Politechnika Wrocławska, Wydział Informatyki i Zarządzania
Zbigniew Szpunar

# Wykład 2.
# Złożoność obliczeniowa algorytmów

Politechnika Wrocławska, Wydział Informatyki i Zarządzania
Zbigniew Szpunar

# Złożoność obliczeniowa algorytmów

Programiści dążą do opracowywania wydajnych algorytmów. Naturalnym kryterium porównania algorytmów wydaje się więc być różnie rozumiana ich efektywność.

Stosuje się różne miary efektywności. Najczęściej rozpatruje się ilość zasobów, które są używane do rozwiązania problemu, w tym głównie:

- czas obliczeń (przekładający się na zużycie czasu procesora),
- wielkość zajętej pamięci operacyjnej,
- wielkość zajętej pamięci dyskowej.

Porównanie dwóch algorytmów na podstawie zużycia zasobów w przykładowych procesach obliczeniowych nie prowadzi do obiektywnych, jednoznacznie poprawnych wniosków.

Politechnika Wrocławska, Wydział Informatyki i Zarządzania
Zbigniew Szpunar

## Złożoność obliczeniowa algorytmów

Zużycie zasobów często niewiele mówi o samych algorytmach, ponieważ jest ono zależne od wielu czynników, np.:

- od szybkości procesora (i innych układów),
- od szybkości pamięci (operacyjnej, dyskowej),
- od jakości kodu generowanego przez kompilator.

Trudno więc na podstawie zużycia zasobów porównywać algorytmy. Wówczas wprowadza się pojęcie złożoności obliczeniowej.

Złożoność obliczeniową można traktować jako miarę ilości zasobów wymaganych przez algorytm do rozwiązania problemu o określonym rozmiarze.

W zależności od rozpatrywanego zasobu mamy złożoność:

- czasową,
- pamięciową.

Politechnika Wrocławska, Wydział Informatyki i Zarządzania
Zbigniew Szpunar

## Złożoność obliczeniowa algorytmów – złożoność czasowa

Złożoność czasowa: miarą złożoności jest liczba operacji podstawowych w zależności od rozmiaru wejścia.

Pomiar czasu zegarowego ma ograniczoną użyteczność, ze względu na jego silną zależność od implementacji algorytmu, kompilatora, maszyny, umiejętności programisty.

Dlatego w charakterze "czasu wykonania" rozpatruje się zazwyczaj liczbę operacji podstawowych (dominujących), np.: podstawienie, pobranie, porównanie, mnożenie, ...

## Złożoność obliczeniowa algorytmów – złożoność czasowa

Kolejne problemy, to:

- w jakim języku programowania formułować algorytmy,
- jakie przyjąć założenie o maszynie, na której algorytm będzie wykonywany (komputery różnią się istotnymi – z punktu widzenia konstruowania algorytmów – parametrami, np. liczbą i rozmiarem rejestrów, repertuarem operacji matematycznych, ponadto ciągle są ulepszane).

Dlatego algorytmy analizuje się wykorzystując abstrakcyjne modele obliczeń, np.:

- maszynę RAM,
- maszynę Turinga,
- ...

Politechnika Wrocławska, Wydział Informatyki i Zarządzania
Zbigniew Szpunar

## Złożoność obliczeniowa algorytmów – złożoność pamięciowa

Złożoność pamięciowa: miarą złożoności jest ilość wykorzystanej pamięci w zależności od rozmiaru wejścia.

W charakterze "ilości pamięci" przyjmuje się najczęściej użytą pamięć maszyny abstrakcyjnej (np. liczbę komórek pamięci maszyny RAM) w funkcji rozmiaru wejścia.

Można też obliczać rozmiar potrzebnej pamięci fizycznej wyrażonej w bitach/bajtach.

Politechnika Wrocławska, Wydział Informatyki i Zarządzania
Zbigniew Szpunar

## **Złożoność obliczeniowa algorytmów -** szacowanie złożoności

Złożoność przeciętna (średnia, oczekiwana)

dla przeciętnego przypadku danych wejściowych

Złożoność optymistyczna (*best-case*)

dla najbardziej korzystnego (najlepszego) przypadku danych wejściowych

Złożoność pesymistyczna (*worst-case*)

dla najmniej korzystnego (najgorszego) przypadku danych wejściowych.

Przykład z sortowaniem liczb algorytmem sortowania bąbelkowego, w aspekcie złożoności czasowej:

Przypadek najkorzystniejszy: liczby są już posortowane tak, jak trzeba,

Przypadek najgorszy: liczby są posortowane "przeciwnie",

Przypadek przeciętny: ani najkorzystniejszy, ani najgorszy - taki "średni".

Politechnika Wrocławska, Wydział Informatyki i Zarządzania
Zbigniew Szpunar

## Porównywanie złożoności obliczeniowej algorytmów

Porównując złożoność algorytmów bierze się pod uwagę asymptotyczne tempo wzrostu,

czyli :

jak zachowuje się funkcja określająca złożoność dla odpowiednio dużych, granicznych argumentów (rozmiarów danych wejściowych), ignorując zachowanie dla małych danych.

