Politechnika wrocławska
Wydział Elektroniki, Fotoniki i Mikrosystemów
Projektowanie Algorytmów i Metody Sztucznej Inteligencji
Automatyka i Robotyka

# Projekt 1

*Autor:* Jakub Jankowiak
*Nr. indeksu:* 258965
*Grupa:* Y03-51a, Pn 15:15
*Prowadzący:* Mgr inż. Marta Emirajsłow

27 marca 2022

# Spis treści


1 Wprowadzenie . . . . . 2
2 Złożoność obliczeniowa . . . . . 2
3 Omównienie przebiegu eksperymentu . . . . . 2
4 Sortowanie introspektywne . . . . . 2
4.1 Opis . . . . . 2
4.2 Złożoność obliczeniowa . . . . . 3
4.3 Wyniki . . . . . 3
4.4 Wykres . . . . . 3
5 Sortowanie przez scalanie . . . . . 4
5.1 Opis . . . . . 4
5.2 Złożoność obliczeniowa . . . . . 4
5.3 Wyniki . . . . . 4
5.4 Wykres . . . . . 5
6 Sortowanie szybkie . . . . . 5
6.1 Opis . . . . . 5
6.2 Złożoność obliczeniowa . . . . . 6
6.3 Wyniki . . . . . 6
6.4 Wykres . . . . . 6
7 Wnioski . . . . . 7
8 Bibliografia . . . . . 7

# 1 Wprowadzenie

Sortowanie to uporządkowanie określonych danych według poszczególnych cech takich jak na przykład w przypadku liczb od najmniejszej do największej. Zazwyczaj komputery wykonują tę czynność wykorzystując do tego konkretne algorytmy. Owe algorytmy różnią się pod względem złożoności obliczeniowej, stabliności i sposobie działania. W tym ćwiczeniu zostaną zrealizowane 3 typy algorytmów sortujących: sortowanie przez scalanie, sortowanie szybkie, sortowanie introspektywne.

# 2 Złożoność obliczeniowa

Złożoność obliczeniowa określa wydajność algorytmu, jest to liczba operacji potrzebnych do rozwiązania danego problemu oraz ilość pamięci jakiej wymaga. Dzielimy ją na pamięciową i czasową. Złożoność zwykle nie zależy od wielkości danych, ale może się znacznie różnić dla danych o tej samej wielkości. W takich przypadkach używamy metody najgorszego przypadku (pesymistyczna złożoność) i określanie sposobu uśrednienia każdego możliwego przypadku (oczekiwana złożoność).

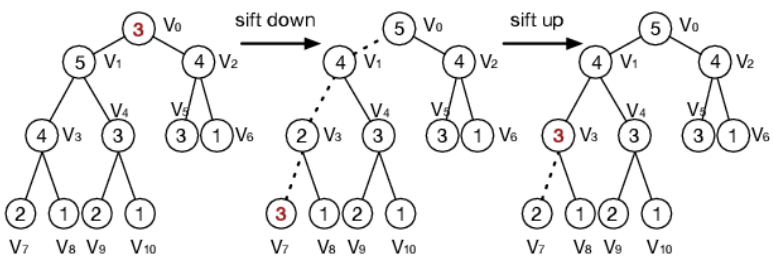
# 3 Omównienie przebiegu eksperymentu

Eksperyent przeprowadzono na stu tabilcach zawierających liczby całkowite. Dla każdej z poniższych ilości elementów zadano odpowiednie dla badnia warunki początkowe i przeprowadzono wybrane sortowanie. Dla pewności, że podczas sortowania nie wystąpił błąd, dla każdej tablicy została wywołana funkcja, która sprawdza poprawność sortowania. Podczas każdego sortowania dodatkowo był mierzony czas działania algorytmu.

