**Akademia Finansów i Biznesu Vistula**

**Kierunek studiów - Informatyka**

**Zofia Wasilonek**

54293

# ALGORYTM SZYBKIEGO SORTOWANIA

**(QUICK SORT)**

Warszawa, 16.04.2022

Algorytm QuickSort jest procedurą rekurencyjną i opiera się na podejściu dziel i zwyciężaj, w którym tablica jest dzielona na podtablice poprzez wybranie elementu przestawnego. Podczas dzielenia tablicy element pivot (inczej zwany też cementem – element, od którego rozpoczyna się porównanie liczb) należy ustawić w taki sposób, aby elementy mniejsze niż pivot były po lewej stronie, a elementy większe niż pivot po prawej.

Ten sam proces jest kontynuowany zarówno dla lewej, jak i prawej podtablicy. Na koniec posortowane elementy są łączone w posortowaną tablicę.

W moim algorytmie element pivot to zawsze ostatni element w tablicy, a więc pierwszy element wprowadzony do konsoli przez użytkownika.

Złożoność szybkiego sortowania

**Złożoność czasu**

To, co najlepsze O(n\*log n)

Najgorszy O(n 2 )

Przeciętny O(n\*log n)

**Złożoność przestrzeni** O(log n)

**Stabilność** Nie

## 1. Złożoność czasowa

**Złożoność najgorszego przypadku [Big-O]** : O(n2) występuje, gdy wybrany element obrotu jest albo największym, albo najmniejszym elementem. Ten stan prowadzi do przypadku, w którym element obrotowy znajduje się na skrajnym końcu posortowanej tablicy. Jedna podtablica jest zawsze pusta, a inna podtablica zawiera elementy. W związku z tym funkcja quicksort jest wywoływana tylko w tej podtablicy. Jednak algorytm szybkiego sortowania ma lepszą wydajność w przypadku rozproszonych przegubów.

**Średnia złożoność przypadku [Big-theta]** : O(n\*log n) występuje, gdy powyższe warunki nie występują, czyli np. gdy elementy tablicy są w pomieszanej kolejności, która nie jest prawidłowo rosnąca i nieprawidłowa malejąca.

**Złożoność najlepszego przypadku [Big-omega]** : O(n\*log n) występuje, gdy element obrotowy jest zawsze elementem środkowym lub blisko elementu środkowego.

## 2. Złożoność przestrzeni

Złożoność przestrzeni dla quicksort to O(log n).

Tabela. Wynik testów dla sortowania szybkiego

|  |  |
| --- | --- |
| **Ilość danych** | **Czas [milisekundy]** |
| 0 | 0 |
| 100 | 0,0608 |
| 400 | 0,3366 |
| 800 | 0,5145 |
| 1200 | 0,7407 |
| 2000 | 3,5735 |
| 5000 | 3,7779 |
| 10000 | 3,8823 |
| 50000 | 12,5035 |

1

2

3

4

5

6

7

8

9

Ilość danych

0

100

400

800

1200

2000

5000

10000

50000

Czas [milisekundy]

0

0,0608

0,3366

0,5145

0,7407

3,5735

3,7779

3,8823

12,5035

0

10000

20000

30000

40000

50000

60000

Wykres zależność ilości danych od czasu dla

sortowania szybkiego

Ilość danych

Czas [milisekundy]

### Program ZWasilonek\_54293\_QuickSort\_App

Jest to program napisany w języku Java, który za pivot przyjmuje zawsze ostatni element w tablicy pobranych od użytkownika liczb.

**import** java**.**util**.**Arrays**;**

**import** java**.**util**.**Scanner**;**

**public** **class** QuickSort

**{**

**public** **static** **void** main**(**String**[]** args**)** **{**

runQuickSort**();**

**}**

**private** **static** **void** runQuickSort**()** **{**

**int[]** numbsArray **=** getNumbers**();**

**int** arrayLength **=** numbsArray**.**length**;**

System**.**out**.**println**(**"\nOriginal order of the entered numbers: "**);**

printNumbers**(**numbsArray**,** arrayLength**);**

**long** startTime **=** System**.**nanoTime**();**

makeQuickSort**(**numbsArray**,** 0**,** arrayLength **-** 1**);**

**long** endTime **=** System**.**nanoTime**();**

System**.**out**.**println**(**"\nAfter sorting order of numbers: "**);**

printNumbers**(**numbsArray**,** arrayLength**);**

**long** timeElapsed **=** endTime **-** startTime**;**

System**.**out**.**println**(**"\nExecution time in nanoseconds: " **+** timeElapsed**);**

