

# Algorytmy i struktury danych - Złożoność algorytmów, Sortowanie

Marcin Żurowski

18 listopada 2019

# 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)
```

# Szacowanie funkcji

$$\exists n_0 \in \mathbb{N} \forall n \geq n_0 \quad 1 \leq \lg n \leq n \leq n \lg n \leq n^2 \leq n^3 \leq \dots \leq n^k \leq \\ \leq \dots \leq 2^n \leq n! \leq n^n$$

## Zadania

- 1  $f(n) = 13n^2 + 4n - 73$
- 2  $f(n) = (n^2 + 1)(2n^4 + 3n - 8)$
- 3  $f(n) = (n^3 + 3n - 1)^4$
- 4  $f(n) = \sqrt{n + 1}$
- 5  $f(n) = (n^2 - 1)^7$
- 6  $f(n) = \sqrt{n^2 - 1}$
- 7  $f(n) = \sqrt{n^2 + n}$
- 8  $f(n) = (n^2 + n + 1)(n^3 + 5)$
- 9  $f(n) = 3^n$
- 10  $f(n) = 2^{n+1}$
- 11  $f(n) = 2^{2n}$
- 12  $f(n) = (n + 1)^2$
- 13  $f(n) = (200n)^2$

# Notacje asymptotyczne

- $O(g(n)) = \{f(n) : \exists c > 0 \exists n_0 \in \mathbb{N} \forall n \geq n_0 0 \leq f(n) \leq cg(n)\}$
- $\Theta(g(n)) =$   
 $\{f(n) : \exists c_1 > 0 \exists c_2 > 0 \exists n_0 \in \mathbb{N} \forall n \geq n_0 0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n)\}$
- $\Omega(g(n)) = \{f(n) : \exists c > 0 \exists n_0 \in \mathbb{N} \forall n \geq n_0 0 \leq cg(n) \leq f(n)\}$

# 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  $T(n) = \Theta(f(n))$ .



## Przykłady:

- 1  $T(n) = 9T(n/3) + n$
- 2  $T(n) = T(2n/3) + 1$
- 3  $T(n) = 3T(n/4) + n \lg n$
- 4  $T(n) = 2T(n/2) + n \lg n$
- 5  $T(n) = 4T(n/2) + n$
- 6  $T(n) = T(n-1) + 1$
- 7  $T(n) = 4T(n/2) + n^2$
- 8  $T(n) = 4T(n/2) + n^3$
- 9  $T(n) = 2T(n/2) + n^2$
- 10  $T(n) = T(9n/10) + n$
- 11  $T(n) = 16T(n/4) + n^2$
- 12  $T(n) = 7T(n/3) + n^2$
- 13  $T(n) = 2T(n/4) + \sqrt{n}$
- 14  $T(n) = T(n-1) + n$
- 15  $T(n) = 9T(n/3) + n^3$
- 16  $T(n) = T(n/3) + 1$

## 1

Napisz definicję procedury sortowania tablicy  $A[1..n]$  zawierającej liczby przez porównanie każdej pary elementów tablicy.

## 2

Podać definicję procedury sortującej tablicę  $A[1..n]$  przez wybór elementów minimalnych.

```
procedure MINIMUM-3(A,d,g,l)
  integer d,g,l
  real min
  real array A[1..n]
min = A[d]
l = d
for i = d + 1 to g
  if A[i] < min
    min = A[i]
    l = i
```

Zapisać definicję procedury sortowania bąbelkowego tablicy  $A[1..n]$ .

Napisz algorytm sortowania przez wstawianie tablicy  $A[1..n]$ .

## 5

Napisz algorytm sortowania przez scalanie tablicy  $A[1..n]$ .

```
procedure SCAL-1(A,p,q,r)
  integer p,q,r
  real array A[1..r]
  integer i,j,l
  real array B[1..r]
  i = p
  j = q + 1
  l = p
  while (i ≤ q) and (j ≤ r)
    if A[i] ≤ A[j]
      B[l] = A[i]
      i = i + 1
    else
      B[l] = A[j]
      j = j + 1
    l = l + 1
```

## 5

```
while i ≤ q
  B[l] = A[i]
  i = i + 1
  l = l + 1
while j ≤ r
  B[l] = A[j]
  j = j + 1
  l = l + 1
for i = p to r
  A[i] = B[i]
```

## 6

Napisz algorytm sortowania szybkiego tablicy  $A[1..n]$ .

```
procedure PODZIEL(A, p, r, q)
```

```
  integer p, r, q, i
```

```
  real array A[1..r]
```

```
  real x
```

```
x = A[r]
```

```
i = p - 1
```

```
for j = p to r - 1
```

```
  if  $A[j] \leq x$ 
```

```
    i = i + 1
```

```
     $A[i] \leftrightarrow A[j]$ 
```

```
 $A[i + 1] \leftrightarrow A[r]$ 
```

```
q = i + 1
```



Napisz algorytm sortowania przez zliczanie tablicy  $A[1..n]$ .