Złożoność algorytmów

MD

Złożoność obliczeniowa

Złożoność obliczeniowa algorytmu – ilość zasobów komputerowych potrzebnych do jego wykonania:

- **Złożoność czasowa** – to ilość czasu potrzebnego do wykonania zadania, wyrażona jako funkcja ilości danych.
- **Złożoność pamięciowa** – to ilość pamięci potrzebnej do wykonania zadania, wyrażona jako funkcja ilości danych.

Można wyróżnić:

- **złożoność pesymistyczną O()** - ilość zasobów potrzebnych do wykonania algorytmu przy założeniu najbardziej "złośliwych/najgorszych" danych,
- **złożoność oczekiwaną Θ()** - ilość zasobów potrzebnych do wykonania algorytmu przy założeniu "typowych" (statystycznie oczekiwanych) danych wejściowych,
- **złożoność optymistyczną** - ilość zasobów potrzebnych do wykonania algorytmu przy założeniu "najlepszych" danych.

Złożoności obliczeniowe:

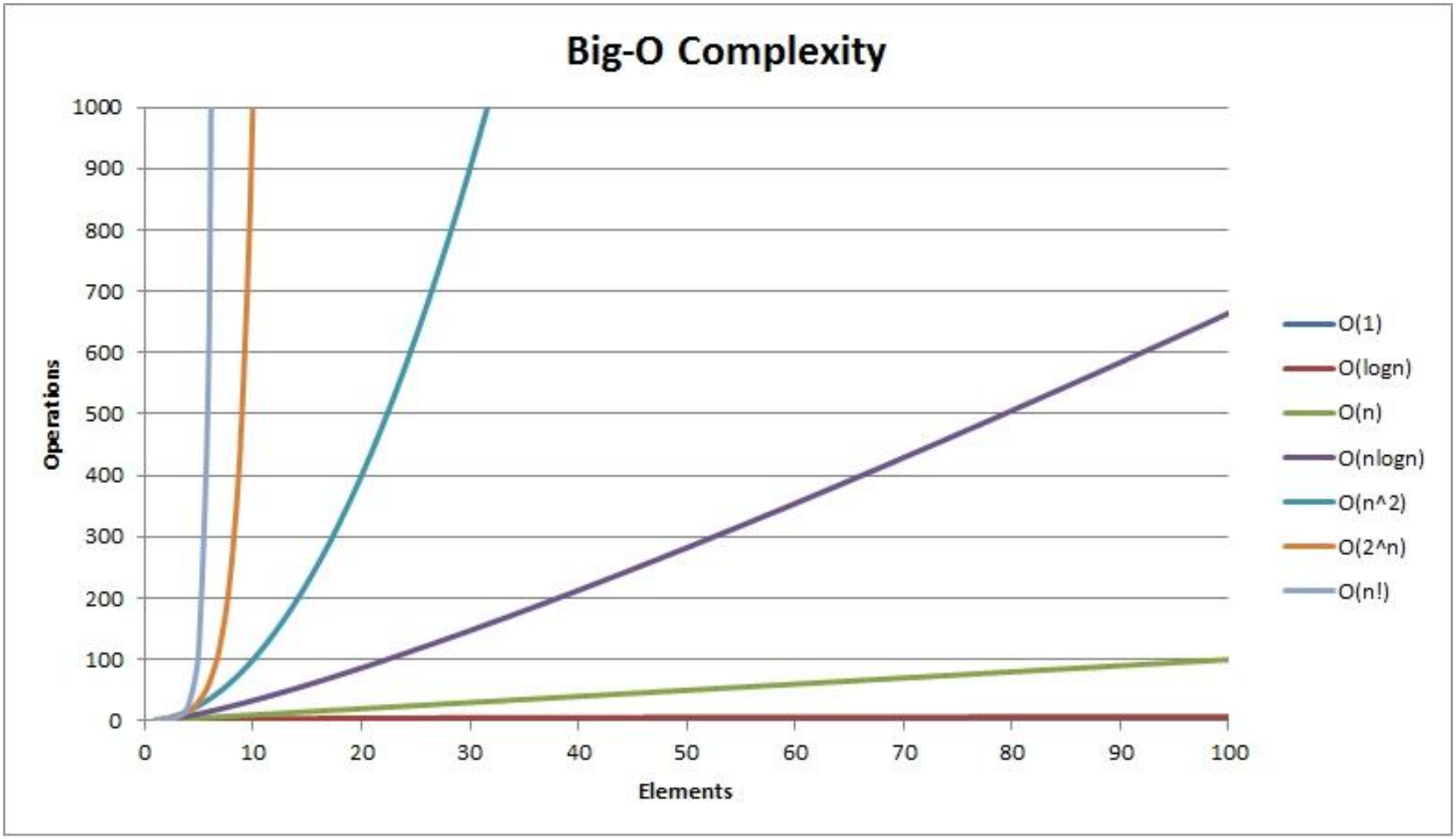
- **stała - Θ(1)** - gdy czas wykonania algorytmu jest stały i niezależny od rozmiaru danych wejściowych,
- **logarytmiczna - Θ(log n)** - kiedy czas ten rośnie logarytmiczne wraz ze wzrostem wielkości danych. Logarytm jest niemal zawsze o podstawie 2 (w przypadku notacji asymptotycznych nie ma to aczkolwiek znaczenia, bowiem podstawa logarytmu może być zmieniona poprzez pomnożenie przez czynnik stały,
- **liniowa - Θ(n)** - czas działania jest proporcjonalny do rozmiaru danych wejściowych,
- **liniowo-logarytmiczna - Θ(n log n)** - złożoność jest iloczynem funkcji liniowej i logarytmicznej,
- **kwadratowa - $\Theta(n^2)$** - liczba instrukcji algorytmu rośnie proporcjonalnie do kwadratu rozmiaru danych wejściowych,
- **sześcienna - $\Theta(n^3)$** - liczba instrukcji algorytmu rośnie proporcjonalnie do sześcianu rozmiaru danych wejściowych,
- **wielomianowa - $\Theta(n^r + n^{r-1} + ... + n^1)$** - liczba instrukcji algorytmu rośnie proporcjonalnie do pewnego wielomianu rozmiaru danych wejściowych,
- **wykładnicza - $\Theta(2^n)$** - czas wykonania rośnie wykładniczo względem rozmiaru danych,
- **silni - Θ(n!)** - czas wykonania rośnie z szybkością silni względem rozmiaru danych.