System**.**out**.**println**(**"Execution time in milliseconds: " **+** timeElapsed **/** 1000000**);**

System**.**out**.**println**();**

**}**

**private** **static** **int[]** getNumbers**()** **{**

System**.**out**.**println**(**"Podaj dowolne cyfry przedzielone spacjami"**);**

Scanner input **=** **new** Scanner**(**System**.**in**);**

**int[]** vals **=** **new** **int[**0**];**

**if** **(**input**.**hasNextLine**())** **{**

String line **=** input**.**nextLine**();**

String**[]** arr **=** line**.**split**(**"\\s+"**);**

vals **=** **new** **int[**arr**.**length**];**

**for** **(int** i **=** 0**;** i **<** arr**.**length**;** i**++)** **{**

vals**[**i**]** **=** Integer**.**parseInt**(**arr**[**i**]);**

**}**

System**.**out**.**println**(**Arrays**.**toString**(**vals**));**

**}**

**return** vals**;**

**}**

***/\* function that consider last element as pivot,***

***place the pivot at its exact position, and place***

***smaller elements to left of pivot and greater***

***elements to right of pivot. \*/***

**private** **static** **int** partition **(int[]** numbsArray**,** **int** start**,** **int** end**)**

**{**

**int** pivot **=** numbsArray**[**end**];** ***// pivot element***

**int** index **=** **(**start **-** 1**);**

**for** **(int** j **=** start**;** j **<=** end **-** 1**;** j**++)**

**{**

***// If current element is smaller than the pivot***

**if** **(**numbsArray**[**j**]** **<** pivot**)**

**{**

index**++;** ***// increment index of smaller element***

**int** t **=** numbsArray**[**index**];**

numbsArray**[**index**]** **=** numbsArray**[**j**];**

numbsArray**[**j**]** **=** t**;**

**}**

**}**

**int** boundaryIndicator **=** numbsArray**[**index**+**1**];**

numbsArray**[**index**+**1**]** **=** numbsArray**[**end**];**

numbsArray**[**end**]** **=** boundaryIndicator**;**

**return** **(**index **+** 1**);**

**}**

***/\* function to implement quick sort \*/***

**private** **static** **void** makeQuickSort**(int[]** numbsArray**,** **int** start**,** **int** end**)**

**{**

**if** **(**start **<** end**)**

**{**

**int** p **=** partition**(**numbsArray**,** start**,** end**);** ***//p is partitioning index***

makeQuickSort**(**numbsArray**,** start**,** p **-** 1**);**

makeQuickSort**(**numbsArray**,** p **+** 1**,** end**);**

**}**

**}**

***/\* function to print an array \*/***

**private** **static** **void** printNumbers**(int[]** numbsArray**,** **int** n**)**

**{**

**int** i**;**

**for** **(**i **=** 0**;** i **<** n**;** i**++)**

System**.**out**.**print**(**numbsArray**[**i**]** **+** " "**);**

**}**

**}**

### Instrukcja uruchamiania programu ZWasilonek\_54293\_QuickSort\_App

Program można uruchomić za pośrednictwem IDE (Integrated Development Environment), czyli zintegrowanego środowiska programistycznego (np. Intellij IDEA) lub w CLI (Command Line Interface), czyli w wierszu poleceń / terminalu.

