P O L I T E C H N I K A R Z E S Z O W S K A

im. Ignacego Łukasiewicza

WYDZIAŁ MATEMATYKI I FIZYKI STOSOWANEJ

Karolina Wolska

Porównanie sortowania przez wstawianie z sortowaniem szybkim

Praca licencjacka / inżynierska /magisterska

kierunek studiów: Inżynieria i analiza danych

Rzeszów 2022

SPIS TREŚCI

1. Sortowanie przez wstawianie 3
1.1 Działanie algorytmu 3
1.2 Schemat blokowy 4
1.3 Pseudokod 5
1.4 Złożoność czasowa 5
1.4.1 Przykładowe czasy dla tablic różnej wielkości 6
1.4.2 Tabela średnich czasów sortowania tablicy 7
1.4.3 Wykresy na podstawie średnich czasów sortowania tablicy 7
1.5 Podsumowanie 8
2. Quicksort 9
2.1 Opis działania algorytmu 9
2.2 Schemat blokowy: 10
2.3 Pseudokod: 11
2.4 Złożoność czasowa 11
2.4.1 Przykładowe czasy sortowania tablic 12
2.4.2 Tabela średnich czasów sortowania tablicy 13
2.4.3 Wykresy na podstawie średnich czasów sortowania tablicy 13
2.5 Podsumowanie 14