# 4 Sortowanie introspektywne

## 4.1 Opis

Głównym założeniem algorytmu Sortowania Introspektywnego jest obsługa najgorszego przypadku algorytmu Sortowania Szybkiego tak, aby zapewnić logarytmiczno-liniową złożoność obliczeniową. Przypomnijmy, że w najgorszym przypadku podziały wykonywane przez procedurę Partition były zdegenerowane i algorytm Quick Sort wykonywał O(n2) porównań.




Rysunek 1: Sortowanie introspektywne

## 4.2 Złożoność obliczeniowa

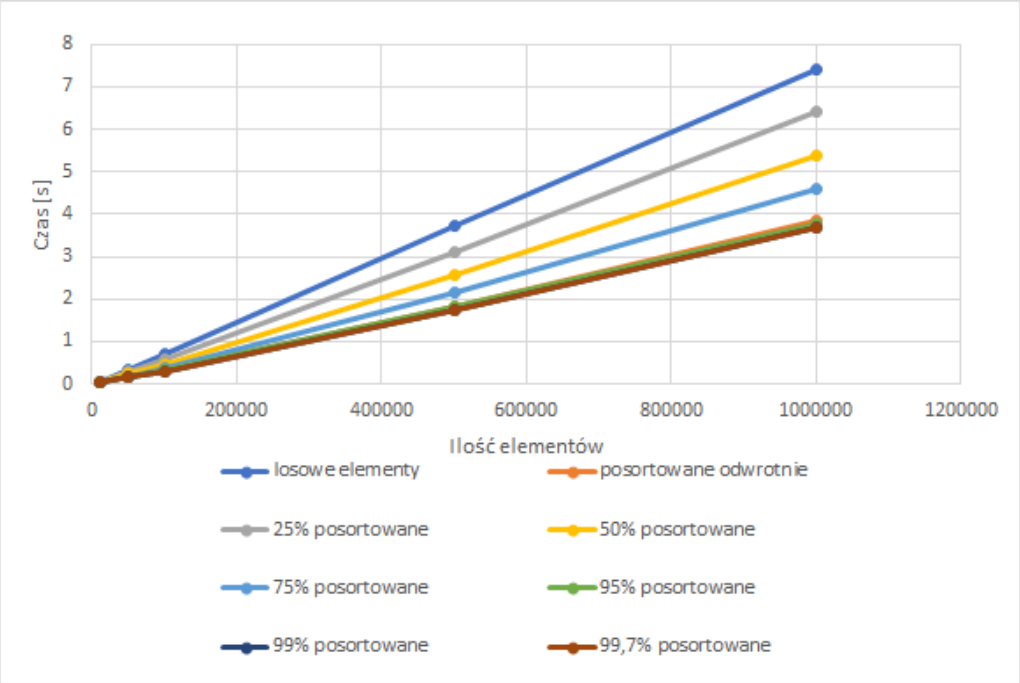
- Złożóność pamięciowa: $\Theta(logn)$
- Złożoność czasowa: $\Theta(n \cdot logn)$
- Złożoności pesymistyczna, średnia, optymistyczna: $\Theta(n \cdot logn)$

## 4.3 Wyniki

| ilość elementów | losowe elmenty | posortowane odwrotnie | 25% posortowane | 50% posortowane | 75% posortowane | 95% posortowane | 99% posortowane | 99,7% posortowane |
|---|---|---|---|---|---|---|---|---|
| 10000 | 0,062 | 0,031 | 0,035 | 0,052 | 0,042 | 0,042 | 0,032 | 0,032 |
| 50000 | 0,33 | 0,166 | 0,279 | 0,235 | 0,184 | 0,16 | 0,158 | 0,149 |
| 100000 | 0,691 | 0,317 | 0,584 | 0,47 | 0,375 | 0,319 | 0,309 | 0,308 |
| 500000 | 3,746 | 1,804 | 3,091 | 2,555 | 2,154 | 1,804 | 1,758 | 1,734 |
| 1000000 | 7,423 | 3,853 | 6,412 | 5,394 | 4,596 | 3,782 | 3,697 | 3,678 |

