

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

Damian Stechnij

**Nr albumu: 173219**

##### PROJEKT

ALGORYTMY I STRUKTURY DANYCH

kierunek studiów: inżynieria i analiza danych

Rzeszów, 2022

**Spis treści**

[1 TEMAT PRACY 3](#_Toc121423239)

[2 PROJEKTOWANIE 4](#_Toc121423240)

[2.1 Problem zagadnienia 4](#_Toc121423241)

[2.2 Teoretyczne podstawy 4](#_Toc121423242)

[2.2.1 Algorytm sortowania grzebieniowego 4](#_Toc121423243)

[2.2.2 Algorytm sortowania kopcowego 4](#_Toc121423244)

[3 SCHEMATY BLOKOWE I PSEUDOKODY 5](#_Toc121423245)

[3.1 Pseudokod 5](#_Toc121423246)

[3.1.1 Algorytm sortowania grzebieniowego 5](#_Toc121423247)

[3.1.2 Algorytm sortowania kopcowego 5](#_Toc121423248)

[3.2 Schemat blokowy 6](#_Toc121423249)

[3.2.1 Algorytm sortowania grzebieniowego 6](#_Toc121423250)

[3.2.2 Algorytm sortowania kopcowego 7](#_Toc121423251)

[4 KOD ŹRÓDŁOWY ALGORYTMÓW SORTUJĄCYCH 9](#_Toc121423252)

[4.1 Sortowanie grzebieniowe 9](#_Toc121423253)

[4.1.1 Funkcja wykonująca znajdująca rozpiętość 9](#_Toc121423254)

[4.1.2 Funkcja wykonująca sortowanie 9](#_Toc121423255)

[4.2 Sortowanie kopcowe 10](#_Toc121423256)

[4.2.1 Funkcja znajdująca większy element 10](#_Toc121423257)

[4.2.2 Funkcja wykonująca sortowanie kopcowe 10](#_Toc121423258)

[5 DZIAŁANIE PROGRAMU 11](#_Toc121423259)

[5.1 Przykład 11](#_Toc121423260)

[5.1.1 Dane wejściowe w pliku tekstowym 11](#_Toc121423261)

[5.1.2 Wyniki wypisane w konsoli 11](#_Toc121423262)

[5.1.3 Wyniki wypisane w plikach tekstowych 11](#_Toc121423263)

[6 ZŁOŻONOŚĆ OBLICZENIOWA 12](#_Toc121423264)

[6.1 Sortowanie kopcowe 12](#_Toc121423265)

[6.2 Sortowanie grzebieniowe 13](#_Toc121423266)

[7 WNIOSKI 14](#_Toc121423267)

# TEMAT PRACY

Porównanie algorytmu sortowania kopcowego i algorytmu sortowania grzebieniowego.

Cechy jakie powinien zawierać program:

* program powinien odczytywać dane wejściowe z pliku tekstowego i zapisywać posortowany już ciąg do pliku tekstowego,
* w celu wykonania testów należy zaimplementować funkcję, która generuje pseudolosowy ciąg elementów o zadanej długości,
* założyć, że sortowanymi elementami są liczby całkowite z przedziału [0, N], gdzie N powinno być „odpowiednio dużym” parametrem ustalonym wewnątrz programu,
* kod powinien zawierać komentarze ułatwiające zrozumienie programu.

# PROJEKTOWANIE

## Problem zagadnienia

W zadaniu trzeba porównać dwie metody sortowania: sortowanie przez kopcowanie oraz sortowanie grzebieniowe.

Aby wykonać testy trzeba utworzyć wygenerowaną tablicę z pseudolosowymi liczbami. Program docelowo ma wczytywać plik tekstowy i wpisywać dane do tablicy.

Później sortujemy liczby obiema metodami, porównujemy czasy sortowań oraz zapisujemy wszystko do pliku.

## Teoretyczne podstawy

### Algorytm sortowania grzebieniowego

Przyjmuje długość tablicy za rozpiętość, którą dzielimy przez 1.3 i odrzucamy część ułamkową. Bada kolejno wszystkie pary obiektów odległych o wyliczoną wcześniej rozpiętość, gdy są ułożone niemonotonicznie zamieniamy je miejscami. Wykonuje się to w pętli, do momentu, gdy rozpiętość osiągnie wartość 1. Gdy rozpiętość osiągnęła wartość 1 zachowuje się tak jak sortowanie bąbelkowe.

