n)) = \{f(n) : \exists_{c>0} \exists_{n_0 \in \mathbb{N}} \forall_{n \geq n_0} 0 \leq f(n) \leq cg(n)\}$$

Przez $O(g(n))$ oznaczamy zbiór wszystkich takich funkcji $f(n)$, dla których istnieją dodatnie stałe c , n_0 , takie że dla wszystkich $n \geq n_0$

$$0 \leq f(n) \leq cg(n)$$

Jeśli $f(n) = O(g(n))$ to mówimy, że $g(n)$ jest asymptotycznym ograniczeniem górnym funkcji $f(n)$.

Notacja O

$$O(g(n)) = \{f(n) : \exists_{c>0} \exists_{n_0 \in \mathbb{N}} \forall_{n \geq n_0} 0 \leq f(n) \leq cg(n)\}$$

Przez $O(g(n))$ oznaczamy zbiór wszystkich takich funkcji $f(n)$, dla których istnieją dodatnie stałe c , n_0 , takie że dla wszystkich $n \geq n_0$

$$0 \leq f(n) \leq cg(n)$$

Jeśli $f(n) = O(g(n))$ to mówimy, że $g(n)$ jest asymptotycznym ograniczeniem górnym funkcji $f(n)$.

Notacja O

$$O(g(n)) = \{f(n) : \exists_{c>0} \exists_{n_0 \in \mathbb{N}} \forall_{n \geq n_0} 0 \leq f(n) \leq cg(n)\}$$

Przez $O(g(n))$ oznaczamy zbiór wszystkich takich funkcji $f(n)$, dla których istnieją dodatnie stałe c , n_0 , takie że dla wszystkich $n \geq n_0$

$$0 \leq f(n) \leq cg(n)$$

Jeśli $f(n) = O(g(n))$ to mówimy, że $g(n)$ jest asymptotycznym ograniczeniem górnym funkcji $f(n)$.

Notacja Ω

$$\Omega(g(n)) = \{f(n) : \exists c > 0 \exists n_0 \in \mathbb{N} \forall n \geq n_0 0 \leq cg(n) \leq f(n)\}$$

Przez $\Omega(g(n))$ oznaczamy zbiór wszystkich takich funkcji $f(n)$, dla których istnieją dodatnie stałe c , n_0 , takie że dla wszystkich $n \geq n_0$

$$0 \leq cg(n) \leq f(n)$$

Jeśli $f(n) \in \Omega(g(n))$ to mówimy, że $g(n)$ jest asymptotycznym ograniczeniem dolnym funkcji $f(n)$.

Notacja Ω

$$\Omega(g(n)) = \{f(n) : \exists_{c>0} \exists_{n_0 \in \mathbb{N}} \forall_{n \geq n_0} 0 \leq cg(n) \leq f(n)\}$$

Przez $\Omega(g(n))$ oznaczamy zbiór wszystkich takich funkcji $f(n)$, dla których istnieją dodatnie stałe c , n_0 , takie że dla wszystkich $n \geq n_0$

$$0 \leq cg(n) \leq f(n)$$

Jeśli $f(n) = \Omega(g(n))$ to mówimy, że $g(n)$ jest asymptotycznym ograniczeniem dolnym funkcji $f(n)$.

Notacja Ω

$$\Omega(g(n)) = \{f(n) : \exists_{c>0} \exists_{n_0 \in \mathbb{N}} \forall_{n \geq n_0} 0 \leq cg(n) \leq f(n)\}$$

Przez $\Omega(g(n))$ oznaczamy zbiór wszystkich takich funkcji $f(n)$, dla których istnieją dodatnie stałe c , n_0 , takie że dla wszystkich $n \geq n_0$

$$0 \leq cg(n) \leq f(n)$$

Jeśli $f(n) = \Omega(g(n))$ to mówimy, że $g(n)$ jest asymptotycznym ograniczeniem dolnym funkcji $f(n)$.

Notacja Θ

$$\Theta(g(n)) = \{f(n) : \exists_{c_1 > 0} \exists_{c_2 > 0} \exists_{n_0 \in \mathbb{N}} \forall_{n \geq n_0} 0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n)\}$$

Przez $\Theta(g(n))$ oznaczamy zbiór wszystkich takich funkcji $f(n)$, dla których istnieją dodatnie stałe c_1 , c_2 , n_0 , takie że dla wszystkich $n \geq n_0$

$$0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n)$$

Jeśli $f(n) = \Theta(g(n))$ to mówimy, że $g(n)$ jest asymptotycznie dokładnym oszacowaniem funkcji $f(n)$.

$$f(n) = \Theta(g(n)) \iff f(n) = O(g(n)) \wedge f(n) = \Omega(g(n))$$

Notacja Θ

$$\Theta(g(n)) = \{f(n) : \exists_{c_1 > 0} \exists_{c_2 > 0} \exists_{n_0 \in \mathbb{N}} \forall_{n \geq n_0} 0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n)\}$$

Przez $\Theta(g(n))$ oznaczamy zbiór wszystkich takich funkcji $f(n)$, dla których istnieją dodatnie stałe c_1 , c_2 , n_0 , takie że dla wszystkich $n \geq n_0$

$$0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n)$$

Jeśli $f(n) = \Theta(g(n))$ to mówimy, że $g(n)$ jest asymptotycznie dokładnym oszacowaniem funkcji $f(n)$.

$$f(n) = \Theta(g(n)) \iff f(n) = O(g(n)) \wedge f(n) = \Omega(g(n))$$

Notacja Θ

$$\Theta(g(n)) = \{f(n) : \exists_{c_1 > 0} \exists_{c_2 > 0} \exists_{n_0 \in \mathbb{N}} \forall_{n \geq n_0} 0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n)\}$$