Rysunek 2: Tabela pomiarowa

## 4.4 Wykres

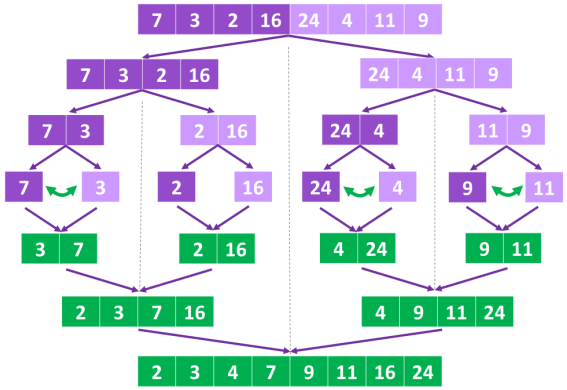



Rysunek 3: Wykres dla introsort

# 5 Sortowanie przez scalanie

## 5.1 Opis

Sortowanie przed scalanie to algorytm, który wykorzystuje metodę "dziel i zwyciężaj". Jednak tutaj złożoność pamięciowa jest większa oraz wymaga dodatkowej struktury danych aby mógł działać. Polega na dzieleniu tablicy na mniejsze części i powtarzaniu tego procesu aż do uzyskania tablic jednoelementowych. Tak pozyskane zbiory są scalane w coraz większe i posortowane, aby ponownie otrzymać całą tablicę, tym razem już posortowaną. Opisany algorytm jest stabilny, wydajny i prosty w implementacji. Przydaje się w przypadku danych dostępnych sekwencyjnie czyli np. tablicy jednokierunkowej albo pliku sekwencyjnego.




Rysunek 4: Sortowanie przez scalanie

## 5.2 Złożoność obliczeniowa

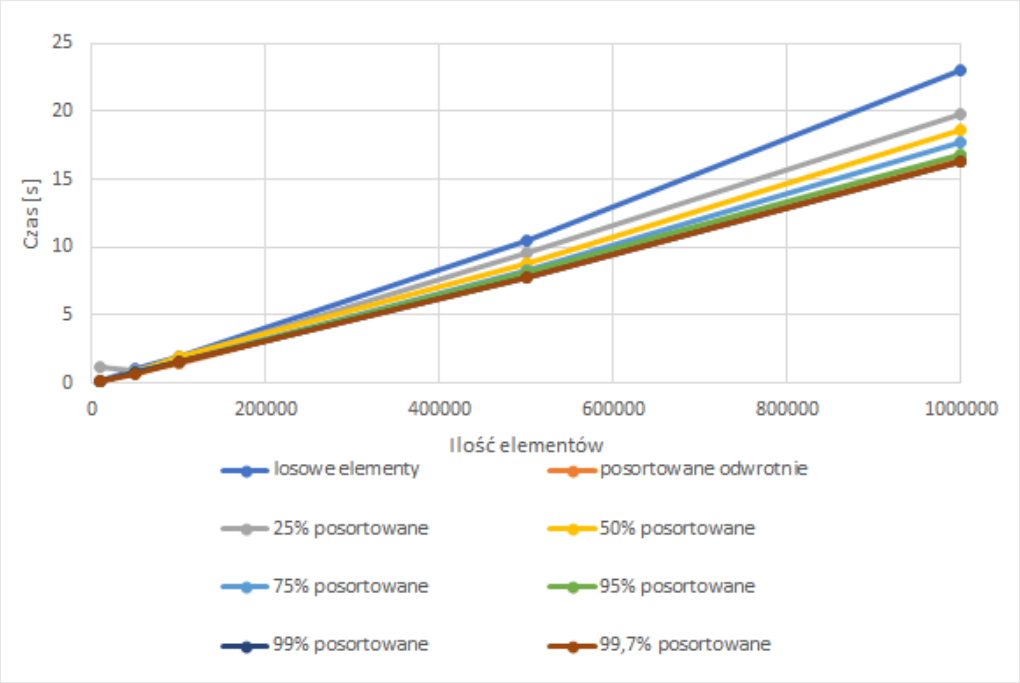
- Złożóność pamięciowa: $\Theta(n)$
- Złożoność czasowa: $\Theta(n \cdot logn)$
- Złożoności pesymistyczna, średnia, optymistyczna: $\Theta(n \cdot logn)$