Asymptotyczne tempo wzrostu – miara określająca zachowanie wartości funkcji wraz ze wzrostem jej argumentów.

Miara jest stosowana często w teorii obliczeń dla opisu złożoności obliczeniowej, czyli zależności ilości potrzebnych zasobów (np. czasu lub pamięci) od rozmiaru danych wejściowych algorytmu.

Asymptotyczne tempo wzrostu opisuje jak szybko dana funkcja rośnie lub maleje, abstrahując od konkretnej postaci tych zmian.

Politechnika Wrocławska, Wydział Informatyki i Zarządzania
Zbigniew Szpunar

# Złożoność obliczeniowa algorytmów

Do opisu asymptotycznego tempa wzrostu stosuje się między innymi notację dużego O (tzw. notacja Landaua).

- $f(x) = O(1)$ – funkcja $f(x)$ jest ograniczona
- $f(x) = O(\log n)$ – funkcja $f(x)$ jest ograniczona przez funkcję logarytmiczną
- $f(x) = O(n)$ – funkcja $f(x)$ jest ograniczona przez funkcję liniową
- $f(x) = O(n \log n)$
- $f(x) = O(n^k)$ – funkcja $f(x)$ jest ograniczona przez funkcję potęgową lub wielomian
- $f(x) = O(a^n)$ – funkcja $f(x)$ jest ograniczona przez funkcję wykładniczą
- $f(x) = O(n!)$ – funkcja $f(x)$ jest ograniczona przez silnię

Politechnika Wrocławska, Wydział Informatyki i Zarządzania
Zbigniew Szpunar

## Złożoność obliczeniowa algorytmów

Najczęstszym zastosowaniem asymptotycznego tempa wzrostu jest szacowanie złożoności problemów obliczeniowych, w szczególności algorytmów. Oszacowanie rzędów złożoności obliczeniowej funkcji pozwala na porównywanie ilości zasobów (np. czasu, pamięci), jakich wymagają do rozwiązania problemu opisanego określoną ilością danych wejściowych.

W dużym uproszczeniu można powiedzieć, że im niższy rząd złożoności obliczeniowej algorytmu, tym będzie on wydajniejszy.

W praktyce na efektywność algorytmu wpływa duża ilość innych czynników, w tym szczegóły realizacji.

Ponadto, dla małych danych wejściowych asymptotyczne tempo wzrostu może nie oddawać zachowania funkcji (funkcja może rosnąć szybciej, ale dla danego rozmiaru problemu wartość tej funkcji może być mniejsza, niż funkcji rosnącej liniowo).

Politechnika Wrocławska, Wydział Informatyki i Zarządzania
Zbigniew Szpunar

# Złożoność obliczeniowa algorytmów

| | |
|---|---|
| $1$ | stała |
| $log_2 n$ | logarytmiczna |
| $n$ | liniowa |
| $nlog_2 n$ | liniowo-logarytmiczna (lub quasi-liniowa) |
| $n^2$ | kwadratowa |
| $n^c$ | wielomianowa |
| $c^n$ | wykładnicza |

Przy szacowaniu złożoności obliczeniowej zazwyczaj pomija się współczynniki proporcjonalności, ponieważ ich znaczenie przy dużych rozmiarach problemu maleje.

Politechnika Wrocławska, Wydział Informatyki i Zarządzania
Zbigniew Szpunar

# Złożoność obliczeniowa algorytmów

Ilustracja tempa wzrostu funkcji służących do oszacowania złożoności obliczeniowej algorytmów:

| $\lg n$ | $n$ | $n \lg n$ | $n (\lg n)^2$ | $n^2$ | $2^n$ |
|---|---|---|---|---|---|
| 3 | 10 | 33 | 110 | 100 | 1024 |
| 7 | 100 | 664 | 4414 | 10000 | 1 i 30 zer |
| 10 | 1000 | 9966 | 99317 | 1000000 | 1 i 300 zer |
| 13 | 10000 | 132877 | 1765633 | 100000000 | 1 i 3000 zer |
| 17 | 100000 | 1660964 | 27588016 | 10000000000 | 1 i 30000 zer |
| 20 | 1000000 | 19931569 | 397267426 | 1000000000000 | 1 i 300000 zer |

Politechnika Wrocławska, Wydział Informatyki i Zarządzania
Zbigniew Szpunar

# Złożoność obliczeniowa algorytmów

Ilustracja tempa wzrostu funkcji służących do oszacowania złożoności obliczeniowej algorytmów, inaczej:

| Sekundy | → Inaczej |
|---|---|
| 100 | ~2 min. |
| 10 000 | ~3 godz. |
| 100 000 | ~1.1 dnia |
| 1 000 000 | ~1.5 tyg. |
| 10 000 000 | ~4 m-ce |
| 100 000 000 | ~3 lata |
| 1 000 000 000 | ~30 lat |

Politechnika Wrocławska, Wydział Informatyki i Zarządzania
Zbigniew Szpunar

# Złożoność obliczeniowa algorytmów

Złożoność obliczeniowa stała:

Utożsamiana z algorytmem wykonującym stałą liczbę operacji, bez względu na rozmiar problemu.
*Algorytm o stałej złożoności nie musi być algorytmem szybkim (w sensie czasu trwania obliczeń)! Może np. liczyć, dla każdego rozmiaru problemu, przez dziesiątki, setki, ... godzin.*

Złożoność liniowa:

Czas wykonania algorytmu jest proporcjonalny do rozmiaru problemu (np. proste sumowanie liczb w ciągu, liczenie średniej arytmetycznej, średniej geometrycznej, itp.).
*Skrócenie czasu wykonania elementarnej operacji skraca łączny czas wykonania algorytmu, ale nie zmniejsza jego złożoności obliczeniowej.*
Są to tzw. algorytmy akceptowalne.