Przez $\Theta(g(n))$ oznaczamy zbiór wszystkich takich funkcji $f(n)$, dla których istnieją dodatnie stałe c_1 , c_2 , n_0 , takie że dla wszystkich $n \geq n_0$

$$0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n)$$

Jeśli $f(n) = \Theta(g(n))$ to mówimy, że $g(n)$ jest asymptotycznie dokładnym oszacowaniem funkcji $f(n)$.

$$f(n) = \Theta(g(n)) \iff f(n) = O(g(n)) \wedge f(n) = \Omega(g(n))$$

Notacja Θ

$$\Theta(g(n)) = \{f(n) : \exists_{c_1 > 0} \exists_{c_2 > 0} \exists_{n_0 \in \mathbb{N}} \forall_{n \geq n_0} 0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n)\}$$

Przez $\Theta(g(n))$ oznaczamy zbiór wszystkich takich funkcji $f(n)$, dla których istnieją dodatnie stałe c_1 , c_2 , n_0 , takie że dla wszystkich $n \geq n_0$

$$0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n)$$

Jeśli $f(n) = \Theta(g(n))$ to mówimy, że $g(n)$ jest asymptotycznie dokładnym oszacowaniem funkcji $f(n)$.

$$f(n) = \Theta(g(n)) \iff f(n) = O(g(n)) \wedge f(n) = \Omega(g(n))$$

Przykład 1

$$2n^2 - 3n + 4 = O(n^3)$$

Dowód:

Dla $n \geq 1$ mamy $2n^2 - 3n + 4 \leq 2n^2 + 4 \leq 2n^3 + 4n^3 = 6n^3$.

Zatem nierówność z definicji O jest spełniona dla $n_0 = 1$, $c = 6$

Przykład 2

$$n^2 + 5n - 2 = \Omega(n)$$

Dowód:

Dla $n \geq 1$ mamy $n^2 + 5n - 2 \geq 5n - 2 \geq 5n - 2n = 3n$. Zatem nierówność z definicji Ω jest spełniona dla $n_0 = 1$, $c = 3$

Przykład 3

$$n^2 + 5n = \Theta(n^2)$$

Dowód:

Dla $n \geq 0$ mamy $n^2 + 5n \geq n^2$. Dla $n \geq 1$ mamy $n^2 + 5n \leq n^2 + 5n^2 = 6n^2$. Zatem nierówność z definicji Θ jest spełniona dla $n_0 = 1$, $c_1 = 1$, $c_2 = 6$

Szacowanie funkcji

- $\Theta(1)$ - złożoność stała
- $\Theta(\log n)$ - złożoność logarytmiczna
- $\Theta(n)$ - złożoność liniowa
- $\Theta(n \log n)$ - złożoność liniowo-logarytmiczna
- $\Theta(n^2)$ - złożoność kwadratowa
- $\Theta(n^3)$ - złożoność sześcienna
- $O(n^c)$ - złożoność wielomianowa
- $\Omega(c^n)$ - złożoność wykładnicza

Szacowanie funkcji

- $\Theta(1)$ - złożoność stała
- $\Theta(\log n)$ - złożoność logarytmiczna
- $\Theta(n)$ - złożoność liniowa
- $\Theta(n \log n)$ - złożoność liniowo-logarytmiczna
- $\Theta(n^2)$ - złożoność kwadratowa
- $\Theta(n^3)$ - złożoność sześcienna
- $O(n^c)$ - złożoność wielomianowa
- $\Omega(c^n)$ - złożoność wykładnicza

Szacowanie funkcji

- $\Theta(1)$ - złożoność stała
- $\Theta(\log n)$ - złożoność logarytmiczna
- $\Theta(n)$ - złożoność liniowa
- $\Theta(n \log n)$ - złożoność liniowo-logarytmiczna
- $\Theta(n^2)$ - złożoność kwadratowa
- $\Theta(n^3)$ - złożoność sześcienna
- $O(n^c)$ - złożoność wielomianowa
- $\Omega(c^n)$ - złożoność wykładnicza

Szacowanie funkcji

- $\Theta(1)$ - złożoność stała
- $\Theta(\log n)$ - złożoność logarytmiczna
- $\Theta(n)$ - złożoność liniowa
- $\Theta(n \log n)$ - złożoność liniowo-logarytmiczna
- $\Theta(n^2)$ - złożoność kwadratowa
- $\Theta(n^3)$ - złożoność sześcienna
- $O(n^c)$ - złożoność wielomianowa
- $\Omega(c^n)$ - złożoność wykładnicza

Szacowanie funkcji

- $\Theta(1)$ - złożoność stała
- $\Theta(\log n)$ - złożoność logarytmiczna
- $\Theta(n)$ - złożoność liniowa
- $\Theta(n \log n)$ - złożoność liniowo-logarytmiczna
- $\Theta(n^2)$ - złożoność kwadratowa
- $\Theta(n^3)$ - złożoność sześcienna
- $O(n^c)$ - złożoność wielomianowa
- $\Omega(c^n)$ - złożoność wykładnicza

Szacowanie funkcji

- $\Theta(1)$ - złożoność stała
- $\Theta(\log n)$ - złożoność logarytmiczna
- $\Theta(n)$ - złożoność liniowa
- $\Theta(n \log n)$ - złożoność liniowo-logarytmiczna
- $\Theta(n^2)$ - złożoność kwadratowa
- $\Theta(n^3)$ - złożoność sześcienna
- $O(n^c)$ - złożoność wielomianowa
- $\Omega(c^n)$ - złożoność wykładnicza