1. Sortowanie przez wstawianie

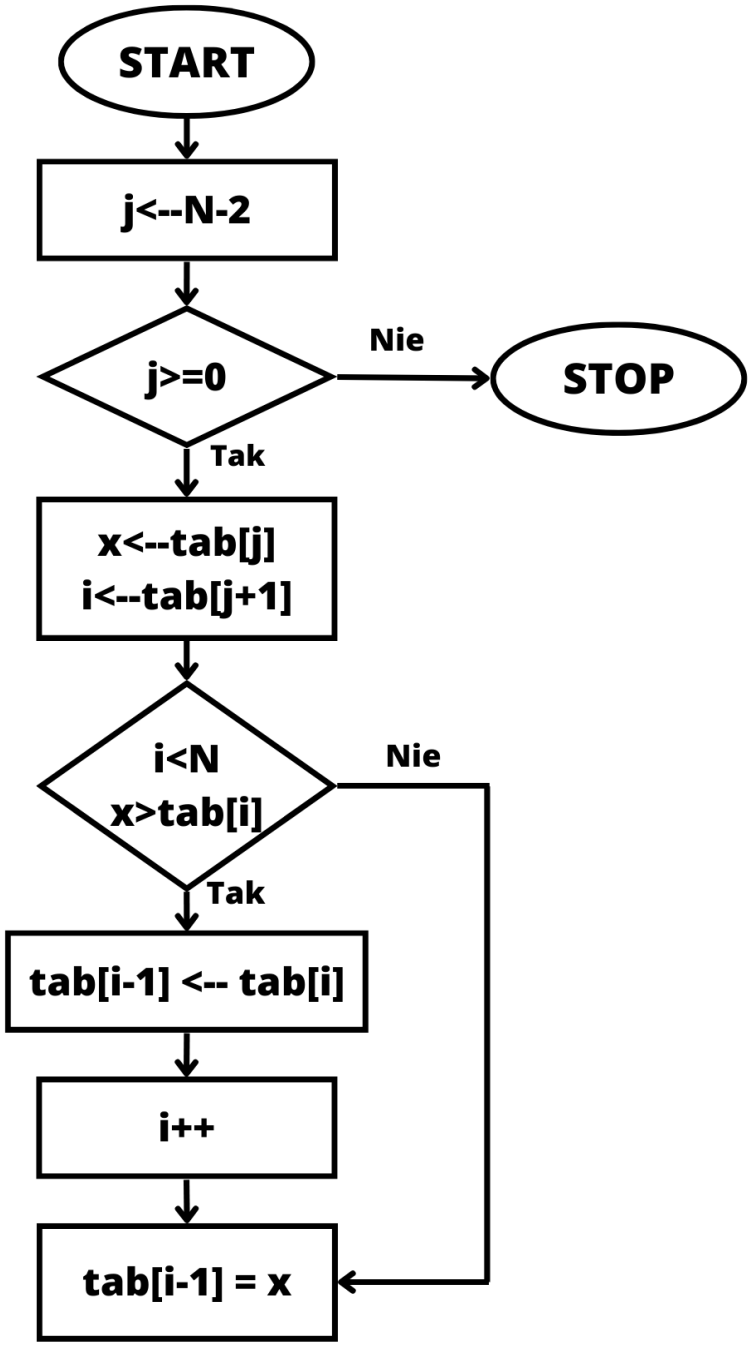
1.1 Działanie algorytmu

Algorytm sortowania przez wstawianie obejmuje posortowaną listę utworzoną na podstawie iteracyjnego porównania każdego elementu na liście z sąsiednim elementem.

Ostatnia wartość tworzy listę jednoelementową uporządkowaną. Na początku sortowania (przedostatni element) bieżąca wartość jest porównywana z wartością sąsiednią po prawej. Jeśli wartość jest większa niż bieżąca wartość, na liście nie są wprowadzane żadne modyfikacje, a lista rozrasta się o jedno miejsce; dzieje się tak również wtedy, gdy sąsiednia wartość i bieżąca wartość są tymi samymi liczbami.

Jeśli jednak sąsiednia wartość na prawo od bieżącej wartości jest mniejsza, pozycja sąsiedniej wartości zostanie przesunięta w lewo i przestanie się przesuwać w lewo tylko wtedy, gdy wartość na lewo od niej będzie mniejsza.

1.2 Schemat blokowy



1.3 Pseudokod:

```
Dla j <-- n-2, j<--n-3, ..., j=0 wykonuj
    x <-- tab[j]
    i <-- j+1
    dopóki i<=n i x>tab[i] wykonuj
        tab[i-1] <-- tab[i]
        i <-- i+1
    tab[i-1] <-- x
```

1.4 Złożoność czasowa

Najlepszym przypadkiem jest tablica, która jest już posortowana. W tym przypadku sortowanie przez wstawianie ma liniowy czas działania O(n). Podczas każdej iteracji pierwszy pozostały element danych wejściowych jest porównywany tylko z sąsiadującym po prawej stronie elementem posortowanej tablicy.

Najgorsza oraz średnia złożoność algorytmu sortowania przez wstawianie to O(n²). Oznacza to, że w najgorszym przypadku czas potrzebny na posortowanie listy jest proporcjonalny do kwadratu liczby elementów na liście, co sprawia, że sortowanie przez wstawianie jest niepraktyczne w przypadku sortowania dużych tablic.

Klasa złożoności obliczeniowej sortowania przez wstawianie	
optymistyczna	O(n)
typowa	O(n^2)
pesymistyczna	O(n^2)

1.4.1 Przykładowe czasy dla tablic różnej wielkości

```
podaj rozmiar tablicy: 16000
podaj przedzial liczb od 1 do: 1000

sortowanie przez wstawianie
czas sortowania tablicy losowej: zmierzony czas: 0.1640000045299530 sekund.
czas sortowania tablicy posortowanej: zmierzony czas: 0.0000000000000000 sekund.
czas sortowania tablicyo dwrotnej: zmierzony czas: 0.3680000007152557 sekund.
```

```
podaj rozmiar tablicy: 32000
podaj przedzial liczb od 1 do: 1000

sortowanie przez wstawianie
czas sortowania tablicy losowej: zmierzony czas: 0.6309999823570252 sekund.
czas sortowania tablicy posortowanej: zmierzony czas: 0.0000000000000000 sekund.
czas sortowania tablicy odwrotnej: zmierzony czas: 1.3059999942779541 sekund.
```

```
podaj rozmiar tablicy: 64000
podaj przedzial liczb od 1 do: 1000

sortowanie przez wstawianie
czas sortowania tablicy losowej: zmierzony czas: 2.5959999561309814 sekund.
czas sortowania tablicy posortowanej: zmierzony czas: 0.0000000000000000 sekund.
czas sortowania tablicy odwrotnej: zmierzony czas: 5.2340002059936523 sekund.
```