### Algorytm sortowania kopcowego

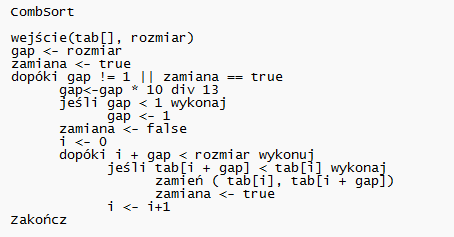
Podstawą algorytmu jest wykorzystanie struktury danych typu kopiec binarny. Złożony jest z dwóch faz, w których w pierwszym elementy reorganizowane są w celu utworzenia kopca, a w drugiej wykonywane jest sortowanie właściwe.

Do utworzenia kopca można wykorzystywać początkową tablicę, w której znajdują się nieposortowane elementy. Na początku do kopca należy tylko pierwszy element tablicy, potem rozszerzamy go o kolejne elementy, sprawdzając czy przy każdym nowym wprowadzonym elemencie jest spełniony warunek kopca. Jeśli takowy nie jest to przemieszczamy elementy w górę kopca, by warunek został spełniony.

# SCHEMATY BLOKOWE I PSEUDOKODY

## Pseudokod

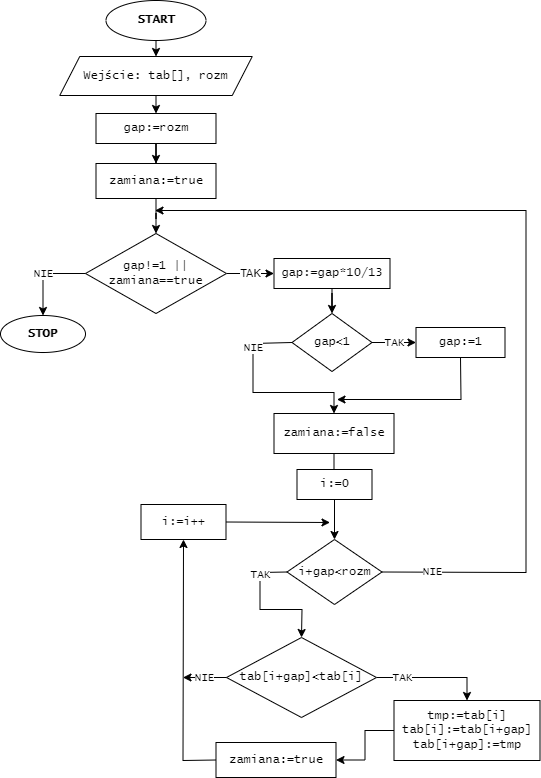
### Algorytm sortowania grzebieniowego



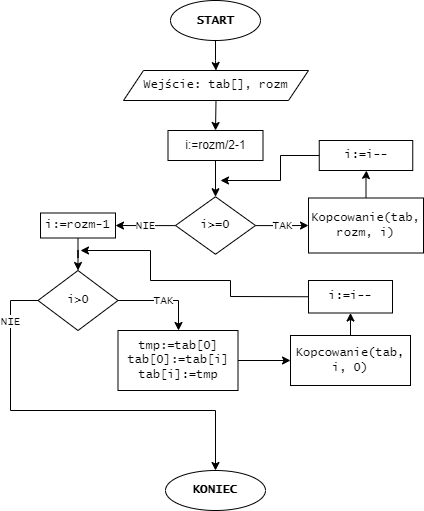
### Algorytm sortowania kopcowego

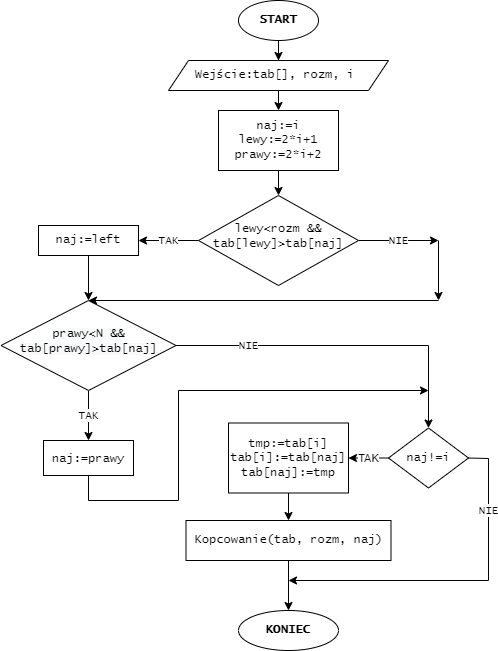
## Schemat blokowy

### Algorytm sortowania grzebieniowego



### Algorytm sortowania kopcowego

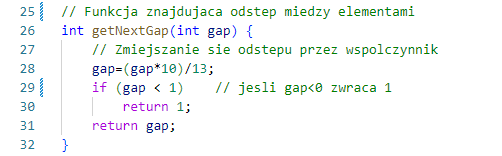




# KOD ŹRÓDŁOWY ALGORYTMÓW SORTUJĄCYCH

## Sortowanie grzebieniowe

### Funkcja wykonująca znajdująca rozpiętość



### Funkcja wykonująca sortowanie

## Sortowanie kopcowe

### Funkcja znajdująca większy element

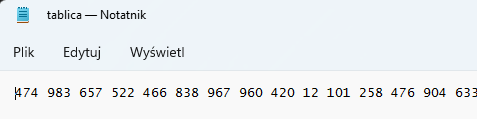
### Funkcja wykonująca sortowanie kopcowe

# DZIAŁANIE PROGRAMU

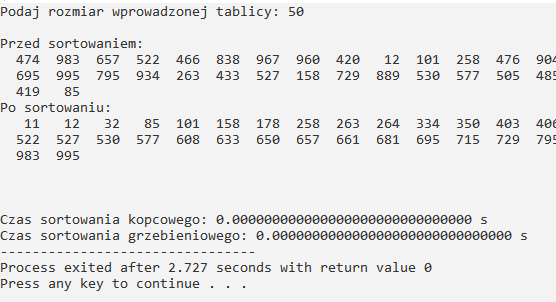
Program wczytuje plik, w którym znajduje się tablica (tablica.txt), oraz zapisuje posortowaną już tablicę do pliku wyniki1.txt dla kopcowego i wyniki2.txt dla grzebieniowego. W konsoli podajemy rozmiar tablicy jaki chcemy wczytać, a po wykonaniu wyświetla się czas sortowania tablic dla każdej z metod.