Szacowanie funkcji

- $\Theta(1)$ - złożoność stała
- $\Theta(\log n)$ - złożoność logarytmiczna
- $\Theta(n)$ - złożoność liniowa
- $\Theta(n \log n)$ - złożoność liniowo-logarytmiczna
- $\Theta(n^2)$ - złożoność kwadratowa
- $\Theta(n^3)$ - złożoność sześcienna
- $O(n^c)$ - złożoność wielomianowa
- $\Omega(c^n)$ - złożoność wykładnicza

Szacowanie funkcji

- $\Theta(1)$ - złożoność stała
- $\Theta(\log n)$ - złożoność logarytmiczna
- $\Theta(n)$ - złożoność liniowa
- $\Theta(n \log n)$ - złożoność liniowo-logarytmiczna
- $\Theta(n^2)$ - złożoność kwadratowa
- $\Theta(n^3)$ - złożoność sześcienna
- $O(n^c)$ - złożoność wielomianowa
- $\Omega(c^n)$ - złożoność wykładnicza

Przykład 4 - potęga iteracyjnie

POTEGA(a , n)

$a \in \mathbb{R}$

$n \in \mathbb{N}$

pot = 1

for $i = 1$ to n

 pot = pot * a

return pot

- pętla for wykonuje się n razy
- złożoność czasowa $\Theta(n)$

Przykład 4 - potęga iteracyjnie

POTEGA(a , n)

$a \in \mathbb{R}$

$n \in \mathbb{N}$

pot = 1

for $i = 1$ to n

 pot = pot * a

return pot

- pętla for wykonuje się n razy
- złożoność czasowa $\Theta(n)$

Przykład 5 - potęga binarnie

```
POTEGA(a, n)
```

```
  a ∈ ℝ
```

```
  n ∈ ℕ
```

```
  i = n
```

```
  pot = 1
```

```
  pod = a
```

```
  while i ≠ 0
```

```
    if i MOD 2 ≠ 0
```

```
      pot = pot * pod
```

```
      pod = pod * pod
```

```
      i = i DIV 2
```

```
  return pot
```

Przykład 5 - rodzaje złożoności

- Złożoność optymistyczna (best-case $\Theta(1)$)
- Złożoność pesymistyczna (worst-case $\Theta(\log n)$)
- Złożoność w średnim przypadku (average-case $\Theta(\log n)$)
 - Złożoność w średnim przypadku jest trudniejsza do wyznaczenia
 - Dla algorytmów rekurencyjnych można użyć metody drzewka
 - W przypadku innych algorytmów
- Ogólnie $O(\log n)$

Przykład 5 - rodzaje złożoności

- Złożoność optymistyczna (best-case $\Theta(1)$)
- Złożoność pesymistyczna (worst-case $\Theta(\log n)$)
- Złożoność w średnim przypadku (average-case $\Theta(\log n)$)
 - Obrazuje oczekiwany czas działania algorytmu dla losowych danych.
 - Do jej obliczania wykorzystujemy często metody rachunku prawdopodobieństwa.
- Ogólnie $O(\log n)$

Przykład 5 - rodzaje złożoności

- Złożoność optymistyczna (best-case $\Theta(1)$)
- Złożoność pesymistyczna (worst-case $\Theta(\log n)$)
- Złożoność w średnim przypadku (average-case $\Theta(\log n)$)
 - Obrazuje oczekiwany czas działania algorytmu dla losowych danych.
 - Do jej obliczania wykorzystujemy często metody rachunku prawdopodobieństwa.
- Ogólnie $O(\log n)$

Przykład 5 - rodzaje złożoności

- Złożoność optymistyczna (best-case $\Theta(1)$)
- Złożoność pesymistyczna (worst-case $\Theta(\log n)$)
- Złożoność w średnim przypadku (average-case $\Theta(\log n)$)
 - Obrazuje oczekiwany czas działania algorytmu dla losowych danych.
 - Do jej obliczania wykorzystujemy często metody rachunku prawdopodobieństwa.
- Ogólnie $O(\log n)$

Przykład 5 - rodzaje złożoności

- Złożoność optymistyczna (best-case $\Theta(1)$)
- Złożoność pesymistyczna (worst-case $\Theta(\log n)$)
- Złożoność w średnim przypadku (average-case $\Theta(\log n)$)
 - Obrazuje oczekiwany czas działania algorytmu dla losowych danych.
 - Do jej obliczania wykorzystujemy często metody rachunku prawdopodobieństwa.
- Ogólnie $O(\log n)$

Przykład 5 - rodzaje złożoności

- Złożoność optymistyczna (best-case $\Theta(1)$)
- Złożoność pesymistyczna (worst-case $\Theta(\log n)$)
- Złożoność w średnim przypadku (average-case $\Theta(\log n)$)
 - Obrazuje oczekiwany czas działania algorytmu dla losowych danych.
 - Do jej obliczania wykorzystujemy często metody rachunku prawdopodobieństwa.
- Ogólnie $O(\log n)$

Przykład 5 - potęga binarnie

POTEGA(a , n)

$a \in \mathbb{R}$

$n \in \mathbb{N}$

$i = n$

pot = 1

pod = a

while $i \neq 0$

if $i \bmod 2 \neq 0$

 pot = pot * pod

 pod = pod * pod

$i = i \text{ DIV } 2$

return pot

- pętla *while* wykonuje się $\log_2 n$ razy
- złożoność czasowa $O(\log n)$

Przykład 5 - potęga binarnie

POTEGA(a , n)