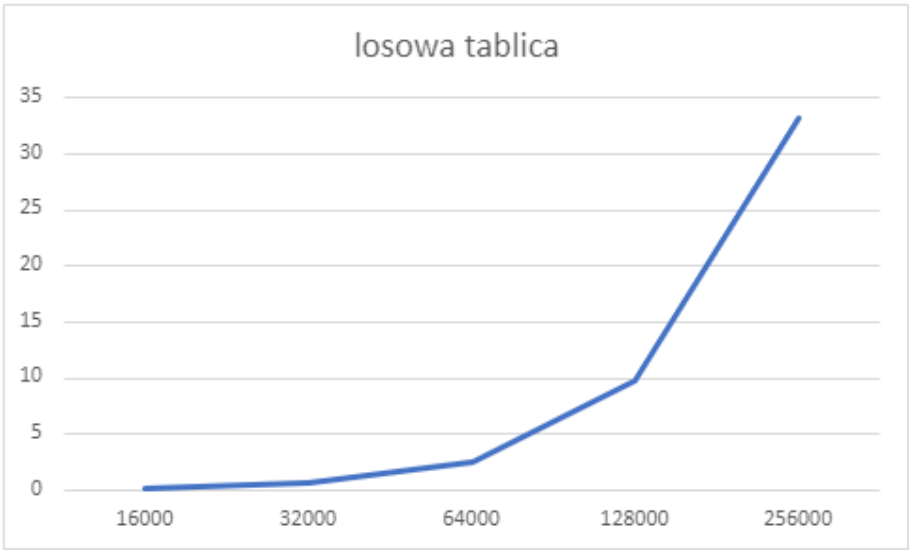
```
podaj rozmiar tablicy: 128000
podaj przedzial liczb od 1 do: 1000

sortowanie przez wstawianie
czas sortowania tablicy losowej: zmierzony czas: 9.7700004577636719 sekund.
czas sortowania tablicy posortowanej: zmierzony czas: 0.0030000000260770 sekund.
czas sortowania tablicy odwrotnej: zmierzony czas: 15.7930002212524410 sekund.
```

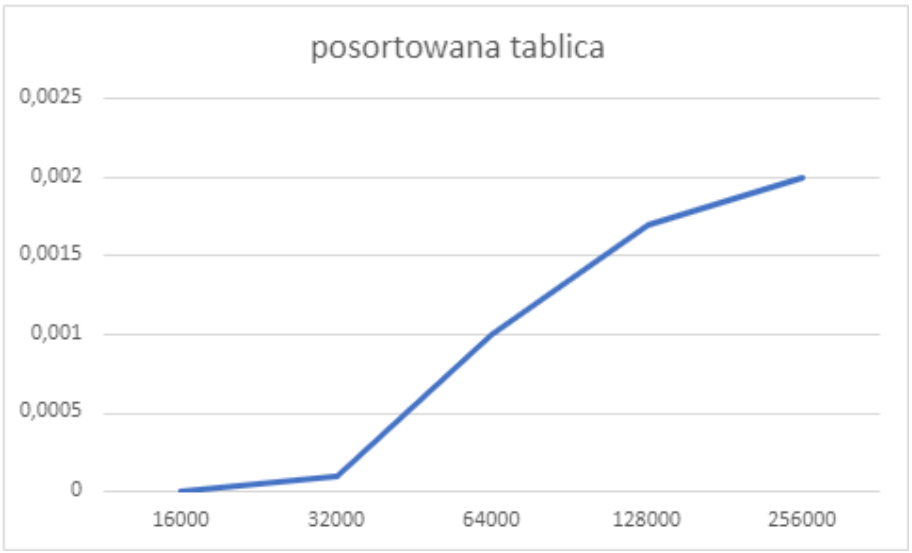
```
podaj rozmiar tablicy: 256000
podaj przedzial liczb od 1 do: 1000

sortowanie przez wstawianie
czas sortowania tablicy losowej: zmierzony czas: 32.6059989929199220 sekund.
czas sortowania tablicy posortowanej: zmierzony czas: 0.0010000000474975 sekund.
czas sortowania tablicy odwrotnej: zmierzony czas: 62.8349990844726560 sekund.
```

