

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

STACH KACPER

Projekt #2 – porównanie algorytmów sortujących

kierunek studiów: Inżynieria i analiza danych

Rzeszów 2022

SPIS TREŚCI

[1. Sortowanie grzebieniowe 4](#_Toc121176117)

[1. Podstawy teoretyczne sortowania grzebieniowego 4](#_Toc121176118)

[2. Kod funkcji combSort() 5](#_Toc121176119)

[3. Schemat blokowy sortowania grzebieniowego 6](#_Toc121176120)

[4. Pseudokod funkcji sortowania grzebieniowego 7](#_Toc121176121)

[5. Wykres złożoności obliczeniowej sortowania grzebieniowego. 8](#_Toc121176122)

[1. Średnia złożoność obliczeniowa. 9](#_Toc121176123)

[2. Złożoność obliczeniowa w przypadku pesymistycznym. 9](#_Toc121176124)

[3. Złożoność obliczeniowa w przypadku optymistycznym: 9](#_Toc121176125)

[2. Sortowanie przez wstawianie 10](#_Toc121176126)

[1. Podstawy teoretyczne sortowania przez wstawianie 10](#_Toc121176127)

[2. Kod funkcji selection\_sort() 11](#_Toc121176128)

[3. Schemat blokowy sortowania przez wstawianie 12](#_Toc121176129)

[4. Pseudokod algorytmu sortowania przez wstawianie 13](#_Toc121176130)

[5. Wykres złożoności czasowej sortowania przez wybieranie 14](#_Toc121176131)

[1. Średnia złożoność obliczeniowa. 15](#_Toc121176132)

[2. Złożoność obliczeniowa w przypadku pesymistycznym. 15](#_Toc121176133)

[3. Złożoność obliczeniowa w przypadku optymistycznym: 15](#_Toc121176134)

[4. Wnioski 16](#_Toc121176135)

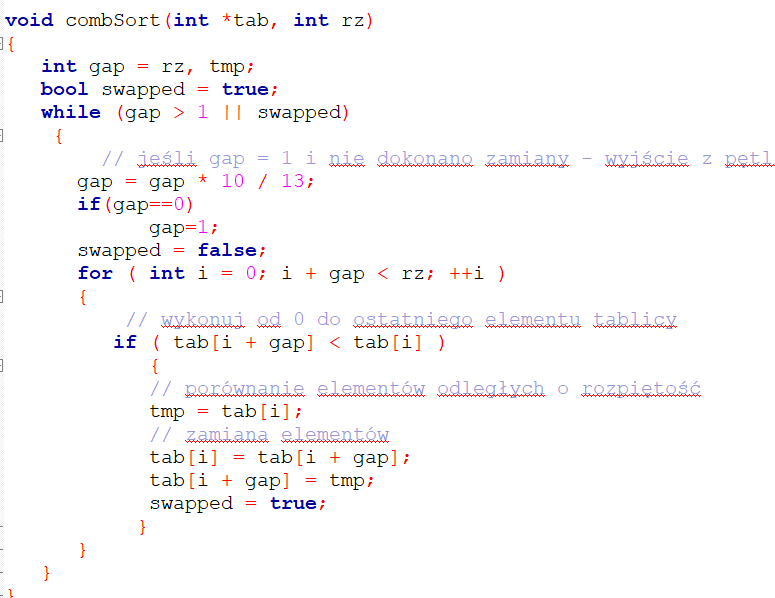
# Sortowanie grzebieniowe

## Podstawy teoretyczne sortowania grzebieniowego

Sortowanie Grzebieniowe opiera się na idei sortowania bąbelkowego. Różnica jednak polega na wykonywaniu zamian w inny sposób - staramy się w każdej iteracji przesunąć wszystkie duże elementy na sam koniec. Można to porównać do czesania z grzebieniem. Początkowo rozczesujemy włosy grzebieniem z dużym odstępem pomiędzy ząbkami - w ten sposób będzie łatwiej i szybciej rozczesać pojedyncze grupki.