$a \in \mathbb{R}$

$n \in \mathbb{N}$

$i = n$

pot = 1

pod = a

while $i \neq 0$

if $i \bmod 2 \neq 0$

 pot = pot * pod

 pod = pod * pod

$i = i \text{ DIV } 2$

return pot

- pętla *while* wykonuje się $\log_2 n$ razy
- złożoność czasowa $O(\log n)$

Przykład 6 - potęga rekurencyjnie

POTEGA(a, n)

$a \in \mathbb{R}$

$n \in \mathbb{N}$

if $n = 0$

return 1

else

return $a * \text{POTEGA}(a, n - 1)$

- funkcja czasu $T(n)$ zależna rekurencyjnie od n
- $T(n) = T(n - 1) + 1$ składnik $+1$ odpowiada operacji mnożenia w 6 linijce
- złożoność czasowa $O(n)$

Przykład 6 - potęga rekurencyjnie

POTEGA(a, n)

$a \in \mathbb{R}$

$n \in \mathbb{N}$

if $n = 0$

return 1

else

return $a * \text{POTEGA}(a, n - 1)$

- funkcja czasu $T(n)$ zależna rekurencyjnie od n
- $T(n) = T(n - 1) + 1$ składnik +1 odpowiada operacji mnożenia w 6 linijce
- złożoność czasowa $O(n)$

Przykład 6 - potęga rekurencyjnie

POTEGA(a, n)

$a \in \mathbb{R}$

$n \in \mathbb{N}$

if $n = 0$

return 1

else

return $a * \text{POTEGA}(a, n - 1)$

- funkcja czasu $T(n)$ zależna rekurencyjnie od n
- $T(n) = T(n - 1) + 1$ składnik +1 odpowiada operacji mnożenia w 6 linijce
- złożoność czasowa $O(n)$

Przykład 7 - obliczanie ciągu Fibonnaciego z użyciem tablicy

```
FIB(n)
```

```
  n ∈ N
```

```
  F[0]=0
```

```
  F[1]=1
```

```
  for i = 2 to n
```

```
    F[i]=F[i-1]+F[i-2]
```

```
  return F[n]
```

- zużycie $n + 1$ komórek tablicy F , dwie zmienne n, i
- złożoność pamięciowa $\Theta(n)$

Przykład 7 - obliczanie ciągu Fibonnaciego z użyciem tablicy

FIB(n)

$n \in \mathbb{N}$

$F[0]=0$

$F[1]=1$

for $i = 2$ to n

$F[i]=F[i-1]+F[i-2]$

return $F[n]$

- zużycie $n + 1$ komórek tablicy F , dwie zmienne n, i
- złożoność pamięciowa $\Theta(n)$

Przykład 8 - obliczanie ciągu Fibonnaciego bez tablicy

FIB(n)

$$n \in \mathbb{N}$$

```
if n=0
```

```
return 0
```

```
else if n=1
```

```
return 1
```

$$F_0 = 0$$
$$F1=1$$

```
for i=2 to n
```

$$F_k = F_0 + F_1$$
$$F_0 = F_1$$
$$F_1 = F_k$$

```
return Fk
```

- zużycie 5 zmienne $n, i, F0, F1, Fk$
- złożoność pamięciowa $\Theta(1)$

Przykład 8 - obliczanie ciągu Fibonnaciego bez tablicy

FIB(n)

$n \in \mathbb{N}$

if $n=0$

return 0

else if $n=1$

return 1

$F_0=0$

$F_1=1$

for $i=2$ to n

$F_k=F_0+F_1$

$F_0=F_1$

$F_1=F_k$

return F_k

- zużycie 5 zmienne n , i , F_0 , F_1 , F_k
- złożoność pamięciowa $\Theta(1)$

Przykład 9 - obliczanie ciągu Fibonnaciego rekurencyjnie

FIB(n)

$n \in \mathbb{N}$

if $n=0$

return 0

else if $n=1$

return 1

else

return FIB($n-1$)+FIB($n-2$)

- W pseudokodzie jest jedna zmienna n , jednak algorytm musi zapamiętywać adresy powrotu i kontekst działania funkcji w rozpoczętych wywołaniach rekurencyjnych. Maksymalna głębokość rekurencyjna jest rzędu n .
- złożoność pamięciowa $\Theta(n)$

Przykład 9 - obliczanie ciągu Fibonnaciego rekurencyjnie

FIB(n)

$n \in \mathbb{N}$

if $n=0$

return 0

else if $n=1$

return 1

else

return FIB($n-1$)+FIB($n-2$)

- W pseudokodzie jest jedna zmienna n , jednak algorytm musi zapamiętywać adresy powrotu i kontekst działania funkcji w rozpoczętych wywołaniach rekurencyjnych. Maksymalna głębokość rekurencyjna jest rzędu n .
- złożoność pamięciowa $\Theta(n)$

Twierdzenie o rekurencji uniwersalnej

Niech $a \geq 1$ i $b > 1$ będą stałymi, niech $f(n)$ będzie pewną funkcją i niech $T(n)$ będzie zdefiniowane dla nieujemnych liczb całkowitych przez rekurencję

$$T(n) = aT(n/b) + f(n),$$

gdzie n/b interpretujemy jako $\lfloor n/b \rfloor$ lub $\lceil n/b \rceil$. Wtedy funkcja $T(n)$ może być ograniczona asymptotycznie w następujący sposób:

- 1 Jeśli $f(n) = O(n^{\log_b a - \varepsilon})$ dla pewnej stałej $\varepsilon > 0$, to $T(n) = \Theta(n^{\log_b a})$.
- 2 Jeśli $f(n) = \Theta(n^{\log_b a})$, to $T(n) = \Theta(n^{\log_b a} \lg n)$.
- 3 Jeśli $f(n) = \Omega(n^{\log_b a + \varepsilon})$ dla pewnej stałej $\varepsilon > 0$ oraz $af(n/b) \leq cf(n)$ dla pewnej stałej $c < 1$ i wszystkich dostatecznie dużych n , to $T(n) = \Theta(f(n))$.