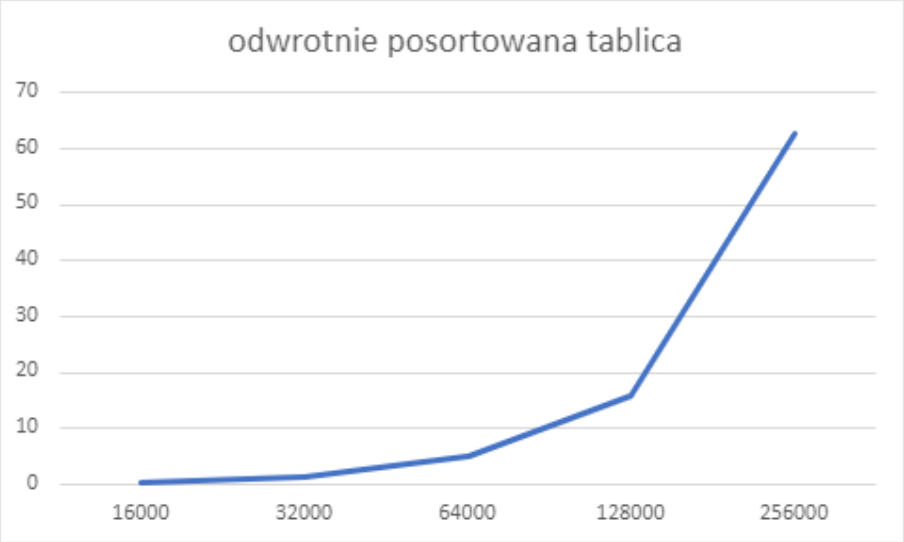
1.4.2 Tabela średnich czasów sortowania tablicy

	Czas sortowania tablicy		
Liczba elementów N	losowa	posortowana	Odwrotnie posortowana
16000	0,16 s	0 s	0,35 s
32000	0,62 s	0,0001 s	1,31 s
64000	2,54 s	0,001 s	5,205 s
128000	9,8 s	0,0017 s	15,8 s
256000	33,3 s	0,002 s	62,6 s

