ALGORYTMY I STRUKTURY DANYCH – ANALIZA ALGORYTMÓW

- **Analiza algorytmu**
  - polega na określeniu **zasobów**, jakie są potrzebne do jego wykonania
  - zasoby:
    - czas procesora
    - wielkość zajmowanej pamięci

    } złożoność obliczeniowa
- **złożoność czasowa**
  - określa zużycie czasu procesora
    - … wyrażone liczbą wykonanych operacji
    - … przy założeniu, że:
      - czas (koszt) wykonania każdej operacji jest stały, a
      - liczba operacji jest uzależniona od rozmiaru problemu (n)

  … czyli złożoność czasowa ***T(n)*** wyrażona jest w formie funkcji zależnej od rozmiaru problemu

Przykład:

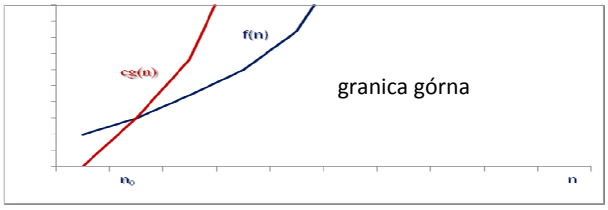
		koszt operacji	liczba wykonań operacji
1.	*wczytaj x i n*	*cR*	*2*
2.	*p ← 1*	*c←*	*1*
3.	*while (n > 0)*	*c>*	*n+1*
4.	*p ← p * x*	*c**	*n*
5.	*n ← n – 1*	*c-*	*n*
6.	*wypisz p*	*cW*	*1*

$T(n) = 2c_R + c_{\leftarrow} + c_{>}*n + c_{>} + c_{*}*n + c_{-}*n + c_W = (c_{>} + c_{*} + c_{-})*n + (2c_R + c_{\leftarrow} + c_{>} + c_W) = Cn + C = C(n+1) = n$

- **uproszczenia stosowane podczas szacowania złożoności**
  - ignorowanie rzeczywistego kosztu operacji
  - abstrakcyjne stałe
  - pominięcie stałych
  - ignorowanie mniej znaczących składników w formule
  - skoncentrowanie się na dominującej operacji

- **złożoność asymptotyczna**
  - czyli oszacowanie ***rzędu wielkości funkcji*** dla liczby operacji przy n → ∞

- **Notacja asymptotyczna**
  - Notacja O – $O(g(n)) = \{f(n): \exists\, c>0 \text{ i } n_0: 0 \le f(n) \le cg(n)\ \forall\, n \ge n_0\}$

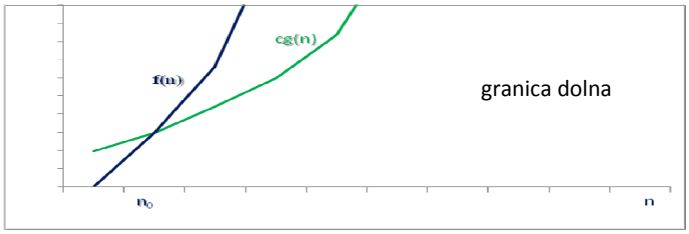


Przykład: $T(n) = ½\, n^2 + 3 = O(n^2)$, bo np. dla c = 3 ½ i n0 = 1 $0 \le ½\, n^2 + 3 \le c\, n^2 / n^2$

  - Notacja o – $o(g(n)) = \{f(n): \forall\, c>0\ \exists\, n_0>0: 0 \le f(n) < cg(n)\ \forall\, n \ge n_0\}$
    - *granica górna niedokładna*

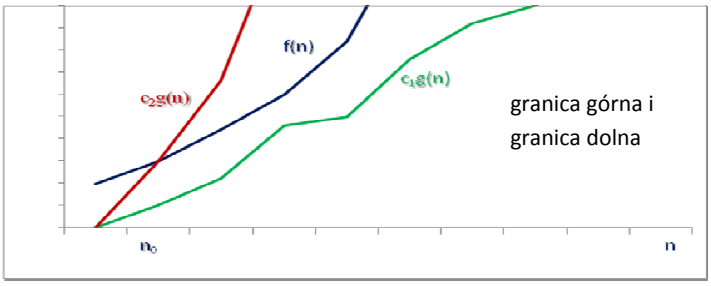
  - Uwaga:
    - $T(n) = 2\,n = o(n^2)$ ale $T(n) = 2\,n \ne o(n)$