Przykład 10 $T(n) = 9T(n/3) + n$

$$T(n) = 9T(n/3) + n$$

$$a = 9$$

$$b = 3$$

$$f(n) = n$$

$$\log_3 9 = 2$$

$$f(n) = n^2 = n^{\log_3 9}$$

$$n = O(n^{2-\epsilon})$$

$$\epsilon = 1 > 0$$

$$\text{Złożoność } \Theta(n^2)$$

Przykład 10 $T(n) = 9T(n/3) + n$

$$T(n) = 9T(n/3) + n$$

$$a = 9$$

$$b = 3$$

$$f(n) = n$$

$$\log_3 9 = 2$$

$$f(n) = n? n^2 = n^{\log_3 9}$$

$$n = O(n^{2-\epsilon})$$

$$\epsilon = 1 > 0$$

$$\text{Złożoność } \Theta(n^2)$$

Przykład 10 $T(n) = 9T(n/3) + n$

$$T(n) = 9T(n/3) + n$$

$$a = 9$$

$$b = 3$$

$$f(n) = n$$

$$\log_3 9 = 2$$

$$f(n) = n? n^2 = n^{\log_3 9}$$

$$n = O(n^{2-\epsilon})$$

$$\epsilon = 1 > 0$$

$$\text{Złożoność } \Theta(n^2)$$

Przykład 10 $T(n) = 9T(n/3) + n$

$$T(n) = 9T(n/3) + n$$

$$a = 9$$

$$b = 3$$

$$f(n) = n$$

$$\log_3 9 = 2$$

$$f(n) = n? n^2 = n^{\log_3 9}$$

$$n = O(n^{2-\epsilon})$$

$$\epsilon = 1 > 0$$

$$\text{Złożoność } \Theta(n^2)$$

Przykład 10 $T(n) = 9T(n/3) + n$

$$T(n) = 9T(n/3) + n$$

$$a = 9$$

$$b = 3$$

$$f(n) = n$$

$$\log_3 9 = 2$$

$$f(n) = n? n^2 = n^{\log_3 9}$$

$$n = O(n^{2-\epsilon})$$

$$\epsilon = 1 > 0$$

$$\text{Złożoność } \Theta(n^2)$$

Przykład 10 $T(n) = 9T(n/3) + n$

$$T(n) = 9T(n/3) + n$$

$$a = 9$$

$$b = 3$$

$$f(n) = n$$

$$\log_3 9 = 2$$

$$f(n) = n? n^2 = n^{\log_3 9}$$

$$n = O(n^{2-\epsilon})$$

$$\epsilon = 1 > 0$$

$$\text{Złożoność } \Theta(n^2)$$

Przykład 10 $T(n) = 9T(n/3) + n$

$$T(n) = 9T(n/3) + n$$

$$a = 9$$

$$b = 3$$

$$f(n) = n$$

$$\log_3 9 = 2$$

$$f(n) = n? n^2 = n^{\log_3 9}$$

$$n = O(n^{2-\epsilon})$$

$$\epsilon = 1 > 0$$

$$\text{Złożoność } \Theta(n^2)$$

Przykład 11 $T(n) = T(2n/3) + 1$

$$T(n) = T(2n/3) + 1 = T(n/\frac{3}{2}) + 1$$

$$a = 1$$

$$b = \frac{3}{2}$$

$$f(n) = 1$$

$$\log_{\frac{3}{2}} 1 = 0$$

$$f(n) = 1? n^0 = n^{\log_{\frac{3}{2}} 1}$$

$$1 = \Theta(n^0)$$

$$\text{Złożoność } \Theta(n^0 \lg n) = \Theta(\lg n)$$

Przykład 11 $T(n) = T(2n/3) + 1$

$$T(n) = T(2n/3) + 1 = T(n/\frac{3}{2}) + 1$$

$$a = 1$$

$$b = \frac{3}{2}$$

$$f(n) = 1$$

$$\log_{\frac{3}{2}} 1 = 0$$

$$f(n) = 1? n^0 = n^{\log_{\frac{3}{2}} 1}$$

$$1 = \Theta(n^0)$$

$$\text{Złożoność } \Theta(n^0 \lg n) = \Theta(\lg n)$$

Przykład 11 $T(n) = T(2n/3) + 1$

$$T(n) = T(2n/3) + 1 = T(n/\frac{3}{2}) + 1$$

$$a = 1$$

$$b = \frac{3}{2}$$

$$f(n) = 1$$

$$\log_{\frac{3}{2}} 1 = 0$$

$$f(n) = 1? n^0 = n^{\log_{\frac{3}{2}} 1}$$

$$1 = \Theta(n^0)$$

$$\text{Złożoność } \Theta(n^0 \lg n) = \Theta(\lg n)$$

Przykład 11 $T(n) = T(2n/3) + 1$

$$T(n) = T(2n/3) + 1 = T(n/\frac{3}{2}) + 1$$

$$a = 1$$

$$b = \frac{3}{2}$$

$$f(n) = 1$$

$$\log_{\frac{3}{2}} 1 = 0$$

$$f(n) = 1? n^0 = n^{\log_{\frac{3}{2}} 1}$$

$$1 = \Theta(n^0)$$

$$\text{Złożoność } \Theta(n^0 \lg n) = \Theta(\lg n)$$

Przykład 11 $T(n) = T(2n/3) + 1$

$$T(n) = T(2n/3) + 1 = T(n/\frac{3}{2}) + 1$$

$$a = 1$$

$$b = \frac{3}{2}$$

$$f(n) = 1$$

$$\log_{\frac{3}{2}} 1 = 0$$

$$f(n) = 1? n^0 = n^{\log_{\frac{3}{2}} 1}$$

$$1 = \Theta(n^0)$$