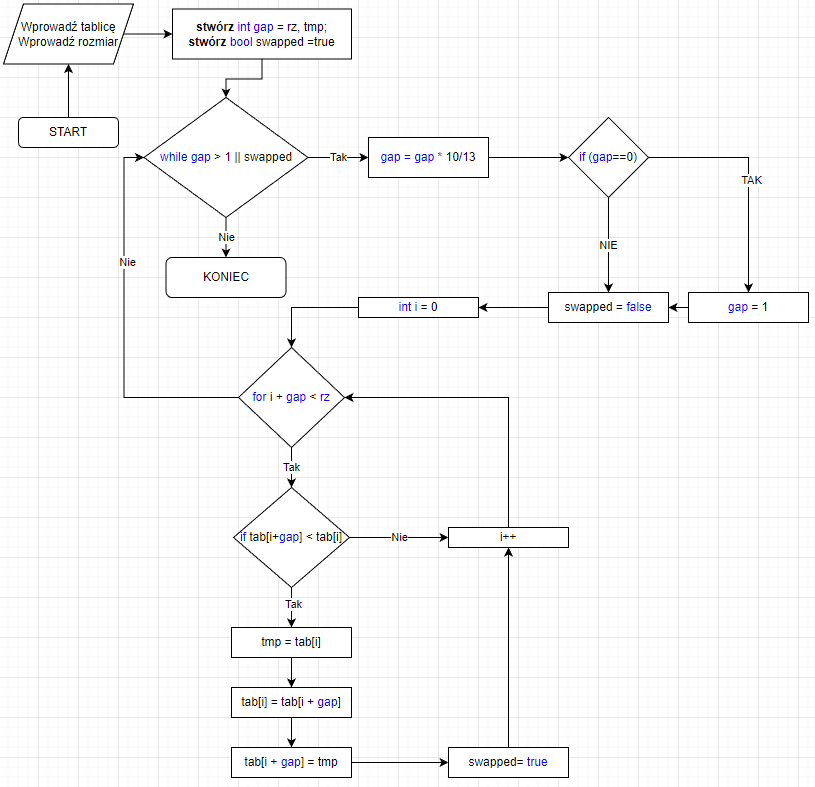
Algorytm sortowania Grzebieniowego zawiera element empiryczny - czyli wyliczony doświadczalnie. Jest to współczynnik *w:=***1.3**. Nie ma on znaczenia dla poprawności posortowania listy, ale ma znaczący wpływ na szybkość wykonywania algorytmu. W ten sposób zwykłe sortowanie bąbelkowe poprawiamy prawdopodobnie do złożoności . Niemniej jest on prawdopodobnie wolniejszy od sortowania szybkiego.

## Kod funkcji combSort()



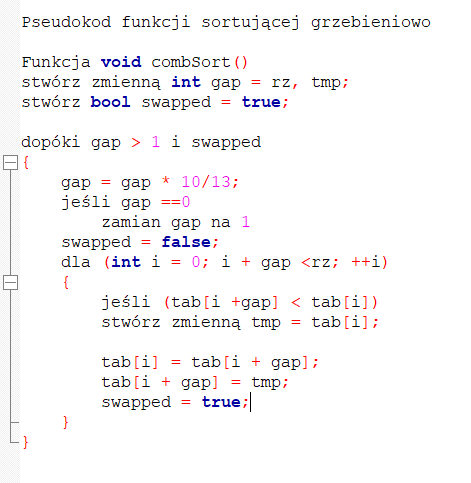
Rysunek 1 - kod funkcji combSort

## Schemat blokowy sortowania grzebieniowego



Rysunek 2 - schemat blokowy sortowania grzebieniowego

## Pseudokod funkcji sortowania grzebieniowego



Rysunek 3 - pseudokod sortowania grzebieniowego

## Wykres złożoności obliczeniowej sortowania grzebieniowego.

Wykres przedstawia zależność czasu od ilości rekordów w tabeli sortowanej. Czas przedstawiony jest w sekundach.

Rysunek 4 - wykres sortowania grzebieniowego

### Średnia złożoność obliczeniowa.

Średnią złożoność obliczeniowa algorytmu sortowania grzebieniowego można wyrazić wzorem:

### Złożoność obliczeniowa w przypadku pesymistycznym.

Złożoność obliczeniową w przypadku pesymistycznym możemy wyrazić poniższym wzorem:

### Złożoność obliczeniowa w przypadku optymistycznym:

Złożoność obliczeniową w przypadku optymistycznym możemy wyrazić wzorem:

)

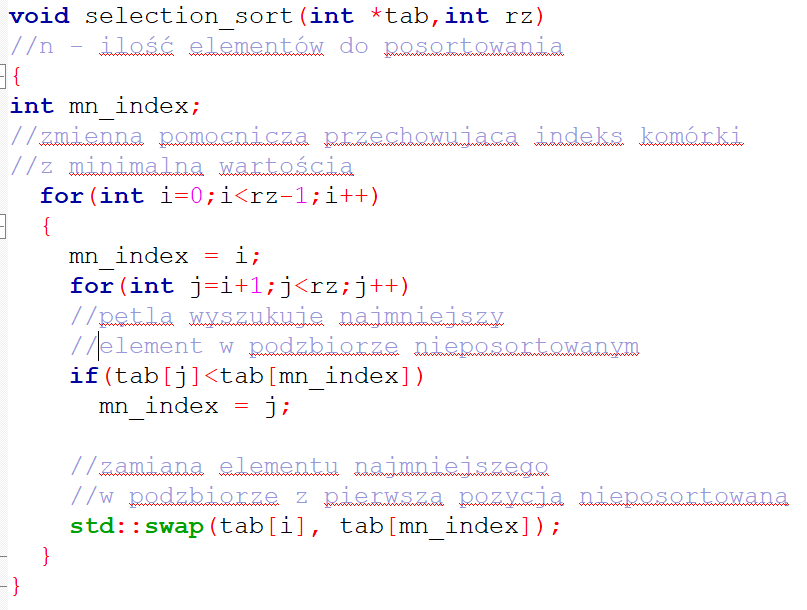
# Sortowanie przez wstawianie

## Podstawy teoretyczne sortowania przez wstawianie

Sortowanie przez wstawianie jest jednym z najprostszych algorytmów sortowania. Został on opracowany w sposób, który może odzwierciedlać np. ustawianie kart – kolejne elementy są ustawiane na odpowiednie miejsca docelowe.

