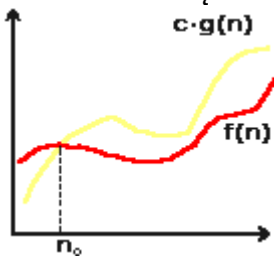
# Złożoność obliczeniowa

Jest to jeden z najważniejszych parametrów charakteryzujących algorytm. Decyduje on o efektywności całego programu. Podstawowymi zasobami systemowymi uwzględnianymi w analizie algorytmów są czas działania oraz obszar zajmowanej pamięci. Na złożoność czasową składają się dwie wartości: pesymistyczna, czyli taka, która charakteryzuje najgorszy przypadek działania oraz oczekiwana.

Najczęściej algorytmy mają złożoność czasową proporcjonalną do funkcji:
log(n)- złożoność logarytmiczna
n - złożoność liniowa
nlog(n) - złożoność liniowo-logarytmiczna
$n^2$ - złożoność kwadratowa
$n^k$ - złożoność wielomianowa
$2^n$ - złożoność wykładnicza
n! - złożoność wykładnicza, ponieważ n!>2n już od n=4

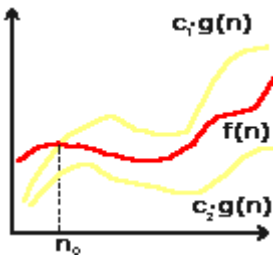
Rząd wielkości służy do opisu czasu działania algorytmu. Istnieją trzy notacje służące do tego celu.
1. Notacja O (omikron).
Jest to ograniczenie funkcji od góry. Gdy mówimy, że pewna f(n) funkcja jest rzędu g(n), co zapisujemy f(n)=O(g(n)), znaczy to, że istnieje taki argument $n_0$, od którego począwszy dla każdego niemniejszego od $n_0$ wartości funkcji f(n) są niewiększe od wartości funkcji g(n) z dokładnością do stałej c. Brzmi to trochę skomplikowanie, więc zademonstruję to na przykładzie:




Na prawo od punktu od n0 funkcja f(n) znajduje się pod funkcją c·g(n), czyli jest przez nią ograniczona z góry.

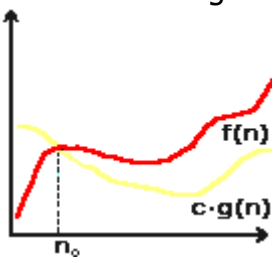
Jest to asymptotyczna granica górna. Służy do szacowania czasu działania algorytmu w przypadku pesymistycznym.
2. Notacja Θ (theta)
Notacja ta ogranicza funkcję f(n) od góry, tak jak notacja O oraz dodatkowo od dołu. Oznacza to, że jeżeli f(n)=Θ(g(n)) to istnieje taki argument $n_0$, od którego począwszy dla każdego argumentu od niego niemniejszego:
wartości funkcji f(n) są niewiększe od wartości funkcji g(n) z dokładnością do stałej $c_1$
wartości funkcji f(n) są niemniejsze od wartości funkcji g(n) z dokładnością do stałej $c_2$

I znów przykład:




Na prawo od punktu od $n_0$ funkcja f(n) znajduje się pod funkcją $c_1$·g(n), czyli jest przez nią ograniczona z góry i nad funkcją $c_2$·g(n), czyli jest przez nią ograniczona z dołu

Jest to asymptotyczne oszacowanie dokładne.
3. Notacja Ω (omega).
Notacja ta ogranicza funkcję f(n) od dołu. Oznacza to, że jeśli f(n)=Ω(g(n)) to istnieje taki argument $n_0$, od którego począwszy dla każdego argumentu od niego niemniejszego funkcja f(n) jest niemniejsza niż g(n) z dokładnością do stałej c:




Na prawo od punktu od $n_0$ funkcja f(n) znajduje się nad funkcją c·g(n).
Jest to asymptotyczna granica dolna. Służy więc do oszacowania działania algorytmu w najlepszym przypadku.

Definicja Złożoność czasowa to liczba operacji dominujących (podstawowych) wykonanych przez algorytm w czasie jego realizacji, wyrażona jako funkcja rozmiaru danych.
Uwaga1. Faktyczny czas wykonania algorytmu jest proporcjonalny do złożoności czasowej.
Uwaga2. Czas wykonania algorytmu jest bardziej interesujący dla dużych n niż dla małych.

Ile czasu potrzeba na rozwiązanie zadania o ustalonym rozmiarze i złożoności?

| T(A,n) / wymiar | log n | n | n log n | $n^2$ | $n^3$ | $2^n$ |
|---|---|---|---|---|---|---|
| $n=10^2$ | 6.6ms | 0.1ms | 0.6ms | 10ms | 1s | $10^6$ lat |
| $n= 10^4$ | 13.3ms | 10ms | 0.1s | 100s | 11dni | $10^{100}$ l |

Jaki jest maksymalny rozmiar problemu, który można rozwiązać w ustalonym czasie, znając złożoność algorytmu?

| czas / T(A,n) | log n | n | n log n | $n^2$ | $n^3$ | $2^n$ |
|---|---|---|---|---|---|---|
| 1s | $2^{1000000}$ | $10^6$ | $63*10^3$ | $10^3$ | $10^2$ | 19 |
| 1H | | $36*10^8$ | $13*10^7$ | $60*10^3$ | $15*10^2$ | 31 |