Jeśliby założono, że pojedyncza instrukcja wykonuje się jedną nanosekundę (czyli na komputerze działającym z częstotliwością 1 GHz) wtedy czasy wykonania względem rozmiaru danych wyglądałyby następująco:

Złożoność obliczeniowa względem rozmiaru danych wejściowych

Rozmiar danych:	10	20	50	100	200	1000
log n	3,32 ns	4,23 ns	5,64 ns	6,64 ns	7,64 ns	9,97 ns
n	10 ns	20 ns	50 ns	100 ns	200 ns	1 µs
n log n	33,21 ns	86,44 ns	282,2 ns	664,4 ns	1,53 µs	9,97 µs
n^2	100 ns	400 ns	2,5 µs	10 µs	40 µs	1 ms
2^n	1 µs	1,05 ms	13 dni	$4 \cdot 10^{13}$ lat	$5{,}1 \cdot 10^{43}$ lat	$3{,}4 \cdot 10^{284}$ lat
n!	3,6 ms	77 lat	$9{,}6 \cdot 10^{44}$ lat	$3 \cdot 10^{141}$ lat	$2{,}5 \cdot 10^{358}$ lat	$1{,}27 \cdot 10^{2551}$ lat

Złożoność obliczeniowa



Złożoność obliczeniowa

- Jeśli jedna część algorytmu o złożoności $\Theta(\log n)$ zawiera się w pętli wykonywanej n razy, wtedy złożoność całego kodu będzie: $\Theta(n) \cdot \Theta(\log n) = \Theta(n \log n)$
- Algorytm ma taką złożoność, jak jego najbardziej czasochłonny fragment, np.: $\Theta(n) + \Theta(n2) = \Theta(n + n2) = \Theta(n2)$

Złożoność czasowa

Złożoność czasowa powinna być własnością samego algorytmu jako metody rozwiązania problemu, więc powinna być niezależna od komputera, języka programowania czy sposobu jego kodowania.

W tym celu wyróżnia się tzw. **operacje dominujące** – takie, których łączna liczba jest proporcjonalna do liczby wykonań wszystkich operacji jednostkowych w dowolnej komputerowej realizacji algorytmu.
Za **jednostkę czasową** przyjmuje się wykonanie jednej operacji dominującej.

Bibliografia

http://home.agh.edu.pl/~horzyk/lectures/pi/ahdydpiwykl8.html