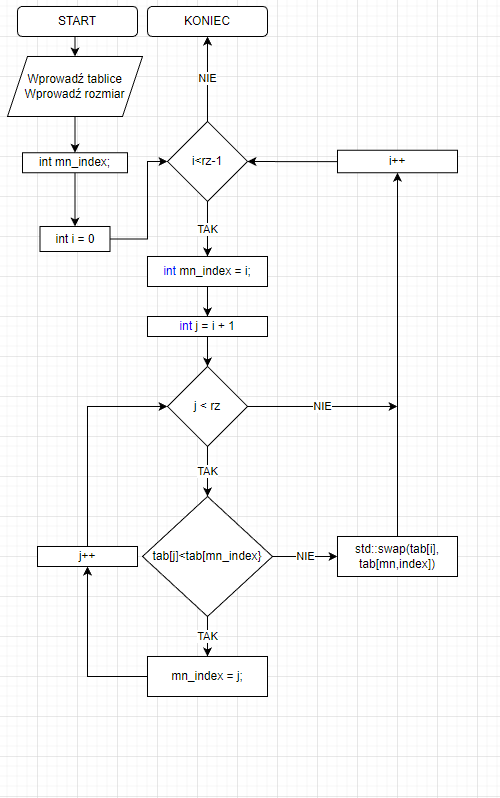
Cechuje go wysoka wydajność dla zbiorów o stosunkowo niewielkiej wielkości. Mimo to, jest znacznie mniej wydajny od innych algorytmów sortujących, chociażby wyżej przedstawionego sortowania grzebieniowego.

## Kod funkcji selection\_sort()



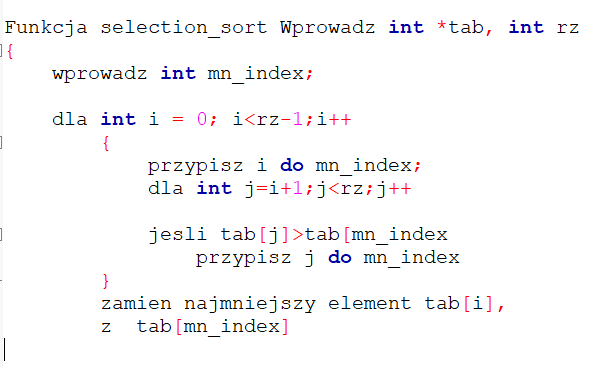
Rysunek 5 - kod funkcji selection\_sort

## Schemat blokowy sortowania przez wstawianie



Rysunek 6 - schemat blokowy sortowania przez wstawianie

## Pseudokod algorytmu sortowania przez wstawianie



Rysunek 7 - pseudokod sortowania przez wstawianie

## Wykres złożoności czasowej sortowania przez wybieranie

Wykres przedstawia zależność czasu od ilości rekordów w tabeli sortowanej. Czas przedstawiony jest w sekundach.

Rysunek 8 - wykres sortowania przez wstawianie

### Średnia złożoność obliczeniowa.

Średnią złożoność obliczeniowa algorytmu sortowania przez wstawianie można wyrazić wzorem:

### Złożoność obliczeniowa w przypadku pesymistycznym.

Złożoność obliczeniową w przypadku pesymistycznym możemy wyrazić poniższym wzorem:

### Złożoność obliczeniowa w przypadku optymistycznym:

Złożoność obliczeniową w przypadku optymistycznym możemy wyrazić wzorem:

# Wnioski

Analizując obie metody sortowania, można dojść do następujących wniosków:

* Metoda sortowania grzebieniowego jest znacznie wydajniejsza od metody sortowania poprzez wstawianie,
* Metoda sortowania przez wstawianie znacznie traci na efektywności, gdy operujemy na dużych ilościach danych.
* Udało się poprawnie sprawdzić złożoność czasową algorytmów na podstawie działania programu
* Udało się poprawnie zaimplementować odczyt z pliku oraz zapis do pliku tekstowego.

SPIS RYSUNKÓW

[Rysunek 1 - kod funkcji combSort 5](#_Toc121176213)

[Rysunek 2 - schemat blokowy sortowania grzebieniowego 6](#_Toc121176214)

[Rysunek 3 - pseudokod sortowania grzebieniowego 7](#_Toc121176215)

[Rysunek 4 - wykres sortowania grzebieniowego 8](#_Toc121176216)

[Rysunek 5 - kod funkcji selection\_sort 11](#_Toc121176217)

[Rysunek 6 - schemat blokowy sortowania przez wstawianie 12](#_Toc121176218)

[Rysunek 7 - pseudokod sortowania przez wstawianie 13](#_Toc121176219)

[Rysunek 8 - wykres sortowania przez wstawianie 14](#_Toc121176220)