Złożoność obliczeniowa algorytmu

Złożoność algorytmów

Niezależnie od rozpatrywanego rodzaju złożoności obliczeniowej, tj. złożoności czasowej czy też złożoności pamięciowej; u podstaw znajduje się teoria będąca częścią wspólną dla obu wymienionych zagadnień. Celem złożoności obliczeniowej jest de-facto możliwość oszacowania teoretycznej pesymistycznej, średniej oraz optymistycznej wydajności danego algorytmu. Aby istniała możliwość szacowania złożoności obliczeniowej, koniecznym było zdefiniowanie wspólnej miary algorytmów, która byłaby niezależna od zastosowanego języka programowania, sprzętu czy też kompilatora. Miara taka została zdefiniowana i jest nią liczba danych wejściowych przekazywanych do algorytmu.

Złożoność czasowa

Kluczową rolę w dziedzinie algorytmiki pełni złożoność czasowa algorytmów. Za pomocą niej określamy jak długo dany algorytm musi pracować aby wykonał swoje zadanie. Czas pracy algorytmu nie jest wyrażany w sekundach, lecz w jednostkach, którym nie jest przypisana żadna rzeczywista miara. Czas pracy algorytmu jest więc w praktyce liczbą, której wartość jest uzależniona od liczby danych wejściowych oraz zasad opisujących sposób ich przetwarzania przez dany algorytm. Warto również wiedzieć, że złożoność czasową algorytmów oznacza się zawsze dużą literą **O**.

Złożoność pamięciowa

Celem złożoności pamięciowej jest oszacowanie zużycia pamięci przez dany algorytm. Złożoność pamięciową podobnie jak w przypadku złożoności obliczeniowej szacuje się na podstawie liczby danych wejściowych, przekazanych do algorytmu oraz sposobu ich przetwarzania przez dany algorytm. W przypadku szacowania złożoności pamięciowej rozpatruje się tylko i wyłącznie pamięć, którą należy dodatkowo zaalokować w trakcie pracy algorytmu tak, aby było możliwe jego wykonanie zgodnie z zasadami określającymi sposób działania algorytmu. Do złożoności pamięciowej nie wlicza się więc rozmiaru danych wejściowych, które zostały przekazane do danego algorytmu. Na koniec warto jeszcze wspomnieć o jednostce miary stosowanej do mierzenia zużycia pamięci. W przypadku

szacowania złożoności pamięciowej, za miarę generalnie przyjmuje się liczbę pamięci RAM, jaką należy dodatkowo zaalokować na abstrakcyjnej maszynie. Zapotrzebowanie pamięci RAM w teorii złożoności obliczeniowej wyrażane jest więc w bitach lub bajtach.

Ponieważ często zużycie zasobów w algorytmie uzależnione jest od postaci przetwarzanych danych, zarówno złożoność czasowa jak i pamięciowa może występować w trzech odmianach:

- $T_O(n)$ - optymistycznej (ang. optimistic)
- $T_A(n)$ - średniej (ang. average)
- $T_W(n)$ - pesymistycznej (ang. worst)

Aby poglądowo wyjaśnić powyższe terminy, rozważmy prosty algorytm wyszukiwania robaczywego jabłka w koszu n jabłek. Algorytm jest bardzo prosty:

Dopóki w koszu są jabłka, wyjmij jedno jabłko z kosza, obejrzyj je, jeśli jest robaczywe, to zakończ. Inaczej odłóż je na bok i wróć do początku.

Rozważmy, ile operacji dominujących (ocena jabłka) wykona ten algorytm dla n jabłek.

1. Zakładamy przypadek optymistyczny - robaczywe jabłko napotkamy za pierwszym razem. Zatem:

   $T_o(n) = 1$ - złożoność optymistyczna

2. Zakładamy przypadek najgorszy - w koszu brak jabłek robaczywych lub robaczywe jabłko będzie wyjęte z kosza jako ostatnie:

   $T_w(n) = n$ - złożoność pesymistyczna

3. W przypadku typowym robaczywe jabłko znajdziemy po przeglądnięciu połowy jabłek w koszu - tzn. raz będzie ono wyciągnięte wcześniej, raz później, a średnio w $^n/_2$ teście:
4. $T_A(n) = {}^n/_2$ - złożoność średnia, oczekiwana

Zatem podsumowując:

Złożoność optymistyczna określa zużycie zasobów dla najkorzystniejszego zestawu danych.

Złożoność średnia określa zużycie zasobów dla typowych (tzw. losowych) danych.

Złożoność pesymistyczna określa zużycie zasobów dla najbardziej niekorzystnego zestawu danych.

Klasy złożoności

Złożoność czasowa	Opis	Szybkość algorytmu
O(**1**)	**stała złożoność** – algorytm wykonuje się w stałej ilości operacji, niezależnie od liczby danych wejściowych.	najszybszy
O(**$\log_2(n)$**)	**złożoność logarytmiczna** – algorytm wykonuje logarytmiczną ilość operacji w stosunku do liczby danych wejściowych. **n** jest to liczba danych wejściowych; podstawa logarytmu zazwyczaj wynosi **2**, jednak czasami może być większa (np. w B-drzewach).	bardzo szybki
O(**n**)	**złożoność liniowa** – algorytm wykonuje wprost proporcjonalną ilość operacji do liczby danych wejściowych. **n** jest to liczba danych wejściowych.	szybki
O(**$n*\log_2(n)$**)	**złożoność liniowo-logarytmiczna** **n** jest to liczba danych wejściowych.	szybki
O(**n^2**)	**złożoność kwadratowa** – ilość operacji algorytmu jest wprost proporcjonalna do liczby danych wejściowych podniesionej do potęgi drugiej. **n** jest to liczba danych wejściowych.	niezbyt szybki
O(**n^X**)	**złożoność wielomianowa** – ilość operacji algorytmu jest wprost proporcjonalna do liczby danych wejściowych podniesionej do potęgi **X**. **n** jest to liczba danych wejściowych; **X** jest stałą o dowolnej wartości. **Uwaga!** Pamiętaj, że zarówno funkcja liniowa jak i funkcja kwadratowa są również wielomianami. W informatyce posługując się terminem złożoności wielomianowej zazwyczaj mamy na myśli algorytmy, których złożoność obliczeniowa (lub pamięciowa) jest co najmniej kwadratowa.	wolny

O(**X^n**)	**złożoność wykładnicza** – ilość operacji algorytmu jest wprost proporcjonalna do stałej **X** większej lub równej **2**, podniesionej do potęgi równej liczbie danych wejściowych. **n** jest to liczba danych wejściowych; **X** jest stałą większą niż **2**.	

Podobnie jak złożoność obliczeniowa, klasa złożoności obliczeniowej może być optymistyczna, typowa lub pesymistyczna.

Znajomość klas złożoności obliczeniowej czasowej i pamięciowej dla różnych algorytmów pozwala informatykowi przewidywać zachowanie się tych algorytmów dla różnych zestawów danych oraz dobierać algorytmy dla określonych sytuacji. Dlatego jest to jedno z kluczowych pojęć algorytmiki.