$$\text{Złożoność } \Theta(n^0 \lg n) = \Theta(\lg n)$$

Przykład 11 $T(n) = T(2n/3) + 1$

$$T(n) = T(2n/3) + 1 = T(n/\frac{3}{2}) + 1$$

$$a = 1$$

$$b = \frac{3}{2}$$

$$f(n) = 1$$

$$\log_{\frac{3}{2}} 1 = 0$$

$$f(n) = 1? n^0 = n^{\log_{\frac{3}{2}} 1}$$

$$1 = \Theta(n^0)$$

$$\text{Złożoność } \Theta(n^0 \lg n) = \Theta(\lg n)$$

Przykład 11 $T(n) = T(2n/3) + 1$

$$T(n) = T(2n/3) + 1 = T(n/\frac{3}{2}) + 1$$

$$a = 1$$

$$b = \frac{3}{2}$$

$$f(n) = 1$$

$$\log_{\frac{3}{2}} 1 = 0$$

$$f(n) = 1? n^0 = n^{\log_{\frac{3}{2}} 1}$$

$$1 = \Theta(n^0)$$

$$\text{Złożoność } \Theta(n^0 \lg n) = \Theta(\lg n)$$

Przykład 12 $T(n) = 3T(n/4) + n \lg n$

$$T(n) = 3T(n/4) + n \lg n$$

$$a = 3$$

$$b = 4$$

$$f(n) = n \lg n$$

$$\log_4 3 \approx 0.7$$

$$f(n) = n \lg n \stackrel{?}{=} n^{0.7} = n^{\log_4 3}$$

$$n \lg n = \Omega(n^{\log_4 3 + \epsilon})$$

$$\epsilon \approx 0.3 > 0$$

$$af(n/b) = 3(n/4) \lg(n/4) \leq (3/4)n \lg n = cf(n)$$

$$c = 3/4 < 1$$

$$\text{Złożoność } \Theta(n \lg n)$$

Przykład 12 $T(n) = 3T(n/4) + n \lg n$

$$T(n) = 3T(n/4) + n \lg n$$

$$a = 3$$

$$b = 4$$

$$f(n) = n \lg n$$

$$\log_4 3 \approx 0.7$$

$$f(n) = n \lg n \stackrel{?}{=} n^{0.7} = n^{\log_4 3}$$

$$n \lg n = \Omega(n^{\log_4 3 + \epsilon})$$

$$\epsilon \approx 0.3 > 0$$

$$af(n/b) = 3(n/4) \lg(n/4) \leq (3/4)n \lg n = cf(n)$$

$$c = 3/4 < 1$$

$$\text{Złożoność } \Theta(n \lg n)$$

Przykład 12 $T(n) = 3T(n/4) + n \lg n$

$$T(n) = 3T(n/4) + n \lg n$$

$$a = 3$$

$$b = 4$$

$$f(n) = n \lg n$$

$$\log_4 3 \approx 0.7$$

$$f(n) = n \lg n \stackrel{?}{=} n^{0.7} = n^{\log_4 3}$$

$$n \lg n = \Omega(n^{\log_4 3 + \epsilon})$$

$$\epsilon \approx 0.3 > 0$$

$$af(n/b) = 3(n/4) \lg(n/4) \leq (3/4)n \lg n = cf(n)$$

$$c = 3/4 < 1$$

$$\text{Złożoność } \Theta(n \lg n)$$

Przykład 12 $T(n) = 3T(n/4) + n \lg n$

$$T(n) = 3T(n/4) + n \lg n$$

$$a = 3$$

$$b = 4$$

$$f(n) = n \lg n$$

$$\log_4 3 \approx 0.7$$

$$f(n) = n \lg n? n^{0.7} = n^{\log_4 3}$$

$$n \lg n = \Omega(n^{\log_4 3 + \epsilon})$$

$$\epsilon \approx 0.3 > 0$$

$$af(n/b) = 3(n/4) \lg(n/4) \leq (3/4)n \lg n = cf(n)$$

$$c = 3/4 < 1$$

$$\text{Złożoność } \Theta(n \lg n)$$

Przykład 12 $T(n) = 3T(n/4) + n \lg n$

$$T(n) = 3T(n/4) + n \lg n$$

$$a = 3$$

$$b = 4$$

$$f(n) = n \lg n$$

$$\log_4 3 \approx 0.7$$

$$f(n) = n \lg n \cdot n^{0.7} = n^{\log_4 3}$$

$$n \lg n = \Omega(n^{\log_4 3 + \epsilon})$$

$$\epsilon \approx 0.3 > 0$$

$$af(n/b) = 3(n/4) \lg(n/4) \leq (3/4)n \lg n = cf(n)$$

$$c = 3/4 < 1$$

$$\text{Złożoność } \Theta(n \lg n)$$

Przykład 12 $T(n) = 3T(n/4) + n \lg n$

$$T(n) = 3T(n/4) + n \lg n$$

$$a = 3$$

$$b = 4$$

$$f(n) = n \lg n$$

$$\log_4 3 \approx 0.7$$

$$f(n) = n \lg n \cdot n^{0.7} = n^{\log_4 3}$$

$$n \lg n = \Omega(n^{\log_4 3 + \epsilon})$$

$$\epsilon \approx 0.3 > 0$$

$$af(n/b) = 3(n/4) \lg(n/4) \leq (3/4)n \lg n = cf(n)$$

$$c = 3/4 < 1$$

Złożoność $\Theta(n \lg n)$

Przykład 12 $T(n) = 3T(n/4) + n \lg n$

$$T(n) = 3T(n/4) + n \lg n$$

