Analiza algorytmów. Szacowanie złożoności obliczeniowej.

Złożoność obliczeniowa (czasowa) algorytmu - zależność pomiędzy rozmiarem danych wejściowych a liczbą operacji elementarnych wykonywanych w trakcie przebiegu algorytmu (podawana jako funkcja rozmiaru danych).

Operacje elementarne to np. liczba porównań dwóch liczb w algorytmie sortowania, liczba mnożeń w algorytmie obliczania silni, liczba dzieleń w algorytmie sprawdzania czy dana liczba jest pierwsza, ogólna liczba operacji arytmetycznych wykonywanych przez dany algorytm itp..

Złożoność obliczeniowa jest jednym z najważniejszych parametrów charakteryzujących algorytm. Decyduje ona o efektywności całego programu. Podstawowymi zasobami systemowymi uwzględnianymi w analizie algorytmów są czas działania (złożoność obliczeniowa algorytmu) oraz obszar zajmowanej pamięci (złożoność pamięciowa algorytmu). Złożoność pamięciowa wynika z liczby i rozmiaru struktur danych wykorzystywanych w algorytmie. Złożoność algorytmu może być rozumiana w sensie złożoności najgorszego przypadku lub złożoności średniej. Złożoność najgorszego przypadku nazywamy złożonością pesymistyczną - jest to maksymalna złożoność dla danych o zadanym rozmiarze n. Złożoność średnia lub oczekiwana to średnia wartość złożoności dla wszystkich problemów rozmiaru n. Najczęściej algorytmy mają złożoność czasową proporcjonalną do funkcji:

— $\log(n)$- złożoność logarytmiczna,
— n - złożoność liniowa,
— $n\log(n)$ - złożoność liniowo-logarytmiczna,
— n^2 - złożoność kwadratowa,
— n^k - złożoność wielomianowa, $(k > 2)$,
— 2^n - złożoność wykładnicza,
— $n!$ - złożoność wykładnicza

Notacja asymptotyczna

Rząd wielkości służy do opisu czasu działania algorytmu. Istnieją 3 notacje służące do tego celu:

Funkcja asymptotycznie niewiększa[1] od funkcji $g(n)$ to taka funkcja $f : \mathbb{N} \longrightarrow \mathbb{R}$, dla której istnieją $c > 0$ i $n_0 \in \mathbb{N}$, że $|f(n)| \leqslant c \cdot |g(n)|$ dla wszystkich $n \geqslant n_0$.

Będziemy też często mówić, że $|f(n)| \leqslant c \cdot |g(n)|$ zachodzi dla prawie wszystkich liczb naturalnych n. Zbiór funkcji asymptotycznie niewiększych niż $g(n)$ oznaczamy przez $O(g(n))$.

Funkcja asymptotycznie niemniejsza od funkcji $g(n)$ to taka funkcja $f : \mathbb{N} \longrightarrow \mathbb{R}$, dla której istnieją $c > 0$ i $n_0 \in \mathbb{N}$, że $c \cdot |g(n)| \leqslant |f(n)|$ dla wszystkich $n \geqslant n_0$.

[1] na podstawie materiałów `http://mediawiki.ilab.pl/index.php/Matematyka_dyskretna_1/Wyk%C5%82ad_9:_Asymptotyka`

Zbiór funkcji asymptotycznie niemniejszych niż $g(n)$ oznaczamy przez $\Omega(g(n))$.

Funkcja asymptotycznie podobna do funkcji $g(n)$ to taka funkcja $f : \mathbb{N} \longrightarrow \mathbb{R}$, dla której istnieją $c_0, c_1 > 0$ i $n_0 \in \mathbb{N}$, że $c_0 \cdot |g(n)| \leqslant |f(n)| \leqslant c_1 \cdot |g(n)|$ dla wszystkich $n \geqslant n_0$.

Zbiór funkcji asymptotycznie podobnych do $g(n)$ oznaczamy przez $\Theta(g(n))$. A zatem $\Theta(g(n)) = O(g(n)) \cap \Omega(g(n))$.

Odpowiednio w stosunku do wcześniej wypisanych funkcji powiemy, że złożoność obliczeniowa algorytmu wynosi (lub algorytm „działa w czasie") $O(log(n))$, $O(n)$, $O(n \log(n))$, itd.

Gdzie np. $\log n \in O(n)$, $n^2 \in \Omega(n)$, $n \in O(n)$, $n \in \Omega(n)$, $n \in \Theta(n)$, $20n \in \Theta(n)$.

Zadania na ćwiczenia

Zadanie 1

Uporządkować rosnąco wg rzędu następujące funkcje:
$2^{n!}$, n^2 , $(2009!)n^2$, $n^3 - n^2$, n^3 , $n^3 + n^2$, 2^n , $n^{2009!}$, $n!$, $n \log n$, $\log n$.
Czy któreś z tych funkcji są równoważne według rzędu (są asymptotycznie podobne)?

Znajdź i wykaż (udowodnij) zależności pomiędzy funkcjami w notacji asymptotycznej:
— $2n$ w odniesieniu do $(20!)n$,
— $n^2 - n$ w odniesieniu do $n^2 + n$,
— $\log_2 n$ w odniesieniu do n,
— 3^n w odniesieniu do 2^n
— $n!$ w odniesieniu do 2^n

Zadanie 2

Dla danej liczby $n \in \mathbb{N}$ należy wyznaczyć wszystkie liczby pierwsze $p \leqslant n$. Porównać złożoność obliczeniową dwóch algorytmów realizujących to zadanie:
— sprawdzamy kolejno dla każdej liczby naturalnej $2 \leqslant k \leqslant n$ czy jest liczbą pierwszą,
— sito Eratostenesa z listy $2, ..., n$ wykreślamy kolejno: liczby podzielne przez 2, liczby podzielne przez 3,...,
Czy można oba algorytmu zaimplementować w taki sposób, aby ich złożoności były równe/różne ?

Zadanie 3

Dany jest wielomian $W(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_{n-1} x^{n-1} + a_n x^n$. Chcemy obliczyć wartość tego wielomianu w punkcie x_0, możemy zrobić to na dwa sposoby:
— obliczając wprost wartość $W(x_0) = a_0 + a_1 x_0 + a_2 x^2 + \ldots + a_n x^n$

— korzystając z tzw. *schematu Hornera* przekształcając wielomian do postaci

$$\begin{aligned} W(x) &= a_0 + x(a_1 + a_2x + \ldots + a_{n-1}x^{n-2} + a_nx^{n-1}) = \\ &= a_0 + x(a_1 + x(a_2 + \ldots + a_{n-1}x^{n-3} + a_nx^{n-2})) = \\ &= a_0 + x(a_1 + x(a_2 + \ldots + x(a_{n-1}x + a_n)\ldots)) \end{aligned}$$

Wyznaczyć złożoność obliczeniową algorytmu w stosunku do tego, gdy jako operacje elementarne przyjmiemy a) liczę dodawań, b) liczbę mnożeń, c)liczbę wszystkich operacji arytmetycznych.

Zadanie 4

Dla algorytmów z zadań 2-5 z poprzednich zajęć oszacować ich złożoność obliczeniową.

Informacje końcowe

Jedno zadanie na kolokwium będzie obejmowało uporządkowanie funkcji wg rzędu i wskazanie zależności asymptotycznej pomiędzy wybranymi parami oraz jedno zadanie na kolokwium będzie obejmowało oszacowanie złożoności obliczeniowej jakiegoś algorytmu/algorytmów.