## 5.3 Wyniki

| ilość elementów | losowe elmenty | posortowane odwrotnie | 25% posortowane | 50% posortowane | 75% posortowane | 95% posortowane | 99% posortowane | 99,7% posortowane |
|---|---|---|---|---|---|---|---|---|
| 10000 | 0,176 | 0,136 | 1,169 | 0,146 | 0,147 | 0,134 | 0,13 | 0,13 |
| 50000 | 0,972 | 0,741 | 0,891 | 0,815 | 0,762 | 0,735 | 0,717 | 0,71 |
| 100000 | 1,972 | 1,483 | 1,822 | 1,933 | 1,588 | 1,563 | 1,501 | 1,508 |
| 500000 | 10,487 | 8,091 | 9,617 | 8,749 | 8,23 | 8,102 | 7,777 | 7,724 |
| 1000000 | 23,069 | 16,606 | 19,757 | 18,6 | 17,665 | 16,783 | 16,303 | 16,3 |

Rysunek 5: Tabela pomiarowa

### 5.4 Wykres

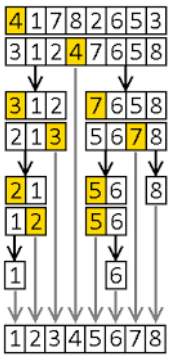



Rysunek 6: Wykres dla Mergesort

## 6 Sortowanie szybkie

### 6.1 Opis

Szybkie sortowanie jest to algorytm sortowania działający w średnim przypadku w czasie liniowo-logarytmicznym. Oparty jest na metodzie dziel i zwyciężaj, jest to technika projektowania algorytmów polegająca na podejściu rekurencyjnym, gdzie problem dzielimy na podproblemy, te podproblemy na jeszcze mniejsze podproblemy, aż dojdzie się do przypadków trywialnych. Nie jest to algorytm stabilny ani wykazujący zachowanie naturalne, jednak ze względu na efektywność jest algorytmem bardzo popularnym.