$$a = 3$$

$$b = 4$$

$$f(n) = n \lg n$$

$$\log_4 3 \approx 0.7$$

$$f(n) = n \lg n \cdot n^{0.7} = n^{\log_4 3}$$

$$n \lg n = \Omega(n^{\log_4 3 + \epsilon})$$

$$\epsilon \approx 0.3 > 0$$

$$af(n/b) = 3(n/4) \lg(n/4) \leq (3/4)n \lg n = cf(n)$$

$$c = 3/4 < 1$$

Złożoność $\Theta(n \lg n)$

Przykład 12 $T(n) = 3T(n/4) + n \lg n$

$$T(n) = 3T(n/4) + n \lg n$$

$$a = 3$$

$$b = 4$$

$$f(n) = n \lg n$$

$$\log_4 3 \approx 0.7$$

$$f(n) = n \lg n \cdot n^{0.7} = n^{\log_4 3}$$

$$n \lg n = \Omega(n^{\log_4 3 + \epsilon})$$

$$\epsilon \approx 0.3 > 0$$

$$af(n/b) = 3(n/4) \lg(n/4) \leq (3/4)n \lg n = cf(n)$$

$$c = 3/4 < 1$$

Złożoność $\Theta(n \lg n)$

Przykład 12 $T(n) = 3T(n/4) + n \lg n$

$$T(n) = 3T(n/4) + n \lg n$$

$$a = 3$$

$$b = 4$$

$$f(n) = n \lg n$$

$$\log_4 3 \approx 0.7$$

$$f(n) = n \lg n \cdot n^{0.7} = n^{\log_4 3}$$

$$n \lg n = \Omega(n^{\log_4 3 + \epsilon})$$

$$\epsilon \approx 0.3 > 0$$

$$af(n/b) = 3(n/4) \lg(n/4) \leq (3/4)n \lg n = cf(n)$$

$$c = 3/4 < 1$$

Złożoność $\Theta(n \lg n)$

Przykład 12 $T(n) = 3T(n/4) + n \lg n$

$$T(n) = 3T(n/4) + n \lg n$$

$$a = 3$$

$$b = 4$$

$$f(n) = n \lg n$$

$$\log_4 3 \approx 0.7$$

$$f(n) = n \lg n \cdot n^{0.7} = n^{\log_4 3}$$

$$n \lg n = \Omega(n^{\log_4 3 + \epsilon})$$

$$\epsilon \approx 0.3 > 0$$

$$af(n/b) = 3(n/4) \lg(n/4) \leq (3/4)n \lg n = cf(n)$$

$$c = 3/4 < 1$$

Złożoność $\Theta(n \lg n)$

Przykład 12 $T(n) = 3T(n/4) + n \lg n$

$$T(n) = 3T(n/4) + n \lg n$$

$$a = 3$$

$$b = 4$$

$$f(n) = n \lg n$$

$$\log_4 3 \approx 0.7$$

$$f(n) = n \lg n \cdot n^{0.7} = n^{\log_4 3}$$

$$n \lg n = \Omega(n^{\log_4 3 + \epsilon})$$

$$\epsilon \approx 0.3 > 0$$

$$af(n/b) = 3(n/4) \lg(n/4) \leq (3/4)n \lg n = cf(n)$$

$$c = 3/4 < 1$$

Złożoność $\Theta(n \lg n)$

Przykład 12 $T(n) = 3T(n/4) + n \lg n$

$$T(n) = 3T(n/4) + n \lg n$$

$$a = 3$$

$$b = 4$$

$$f(n) = n \lg n$$

$$\log_4 3 \approx 0.7$$

$$f(n) = n \lg n? n^{0.7} = n^{\log_4 3}$$

$$n \lg n = \Omega(n^{\log_4 3 + \epsilon})$$

$$\epsilon \approx 0.3 > 0$$

$$af(n/b) = 3(n/4) \lg(n/4) \leq (3/4)n \lg n = cf(n)$$

$$c = 3/4 < 1$$

$$\text{Złożoność } \Theta(n \lg n)$$

Maszyna Turinga

- **Deterministyczna Maszyna Turinga**

$$DTM = (Q, \Sigma, \delta, \Gamma, q_0, \phi, F)$$

- Q - skończony zbiór stanów
- q_0 - stan początkowy $q_0 \in Q$
- F - zbiór stanów końcowych
- Γ - skończony zbiór dopuszczalnych symboli
- ϕ - symbol pusty $\phi \in \Gamma$
- Σ - zbiór symboli wejściowych, podzbiór zbioru Γ
- δ - funkcja $(Q, \Gamma) \rightarrow (Q, \Gamma, \{L, -, R\})$

Maszyna Turinga

- **Deterministyczna Maszyna Turinga**

$$DTM = (Q, \Sigma, \delta, \Gamma, q_0, \phi, F)$$

- Q - skończony zbiór stanów
- q_0 - stan początkowy $q_0 \in Q$
- F - zbiór stanów końcowych
- Γ - skończony zbiór dopuszczalnych symboli
- ϕ - symbol pusty $\phi \in \Gamma$
- Σ - zbiór symboli wejściowych, podzbiór zbioru Γ
- δ - funkcja $(Q, \Gamma) \rightarrow (Q, \Gamma, \{L, -, R\})$