Uruchamianie z poziomu CLI:   
  
**Krok 1.**   
Na początek należy upewnić się, że mamy zainstalowaną Jave na naszym systemie operacyjnym (instrukcja będzie pokazana dla systemu Windows 11). W tym celu należy przejść do CLI i wpisać komendę java -version  
Obraz zawierający tekst

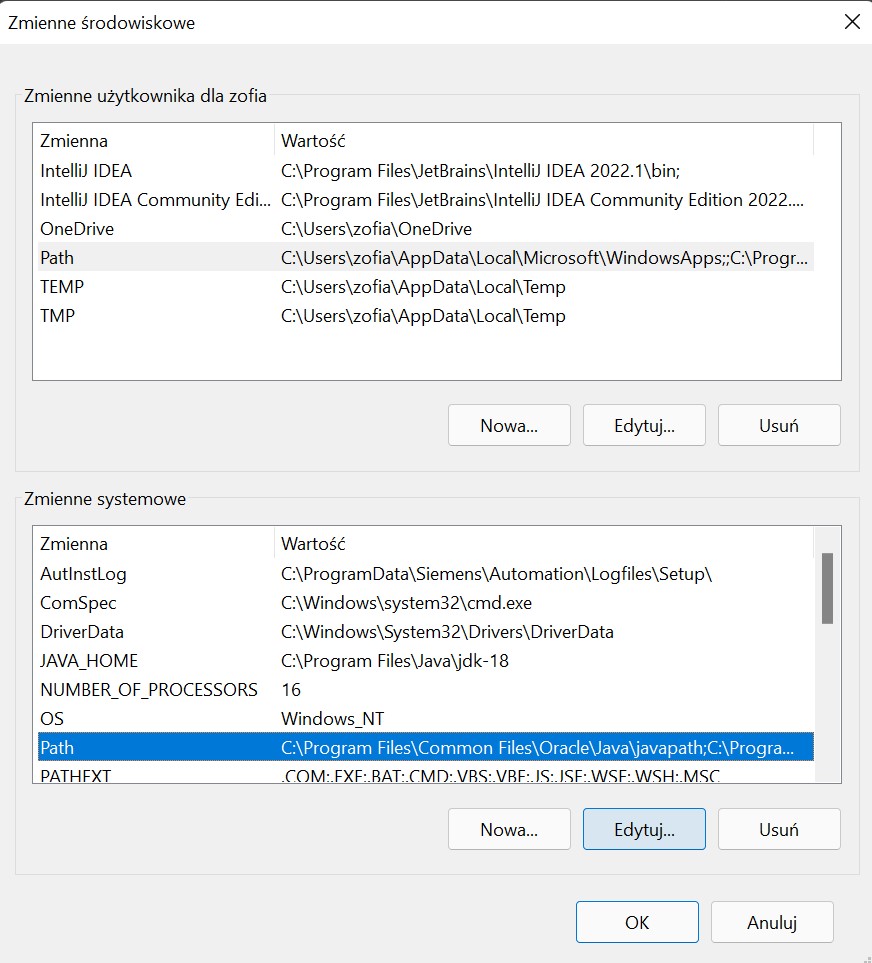
Opis wygenerowany automatycznie Jeśli pojawi się komunikat z podaną wersją, możemy przejść do kroku 2, a dla komunikatu z błędem należy najpierw upewnić się czy JDK (zalecam przynajmniej wersję 8.0 i nowsze) zostało prawidłowo zainstalowane (można pobrać za darmo z [https://www.oracle.com/java/technologies/downloads/)](https://www.oracle.com/java/technologies/downloads/) a następnie wykonać następującą konfigurację aby nasz system polecenia java / javac mógł wykonać globalnie z dowolnej ścieżki systemowej:

* Wejdź w Ustawienia/System/Zaawansowane ustawienia systemu i przejdź do zakładki Zaawansowane a następnie do Zmienne środowiskowe