Rysunek 7: Sortowanie szybkie

## 6.2 Złożoność obliczeniowa

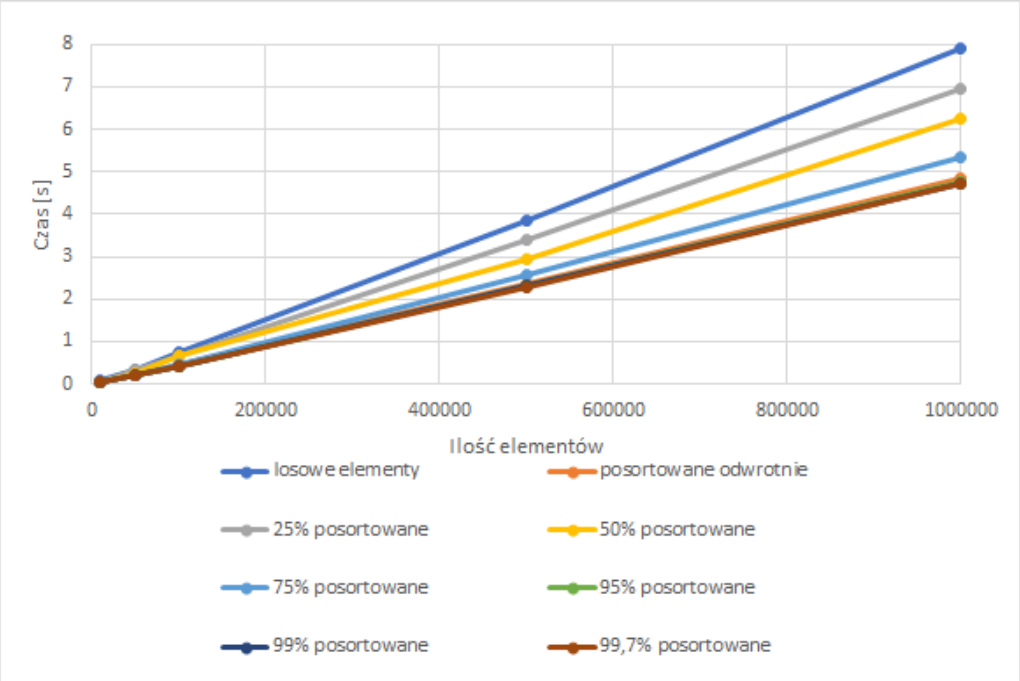
- Złożóność pamięciowa: $\Theta(n)$
- Złożoność czasowa: $\Theta(n \cdot logn)$
- Złożoności średnia, optymistyczna: $\Theta(n \cdot logn)$
- Złożoność pesymistyczna: $\Theta(n^2)$

## 6.3 Wyniki

| ilość elementów | losowe elmenty | posortowane odwrotnie | 25% posortowane | 50% posortowane | 75% posortowane | 95% posortowane | 99% posortowane | 99,7% posortowane |
|---|---|---|---|---|---|---|---|---|
| 10000 | 0,075 | 0,062 | 0,035 | 0,038 | 0,047 | 0,046 | 0,05 | 0,056 |
| 50000 | 0,348 | 0,211 | 0,314 | 0,246 | 0,224 | 0,202 | 0,207 | 0,203 |
| 100000 | 0,759 | 0,444 | 0,66 | 0,656 | 0,46 | 0,434 | 0,423 | 0,425 |
| 500000 | 3,855 | 2,341 | 3,403 | 2,957 | 2,565 | 2,301 | 2,302 | 2,268 |
| 1000000 | 7,904 | 4,84 | 6,945 | 6,262 | 5,33 | 4,759 | 4,737 | 4,729 |

Rysunek 8: Tabela pomiarowa

## 6.4 Wykres




Rysunek 9: Wykres dla Quicksort

# 7 Wnioski

- Im większy procent posortowania tablicy tym szybciej jest ona sortowana w całości, świadczy to o poprawnych działaniu programów.
- Sortowanie tablic zawierających wszytkie elementy losowe oraz nieposortowane w żaden sposób charakteryzuje się największym czasem.
- Sortowanie tablic które zostały już wcześniej posortowane w odwrotnej kolejności zajmuje stosunkowo mało czasu, jest on porównywalny do posortowania tablicy która jest już posortowana w mniej więcej 97% .
- Sortowanie szybkie jest najszybciej działającym algorytmem sortowania, ponieważ charakteryzuje się ono mało złożonością, jednak posiada również wadę którą jest słaba wartość przypadku pesymistycznego.

# 8 Bibliografia


- $http://algorytmy.ency.pl/artykul/quicksort$
- $http://math.uni.wroc.pl/\ jagiella/p2python/skrypt_html/wyklad5-2.html$
- $https://www.101computing.net/merge-sort-algorithm/$
- $http://www.algorytm.edu.pl/algorytmy-maturalne/quick-sort.html$
- $https://pl.wikipedia.org/wiki/Sortowanie$
- $http://www.algorytm.edu.pl/algorytmy-maturalne/sortowanie-przez-scalanie.html$
- $https://pl.wikipedia.org/wiki/Sortowanie_introspektywne$
- $https://www.semanticscholar.org/topic/Introsort/2636636$