## Przykład

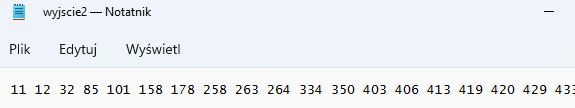
### Dane wejściowe w pliku tekstowym



### Wyniki ****wypisane**** w konsoli



### Wyniki wypisane w plikach tekstowych



# ZŁOŻONOŚĆ OBLICZENIOWA

## Sortowanie kopcowe

Algorytm sortowania przez kopcowanie jest niestabilnym, ale mimo to szybkim i niepochłaniającym wiele pamięci. Jego złożoność czasowa wynosi O(n log n), a pamięciowa O(n), złożoność pamięciowa dodatkowych struktur wynosi O(1), czyli wykonuje się ”w miejscu”. Jest odporny na tak spreparowane dane powodujące wolniejsze działanie, co widać na wykresie. Jego działanie na różnych próbkach danych jest mniej więcej podobne czasowo.

## Sortowanie grzebieniowe

Algorytm sortowania grzebieniowego oparty jest na sortowaniu bąbelkowym. Jego złożoność czasowa prawdopodobnie wynosi O(n log n). Współczynnik 1.3 wyznaczono doświadczalnie.

Działa o wiele szybciej dla tablicy, w której to elementy są ułożone malejąco niż dla pozostałych próbek danych.

# WNIOSKI

Algorytm grzebieniowy szybszy niż algorytm sortowania przez kopcowanie.

Dla tablicy, w której elementy są ułożone malejąco sortowanie grzebieniowe jest o wiele szybsze niż dla tablicy, w której elementy są pseudolosowe lub do połowy posortowane.

W sortowaniu przez kopcowanie nie ma większych różnic czasowych w różnie zadanych próbkach danych.

Program może pobierać dane wejściowe z pliku jak i wypisywać dane wyjściowe do pliku.

Czas jest mierzony dla obu sposobów sortowania.

Kod zawiera komentarze, które pomogą zrozumieć działanie programu.

Schemat blokowy i pseudokod zostały sporządzone do obu algorytmów.

Wersje programu po zmianach były umieszczane na serwer GitHub, do którego link został wysłany mailowo.