Obraz zawierający tekst

Opis wygenerowany automatycznie

* Wyszukujemy Path i klikamy Edytuj…

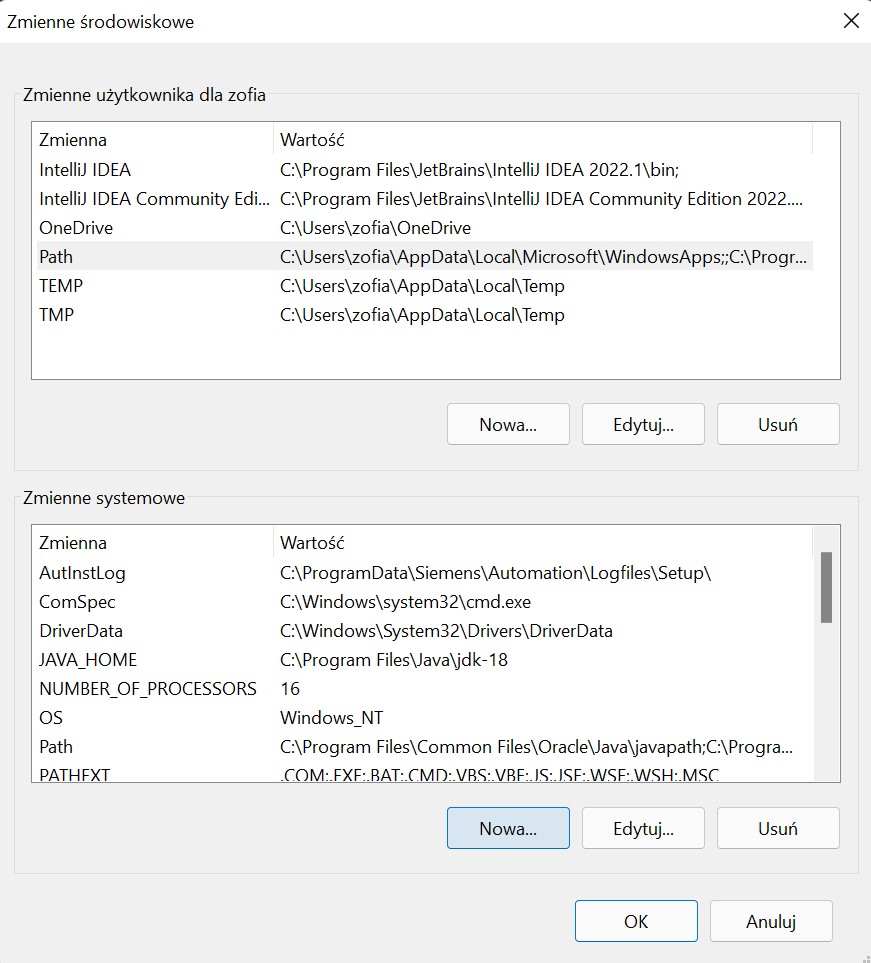


* Dodajemy nową ścieżkę, która jest ścieżką do zainstalowanego na komputerze JDK wraz z folderem „bin” i zatwierdzamy przyciskiem OK.

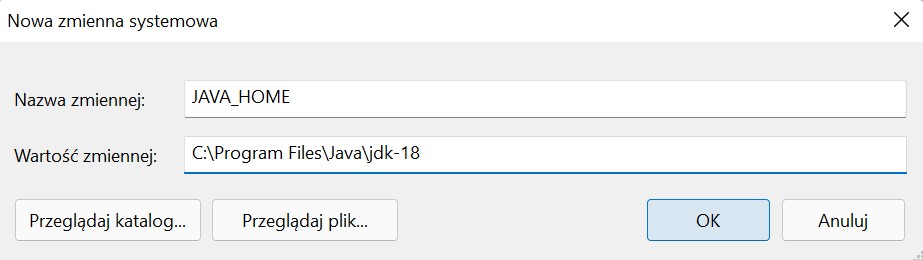
Obraz zawierający tekst

Opis wygenerowany automatycznie

* Dodajemy nową ścieżkę „JAVA\_HOME, jeśli takiej jeszcze nie ma, klikając Nowa…



* Wpisujemy JAVA\_HOME oraz ścieżkę do zainstalowanego JDK i zatwierdzamy klikając OK



* Zatwierdzamy wszystkie zmiany klikając OK

Obraz zawierający tekst

Opis wygenerowany automatycznie

**Krok 2.**

Uruchamiamy CLI i przechodzimy do ścieżki, w której znajduje się program

ZWasilonek\_54293\_QuickSort\_App

* Przejdź do folderu src – cd src
* Wykonaj kompilację kodu java – javac QuickSort.java
* Uruchom skompilowany program – java QuickSort
* Gotowe, dalej postępuj zgodnie z instrukcją programu

Obraz zawierający tekst

Opis wygenerowany automatycznie