1.4.3 Wykresy na podstawie średnich czasów sortowania tablicy







1.5 Podsumowanie

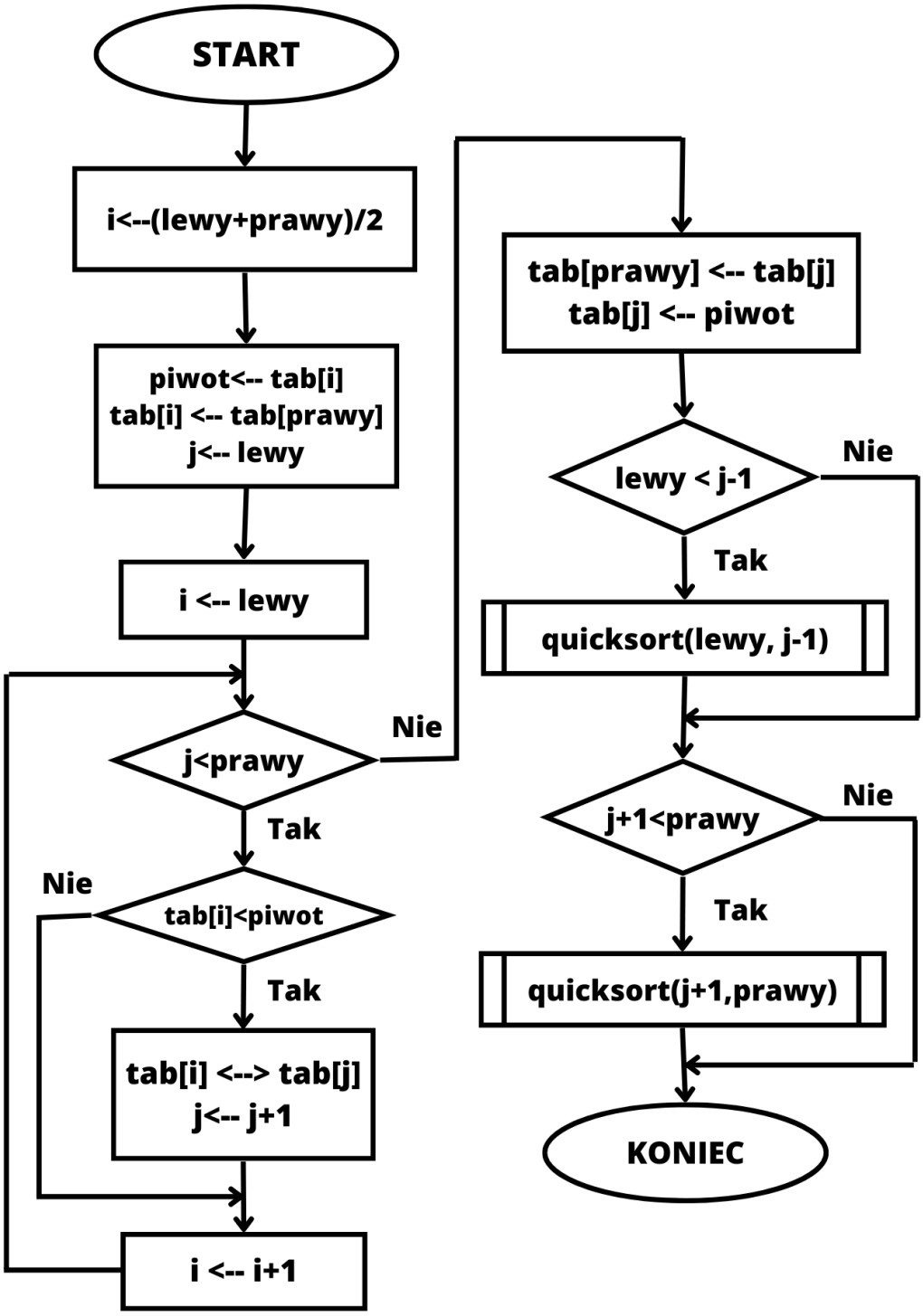
Główną zaletą sortowania przez wstawianie jest jego prostota i łatwość w zaimplementowaniu. Jest również algorytmem sortowania na miejscu, więc wymagania dotyczące miejsca są minimalne. Sortowanie przez wstawianie jest jednym z najszybszych algorytmów sortowania bardzo małych tablic oraz zbiorów wstępnie posortowanych. Jednak jest nieefektywne dla dużych tablic, ze względu na złożoność czasową O(n^2).

2. Quicksort

2.1 Opis działania algorytmu

Quicksort to algorytm działający na zasadzie „dziel i zwyciężaj”. Działa poprzez wybranie elementu zwanego „piwotem” z tablicy i podzielenie pozostałych elementów na dwie podtablice, w zależności od tego, czy są mniejsze, czy większe od osi obrotu. Podtablice są następnie sortowane rekurencyjnie. Po każdym wywołaniu wybrany element przestawny zajmuje właściwą pozycję w tablicy, tak jak ma to miejsce w tablicy posortowanej. Tak więc z każdym krokiem nasz problem zmniejsza się o 2, co prowadzi do szybkiego sortowania.

2.2 Schemat blokowy:



2.3 Pseudokod:

```
quicksort(lewy, prawy)
 i <-- (lewy+prawy)/2
piwot  <-- tab[ i ]
tab[ i ] <-- tab[prawy]
j <-- lewy
i <-- ostatni
dopóki i<prawy wykonuj
    Jeśli tab[ i ] < piwot to
        tab[ i ] <=> tab [ j ]
        j <- j+1
    i = i+1
tab[prawy] <-- tab[ j ]
tab[ j ] <-- piwot
jeśli lewy<j-1 to
    quicksort(lewy,j-1)
jeśli j+1<prawy, to
    quicksort(j+1,prawy)
```

2.4 Złożoność czasowa

W optymistycznym i przeciętnym przypadku za każdym razem, gdy dokonujemy podziału, dzielimy listę na dwie prawie równe części. Oznacza to, że każde wywołanie rekurencyjne przetwarza listę o połowę mniejszą. W związku z tym możemy wykonać tylko $\log_2 n$ zagnieżdżonych wywołań, zanim dotrzemy do listy o rozmiarze 1, a więc złożoność czasowa wynosi O(nlogn) W najgorszym przypadku, gdy sortujemy ciąg, który jest uporządkowany odwrotnie, realizowanych jest n^2 porównań. Wynika stąd, że wtedy złożoność czasowa algorytmu wynosi O(n^2).

