Wprowadzenie do Algorytmiki. Proste algorytmy i złożoność obliczeniowa.

Artur Laskowski

28 października 2021, Poznań

Logarytm

Logarytm to odwrotność potęgowania.
Do jakiej potęgi należy podnieść liczbę, aby otrzymać określony wynik.

$$10^2 = 100 \longrightarrow \log_{10} 100 = 2$$

$$2^8 = 256 \longrightarrow \log_2 256 = 8$$

$$\log_{10} 89 = 1.95 \longleftarrow 10^{1.95} = 89$$

Rekurencja

Rekurencją jest wywołanie funkcji przez nią samą.
Rekurencja musi posiadać:

- Warunek stopu
- Wywołanie rekurencyjnie

Rekurencyjne obliczania silni

$$n! = 1 \cdot 2 \cdot \ldots \cdot (n-1) \cdot n = (n-1)! \cdot n$$

Rozwiązanie:

```
int silnia(int n) {
    if (n <= 1) return 1;
    return silnia(n-1) * n;
}
```

O rekurencji

Zalety:

- prostota
- czytelność

Wady:

- trochę wolniejszy
- pamięciożerność

Największy wspólny dzielnik

Algorytm naiwny - jeżeli x dzieli a oraz b, to dzieli róznież $|a - b|$.

```
int nwd(int a, int b) {
    if (a == b) return a;
    return nwd(max(a, b) - min(a, b), min(a, b))
}
```

NWD - przykład

15	12
3	12
3	9
3	6
3	3

NWD - przykład 2

1000	1
999	1
998	1
997	1
996	1

...

Algorytm Euklides

```
int euclides(int a, int b) {
    return (b == 0) ? a :
        euclides(b, a % b);
}
```

Ile operacji wykonuje for

```
for(int i = 0; i < n; ++i) suma += i;
```

- n - tyle wykonujemy sumowań
- $2n$ - każda pętla inkrementuje i
- $3n$ - każda pętla sprawdza warunek
- $3n + 1$ - bo inicjalizujemy i
- $4n + 1$ - bo warunek w praktyce to dwie operacje (CMP+JMP)
- $5n + 1$ - instrukcja wykonuje się 2 cykle
- ...

Liczba operacji

n	$10n+1$	n^2	$0.01n^3$	$n!$
1	11	1	1	1
10	10^2+1	10^2	10	3628800
10^2	10^3+1	10^4	10^4	$\approx 10^{158}$
10^3	10^4+1	10^6	10^7	$\approx 10^{2568}$
10^6	10^7+1	10^{12}	10^{16}	?

Notacja O

Notacja O() opisuje złożoność algorytmu, a nie problemu!
Istnieją algorytmy:

- wielomianowe $(1, n, n^c, \log n)$
- wykładnicze $(n!, 2^n, n^n)$

Przykłady

```
for (int i = 0; i < n; ++i) suma += i;
```

Przykłady

```
for (int i = 0; i < n; ++i) suma += i;
```

$O(n)$

Przykłady

```
suma = n * n;
```

Przykłady

```
suma = n * n;
```

$O(1)$

Przykłady

```
for (int i = 0; i < n / 1000; i = i + 10) suma += 10;
```

Przykłady

```
for (int i = 0; i < n / 1000; i = i + 10) suma += 10;
```

$O(n)$

Przykłady

```
for (int i = 0; i < n; ++i) {
    for (int j = 0; j < i; ++j) {
        suma += i * j;
    }
}
```

Przykłady

```
for (int i = 0; i < n; ++i) {
    for (int j = 0; j < i; ++j) {
        suma += i * j;
    }
}
```

$O(n^2)$

Przykłady

```
for (int i = 0; i < n; ++i) {
    for (int j = 0; j < m; ++j) {
        suma += i * j;
    }
}
```

Przykłady

```
for (int i = 0; i < n; ++i) {
    for (int j = 0; j < m; ++j) {
        suma += i * j;
    }
}
```

$O(nm)$

Przykłady

```
for (int i = 0; i < n; ++i) {
    for (int j = 0; j < m; ++j) {
        suma += i * j;
    }
}
for (int i = 0; i < n; ++i) {
    suma += i;
}
```

Przykłady

```
for (int i = 0; i < n; ++i) {
    for (int j = 0; j < m; ++j) {
        suma += i * j;
    }
}
for (int i = 0; i < n; ++i) {
    suma += i;
}
```

$O(nm)$

Przykłady

```
for (int i = n - 100; i < n; ++i) {
    for (int j = 0; j < m; ++j) {
        suma += i * j;
    }
}
```

Przykłady

```
for (int i = n - 100; i < n; ++i) {
    for (int j = 0; j < m; ++j) {
        suma += i * j;
    }
}
```

$O(m)$

Przykłady

```
int i = 0;
while (i < n) suma += i++;
```

Przykłady

```
int i = 0;
while (i < n) suma += i++;
```

$O(n)$

Przykłady

```
for (int i = 1; i < n; i = 2 * i) suma += i;
```

Przykłady

```
for (int i = 1; i < n; i = 2 * i) suma += i;
```

$O(\log n)$

Przykłady

```
int sumuj(int n) {
    if (n == 0) return 0;
    else return sumuj(n-1) + n;
}
```

Przykłady

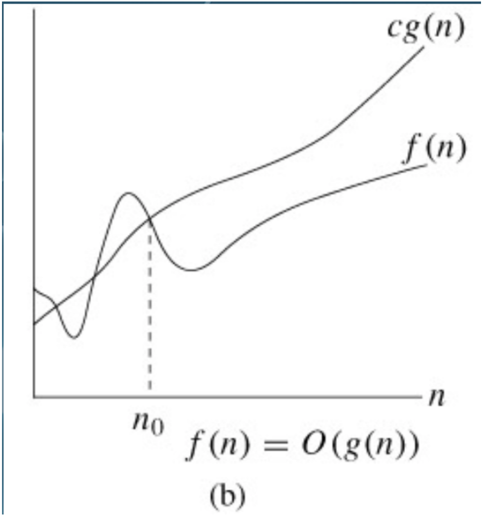
```
int sumuj(int n) {
    if (n == 0) return 0;
    else return sumuj(n-1) + n;
}
```

$O(n)$

Definicja

$O(g(n)) = \{f(n)$:istnieją dodatnie stałe c i n_0 takie, że
$0 <= f(n) <= c \cdot g(n)$ dla wszystkich $n \geqslant n_0)$
Odpowiednio wyskalowaną funkcją $g()$
można ograniczyć od góry funkcję $f()$

Przykład graficzny



Przykład

Przykład: dla f(n) = 5n + 1

$g(n) = n$

$c = 6, n_0 = 1$

$f(n_0) = 5 \cdot 1 + 1 = 6 \leqslant 6 \cdot g(n_0) = 6 \cdot 1 = 6$

$f(n_0 + 1) = 5 \cdot 2 + 1 = 11 \leqslant 6 \cdot g(n_0 + 1) = 6 \cdot 2 = 12$

$O()$ to notacja, a nie funkcja

Rozmiar instancji

Czym jest właściwie n w opisie złożoności?
Jest to rozmiar instancji.

Zadanie

Mamy dany ciąg n rezerwacji biletów kolejowych, dla każdej znane są stacja początkowa a_i oraz końcowa b_i.

Na którym odcinku trasy pociąg będzie najbardziej wypełniony?

Laboratoria

https://www.hackerrank.com/wda-02-2021