Politechnika Wrocławska, Wydział Informatyki i Zarządzania
Zbigniew Szpunar

# Złożoność obliczeniowa algorytmów

Złożoność kwadratowa:

Czas wykonania algorytmu jest proporcjonalny do kwadratu rozmiaru problemu (np. porównanie każdej pary liczb w zbiorze, szukanie najkrótszego odcinka łączącego parę punktów na płaszczyźnie poprzez pełny przegląd możliwych par, itp.).

Złożoność logarytmiczna:

Czas wykonania algorytmu jest proporcjonalny do logarytmu z rozmiaru problemu.
Inaczej: geometryczny wzrost rozmiaru problemu powoduje arytmetyczny wzrost ("przyrost") złożoności obliczeniowej.
*Np. w algorytmach wyszukiwania binarnego uzyskuje się taką złożoność dzięki pominięciu znaczącej liczby danych wejściowych.*

Politechnika Wrocławska, Wydział Informatyki i Zarządzania
Zbigniew Szpunar

# Złożoność obliczeniowa algorytmów

## Złożoność rzędu N!:

Dla kilku początkowych wartości *N* funkcja *N!* rośnie wolniej, niż N-kwadrat, ale potem zaczyna rosnąć lawinowo. Podobnie, jak złożoność kwadratowa, dyskwalifikuje algorytm dla dużych rozmiarów problemu.

Politechnika Wrocławska, Wydział Informatyki i Zarządzania
Zbigniew Szpunar

# Złożoność obliczeniowa algorytmów

Problem NP
(niedeterministycznie wielomianowy, *nondeterministic polynomial*)

Problem decyzyjny, dla którego rozwiązanie można zweryfikować w czasie wielomianowym.
Problem jest w klasie NP, jeśli może być rozwiązany w wielomianowym czasie na niedeterministycznej maszynie Turinga.

Problem P

Problem decyzyjny, dla którego można znaleźć rozwiązanie w czasie wielomianowym (a więc ma wielomianową złożoność obliczeniową).

Różnica polega na tym, że dla problemu NP samo sprawdzenie podanego „z zewnątrz” rozwiązania ma mieć złożoność wielomianową.

Politechnika Wrocławska, Wydział Informatyki i Zarządzania
Zbigniew Szpunar

# Złożoność obliczeniowa algorytmów

Przykład problemu NP:

*Znaleźć niepusty podzbiór zadanego zbioru liczb, który sumuje się do zera.*

Trudno znaleźć rozwiązanie tego zagadnienia w czasie wielomianowym. Nasuwający się algorytm sprawdzenia wszystkich możliwych podzbiorów ma złożoność wykładniczą ze względu na liczebność zbioru. Nie wiadomo zatem, czy problem ten jest klasy P.
Natomiast mając „z zewnątrz" kandydata na rozwiązanie (np. {-2,6,-3,-1}) możemy w liniowym (a zatem wielomianowym) czasie sprawdzić, czy sumuje się do zera. Jest to zatem problem NP.

Wszystkie problemy klasy P są problemami klasy NP, ponieważ można je sprawdzić w czasie wielomianowym.

Nie wiadomo natomiast, czy istnieje problem NP, który nie jest w klasie P. Jest to jedno z wielkich nierozwiązanych zagadnień teoretycznych współczesnej informatyki.

Politechnika Wrocławska, Wydział Informatyki i Zarządzania
Zbigniew Szpunar

# Złożoność obliczeniowa algorytmów

## Problemy NP-zupełne

Problem NP-zupełny to taki problem, do którego może zostać zredukowany każdy inny problem klasy NP.

Nie ma tradycyjnej definicji problemu NP zupełnego. Przyjmuje się, że problem jest NP zupełny gdy można go sprowadzić do innego problemu uznawanego za NP zupełny za pomocą obliczeń, które można wykonać w czasie wielomianowym.

Taka definicja problemów NP zupełnych implikuje fakt, że jeśli tylko potrafimy rozwiązać jakikolwiek problem NP zupełny w czasie wielomianowym, to potrafimy rozwiązać w czasie wielomianowym wszystkie problemy NP zupełne.

Politechnika Wrocławska, Wydział Informatyki i Zarządzania
Zbigniew Szpunar

# Złożoność obliczeniowa algorytmów

Sztuka programowania polega między innymi na poszukiwaniu takiego rozwiązania, by uzyskany algorytm miał możliwie najkorzystniejszą złożoność obliczeniową.