Maszyna Turinga

- **Deterministyczna Maszyna Turinga**

$$DTM = (Q, \Sigma, \delta, \Gamma, q_0, \phi, F)$$

- Q - skończony zbiór stanów
- q_0 - stan początkowy $q_0 \in Q$
- F - zbiór stanów końcowych
- Γ - skończony zbiór dopuszczalnych symboli
- ϕ - symbol pusty $\phi \in \Gamma$
- Σ - zbiór symboli wejściowych, podzbiór zbioru Γ
- δ - funkcja $(Q, \Gamma) \rightarrow (Q, \Gamma, \{L, -, R\})$

Maszyna Turinga

- **Deterministyczna Maszyna Turinga**

$$DTM = (Q, \Sigma, \delta, \Gamma, q_0, \phi, F)$$

- Q - skończony zbiór stanów
- q_0 - stan początkowy $q_0 \in Q$
- F - zbiór stanów końcowych
- Γ - skończony zbiór dopuszczalnych symboli
- ϕ - symbol pusty $\phi \in \Gamma$
- Σ - zbiór symboli wejściowych, podzbiór zbioru Γ
- δ - funkcja $(Q, \Gamma) \rightarrow (Q, \Gamma, \{L, -, R\})$

Maszyna Turinga

- **Deterministyczna Maszyna Turinga**

$$DTM = (Q, \Sigma, \delta, \Gamma, q_0, \phi, F)$$

- Q - skończony zbiór stanów
- q_0 - stan początkowy $q_0 \in Q$
- F - zbiór stanów końcowych
- Γ - skończony zbiór dopuszczalnych symboli
- ϕ - symbol pusty $\phi \in \Gamma$
- Σ - zbiór symboli wejściowych, podzbiór zbioru Γ
- δ - funkcja $(Q, \Gamma) \rightarrow (Q, \Gamma, \{L, -, R\})$

Maszyna Turinga

- **Deterministyczna Maszyna Turinga**

$$DTM = (Q, \Sigma, \delta, \Gamma, q_0, \phi, F)$$

- Q - skończony zbiór stanów
- q_0 - stan początkowy $q_0 \in Q$
- F - zbiór stanów końcowych
- Γ - skończony zbiór dopuszczalnych symboli
- ϕ - symbol pusty $\phi \in \Gamma$
- Σ - zbiór symboli wejściowych, podzbiór zbioru Γ
- δ - funkcja $(Q, \Gamma) \rightarrow (Q, \Gamma, \{L, -, R\})$

Maszyna Turinga

- **Deterministyczna Maszyna Turinga**

$$DTM = (Q, \Sigma, \delta, \Gamma, q_0, \phi, F)$$

- Q - skończony zbiór stanów
- q_0 - stan początkowy $q_0 \in Q$
- F - zbiór stanów końcowych
- Γ - skończony zbiór dopuszczalnych symboli
- ϕ - symbol pusty $\phi \in \Gamma$
- Σ - zbiór symboli wejściowych, podzbiór zbioru Γ
- δ - funkcja $(Q, \Gamma) \rightarrow (Q, \Gamma, \{L, -, R\})$

Maszyna Turinga

- **Deterministyczna Maszyna Turinga**

$$DTM = (Q, \Sigma, \delta, \Gamma, q_0, \phi, F)$$

- Q - skończony zbiór stanów
- q_0 - stan początkowy $q_0 \in Q$
- F - zbiór stanów końcowych
- Γ - skończony zbiór dopuszczalnych symboli
- ϕ - symbol pusty $\phi \in \Gamma$
- Σ - zbiór symboli wejściowych, podzbiór zbioru Γ
- δ - funkcja $(Q, \Gamma) \rightarrow (Q, \Gamma, \{L, -, R\})$

Maszyna Turinga - przykład

	q_0	q_1	q_2
ϕ	q_0, ϕ, L	$q_2, 1, L$	K
0	$q_2, 1, L$	$q_2, 1, L$	K
1	$q_1, 0, L$	$q_1, 0, L$	K

Klasy P i NP

- **Niedeterministyczna Maszyna Turinga (NDTM)**
niedeterministyczny odpowiednik DTM
- Klasę **P** tworzą wszystkie problemy decyzyjne, które DTM rozwiązuje w co najwyżej wielomianowym czasie
- Klasę **NP** tworzą wszystkie problemy decyzyjne, które NDTM rozwiązuje w co najwyżej wielomianowym czasie
- Nie wiadomo, czy $P \neq NP$

Klasy P i NP

- **Niedeterministyczna Maszyna Turinga (NDTM)**
niedeterministyczny odpowiednik DTM
- **Klasę P** tworzą wszystkie problemy decyzyjne, które DTM rozwiązuje w co najwyżej wielomianowym czasie
- Klasę NP tworzą wszystkie problemy decyzyjne, które NDTM rozwiązuje w co najwyżej wielomianowym czasie
- Nie wiadomo, czy $P \neq NP$

Klasy P i NP

- **Niedeterministyczna Maszyna Turinga (NDTM)**
niedeterministyczny odpowiednik DTM
- **Klasę P** tworzą wszystkie problemy decyzyjne, które DTM rozwiązuje w co najwyżej wielomianowym czasie
- **Klasę NP** tworzą wszystkie problemy decyzyjne, które NDTM rozwiązuje w co najwyżej wielomianowym czasie
- Nie wiadomo, czy $P \neq NP$

Klasy P i NP

- **Niedeterministyczna Maszyna Turinga (NDTM)**
niedeterministyczny odpowiednik DTM
- **Klasę P** tworzą wszystkie problemy decyzyjne, które DTM rozwiązuje w co najwyżej wielomianowym czasie
- **Klasę NP** tworzą wszystkie problemy decyzyjne, które NDTM rozwiązuje w co najwyżej wielomianowym czasie
- Nie wiadomo, czy $P \neq NP$