**Projektowanie Algorytmów i Metody Sztucznej Inteligencji**

Projekt I

Paweł Prucnal 248937

26.03.2020r.

Piątek 1315 mgr inż. Marta Emirsajłow

# Wprowadzenie

Celem projektu było zapoznanie się z popularnymi algorytmami sortowania stosowanymi w informatyce. Należało zrozumieć, opracować, a następnie zaimplementować trzy wybrane algorytmy sortowania. Następnie, w celu ich głębszego poznania, przeprowadzić szereg testów uwidaczniających ich wady oraz zalety. Pozwoliło to zrozumieć, że żaden ze znanych nam algorytmów nie jest idealny oraz że żaden z nich nie jest najszybszy w każdym przypadku.

# Przebieg eksperymentów

Testy polegały na zmierzeniu czasu sortowania losowo wygenerowanych tablic o różnych rozmiarach dla każdego z badanych algorytmów. By tego dokonać, zaimplementowano je w prostym programie w języku C++, którego zadaniem było wielokrotne deklarowanie, wypełnianie oraz sortowanie tablic. Przed rozpoczęciem sortowania oraz natychmiast po jego zakończeniu pobierał on dokładny czas z komputera przy pomocy biblioteki ctime. Następnie obliczał on różnicę między tymi dwoma punktami czasu w milisekundach i zapisywał ją do pliku. Proces ten wykonywany był sto razy dla każdej z kombinacji warunków. Należało bowiem sprawdzić jak algorytm poradzi sobie z tablicami o rozmiarach 10000, 50000, 100000, 500000 oraz 1000000 elementów, jak i tablicami nieposortowanymi, posortowanymi w 25%, 50%, 75%, 95%, 99%, 99,7% oraz posortowanymi, a następnie odwróconymi. Przy trzech badanych algorytmach daje to razem 120 kombinacji, dla każdej z których zostało przeprowadzone 100 pomiarów. Wyniki spisane w plikach zaimportowano do programu Microsoft Excel w celu sporządzenia wykresów.

Przeprowadzono testy na algorytmach:

* Sortowania przez scalanie (mergesort),
* Sortowania szybkiego (quicksort),
* Sortowania introspektywnego (introsort).

Otrzymano następujące wyniki pomiarów:



# Omówienie badanych algorytmów

Badane algorytmy znane są ze swojej wysokiej wydajności, zwłaszcza sortowanie szybkie oraz introspektywne.

Sortowanie przez scalanie polega na rozbijaniu tablicy na dwie części do momentu uzyskania zbioru jednoelementowego, który jest zawsze posortowany. Następnie rozbity zbiór jest scalany już z posortowanymi elementami. Dzięki temu problem posortowania tablicy rozbijany jest na szereg prostszych problemów. Złożoność tego algorytmu zarówno dla średniego jak i najgorszego przypadku powinna wynosić O(*n* log *n*).

Sortowanie szybkie (quicksort) polega na wybraniu jednego elementu z tablicy (punktu odniesienia) a następnie porównaniu go z każdym elementem z jego lewej oraz prawej strony. Gdy znaleziona zostanie para elementów, w której element z lewej strony jest większy od punktu odniesienia oraz element z prawej strony jest mniejszy od punktu odniesienia, zamieniane są one miejscami. Gdy analizowane indeksy przekroczą punkt odniesienia, utworzą one 3 tablice, z których środkowa jest już posortowana. Na pozostałych wykonywane są ponownie powyższe operacje. Quicksort charakteryzuje się złożonością obliczeniową rzędu O(*n* log *n*) dla przypadku średniego oraz O(n2) dla przypadku najgorszego.

Sortowanie introspektywne jest hybrydą sortowania szybkiego, sortowania przez kopcowanie (heapsort) oraz sortowania przez wstawianie (insertionsort). Poprzez „przełączanie” na algorytm sortowania przez kopcowanie gdy głębokość rekursji przekroczy maksymalną wartość, pozwala on ominąć najgorszy przypadek sortowania szybkiego O(n2). Stosuje on również sortowanie przez wstawianie dla niedużych zbiorów danych (np. 16 elementów), gdzie quicksort poradziłby sobie znacznie gorzej. Sortowanie introspektywne charakteryzuje się złożonością obliczeniową rzędu O(*n* log *n*) zarówno dla przypadku najgorszego, jak i średniego.

# Podsumowanie wyników i wnioski

Powyższe wykresy świetnie obrazują najgorszy przypadek sortowania szybkiego oraz introspektywnego. Podanie na wejściu tych algorytmów tablicy posortowanej do połowy znacznie wydłuża czas sortowania. Porównując te pomiary do reszty (np. 25%) można założyć, iż w przypadku quicksorta ten pomiar przypomina bardziej funkcję wykładniczą, gdy reszta- funkcję logarytmiczną. Zastosowanie sortowania introspektywnego zmniejszyło średni czas w tym przypadku niemal o połowę. Najgorszy czas okazał się natomiast około dziesięć razy krótszy w porównaniu do najgorszego czasu zmierzonego dla sortowania szybkiego.

# Literatura

* [www.geeksforgeeks.org](http://www.geeksforgeeks.org),
* [www.wikipedia.org](http://www.wikipedia.org),
* [www.youtube.com](http://www.youtube.com), w szczególności kanały kakaboc oraz Coding Blocks,
* [www.algorytm.edu.pl](http://www.algorytm.edu.pl).