- Notacja Ω - $\Omega(g(n)) = \{f(n): \exists\ c>0 \text{ i } n_0>0: 0 \le cg(n) \le f(n)\ \forall\ n \ge n_0\}$



- Notacja **ω** - $\omega(g(n)) = \{f(n): \forall\ c>0 \text{ i } \exists\ n_0>0: 0 \le cg(n) < f(n)\ \forall\ n \ge n_0\}$
  - *granica dolna niedokładna*
- Uwaga:
  - $T(n) = 2\,n^2 = \omega(n)$ ale $T(n) = 2\,n \ne \omega(n)$

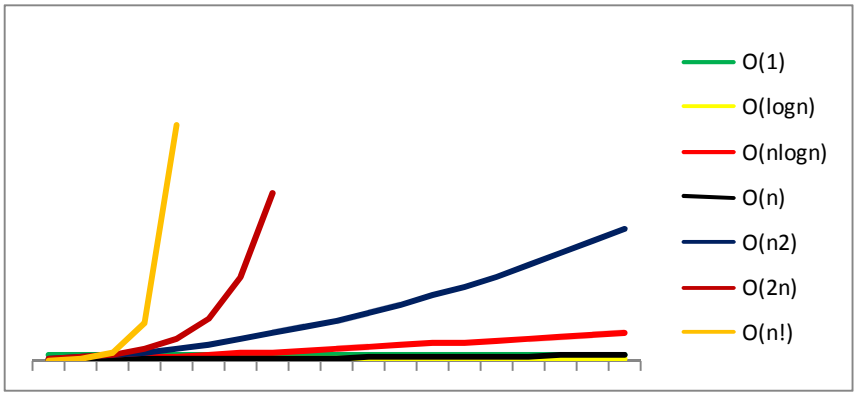
- Notacja Θ - $\Theta(g(n)) = \{f(n): \exists\ c_1>0\ c_2>0 \text{ i } n_0: 0 \le c_1 g(n) \le f(n) \le c_2 g(n)\ \forall\ n \ge n_0\}$



Złożoność obliczeniowa

- złożoność pesymistyczna ***W(n)*** *= worst case*
  - wielkość zasobów potrzebnych przy najgorszych danych
- złożoność oczekiwana ***A(n)*** *= avarage case*
  - wielkość zasobów potrzebnych przy typowych danych
- złożoność optymistyczna ***B(n)*** *= best case*
  - wielkość zasobów potrzebnych przy najlepszych danych

Typowe złożoności



Metoda rekurencji uniwersalnej

- ogólny sposób rozwiązywania równań rekurencyjnych

- Twierdzenie
  - Niech $a \ge 1$ i $b > 1$, będą stałymi, niech f(n) będzie pewną funkcją i niech T(n) będzie zdefiniowane dla nieujemnych liczb całkowitych przez rekurencję $T(n) = aT(n/b) + f(n)$.
  - Wtedy funkcja T(n) może być ograniczona asymptotycznie w następujący sposób:
    - jeśli $f(n) = O(n^{\log_b(a)-\varepsilon})$ dla pewnej stałej $\varepsilon > 0$, to $T(n) = \Theta(n^{\log_b(a)})$
    - jeśli $f(n) = \Theta(n^{\log_b(a)})$, to $T(n) = \Theta(n^{\log_b(a)}\log(n))$
    - jeśli $f(n) = \Omega(n^{\log_b(a)+\varepsilon})$ dla pewnej stałej $\varepsilon > 0$ i jeśli $af(n/b) \le cf(n)$ dla pewnej stałej $c < 1$ i wszystkich dostatecznie dużych n,to $T(n) = \Theta(f(n))$