Klasa złożoności obliczeniowej Quicksort	
optymistyczna	O(nlogn)
typowa	O(nlogn)
pesymistyczna	O()

2.4.1 Przykładowe czasy sortowania tablic

```
podaj rozmiar tablicy: 16000
podaj przedzial liczb od 1 do: 1000

Quicksort
czas sortowania tablicy losowej: 0.002000 sekund.
czas sortowania tablicy posortowanej: 0.001000 sekund.
czas sortowania tablicy odwrotnej: 0.001000 sekund.
```

```
podaj rozmiar tablicy: 32000
podaj przedzial liczb od 1 do: 1000

Quicksort
czas sortowania tablicy losowej: 0.003000 sekund.
czas sortowania tablicy posortowanej: 0.002000 sekund.
czas sortowania tablicy odwrotnej: 0.002000 sekund.
```

```
podaj rozmiar tablicy: 64000
podaj przedzial liczb od 1 do: 1000

Quicksort
czas sortowania tablicy losowej: 0.007000 sekund.
czas sortowania tablicy posortowanej: 0.004000 sekund.
czas sortowania tablicy odwrotnej: 0.003000 sekund.
```

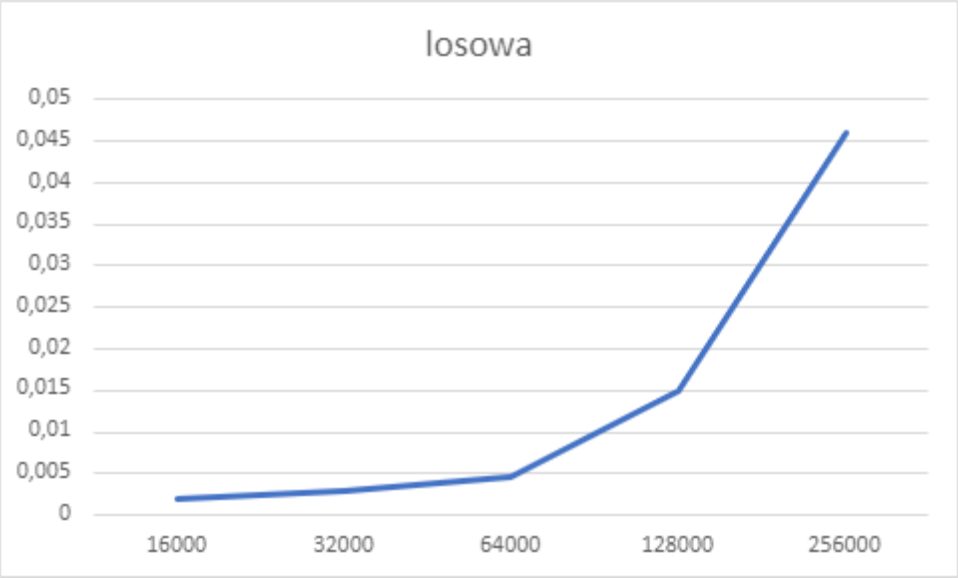
```
podaj rozmiar tablicy: 128000
podaj przedzial liczb od 1 do: 1000

Quicksort
czas sortowania tablicy losowej: 0.017000 sekund.
czas sortowania tablicy posortowanej: 0.011000 sekund.
czas sortowania tablicy odwrotnej: 0.010000 sekund.
```

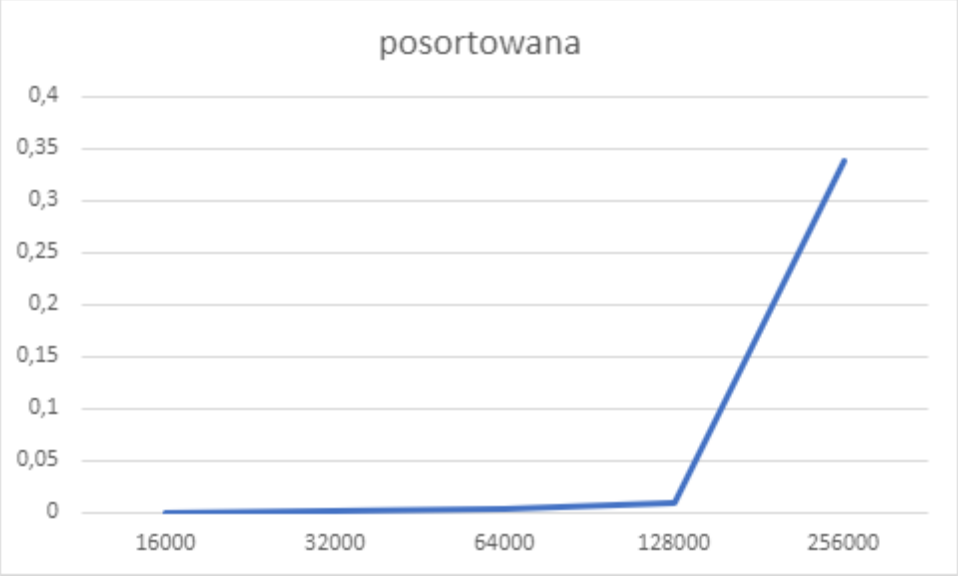
```
podaj rozmiar tablicy: 256000
podaj przedzial liczb od 1 do: 1000

Quicksort
czas sortowania tablicy losowej: 0.046000 sekund.
czas sortowania tablicy posortowanej: 0.034000 sekund.
czas sortowania tablicy odwrotnej: 0.033000 sekund.
```

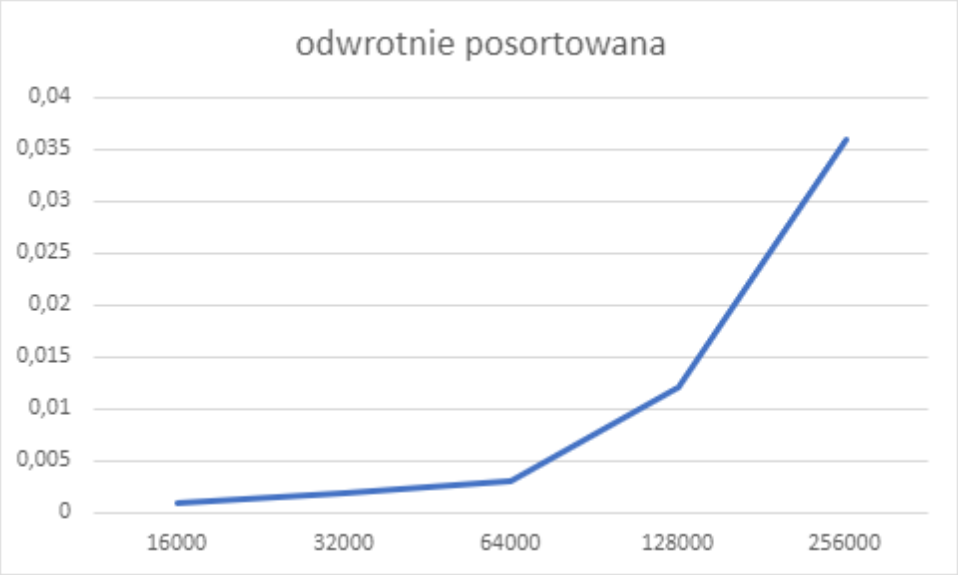
2.4.2 Tabela średnich czasów sortowania tablicy

	Czas sortowania tablicy		
Liczba elementów N	losowa	posortowana	Odwrotnie posortowana
16000	0,002 s	0,001 s	0,001 s
32000	0,003 s	0,002 s	0,002 s
64000	0,0045 s	0,004 s	0,0031 s
128000	0,015 s	0,01 s	0,0122 s
256000	0,046 s	0,34 s	0,036 s

2.4.3 Wykresy na podstawie średnich czasów sortowania tablicy







2.5 Podsumowanie

Algorytm szybkiego sortowania jest bardzo wydajnym algorytmem, bardzo dobrze radzącym sobie z dużymi ilościami danych, ze względu na złożoność czasową O(nlogn). Ponieważ sortuje w miejscu, nie jest wymagane żadne dodatkowe miejsce do przechowywania. Wadą jest, że w pesymistycznym przypadku jego złożoność czasowa wynosi O(n^2).