

Niniejszy projekt opisuje dwa zaawansowane algorytmy sortowania. Są to algorytmy: Quick Sort oraz Shell Sort. W projekcie tym skupimy się na złożoności obliczeniowej oraz porównaniu czasów uzyskanych dla różnych porcji danych.

Aby sprostać wymienionemu zadaniu został utworzony harmonogram prac, dzięki któremu jesteśmy w stanie określić tempo wykonywanych prac i ustanowić czas pracy nad projektem. Praca będzie polegać na równomiernym rozłożeniu zadań projektowych między członków zespołu i wykonywanie ich w ściśle określonych przedziałach czasowych. Plan powstał odpowiednio wcześnie uwzględniając pozostały czas do końcowego terminu, stąd można założyć iż projekt zostanie zrealizowany w stu procentach.

Zdecydowaliśmy o wyborze technologii języka Java przy pracach nad projektem. Decyzja ta została uwarunkowana doświadczeniem w pracy w tym języku jakim może popisać się zespół. Kolejnym z powodów jest uniezależnienie programów napisanych w Javie od platformy, co likwiduje problem przenośności tworzonego oprogramowania dla celów prezentacji. Szybkość działania aplikacji Javowych upewniła nas w podjęciu decyzji wyboru tej technologii jak i świadomość, iż jest to technologia często wykorzystywana w dzisiejszym świecie.

**Wymagania funkcjonalne:**  
Funkcja sortująca, do której użytkownik podaję jako argument tablicę liczb typu Integer lub Float. Po wykonaniu zwraca ona tablice posortowanych elementów oraz czas w sekundach, jaki był potrzebny do przeprowadzenia operacji.  
Graficzny interfejs, za pomocą którego użytkownik może wybrać rodzaj sortowania, z którego chce skorzystać oraz pole, do którego użytkownik może podać tablicę liczb.  
  
**Wymaganie niefunkcjonalne:**  
Funkcja sortująca powinna wykonać operację w jak najszybszym czasie, z gwarancją niezawodności za każdym jej użyciem. Funkcja powinna być zaimplementowana w taki sposób, żeby złożoność obliczeniowa była jak najmniejsza.

Graficzny interfejs powinien być prosty w obsłudze dla każdego użytkownika.

**Architektura**

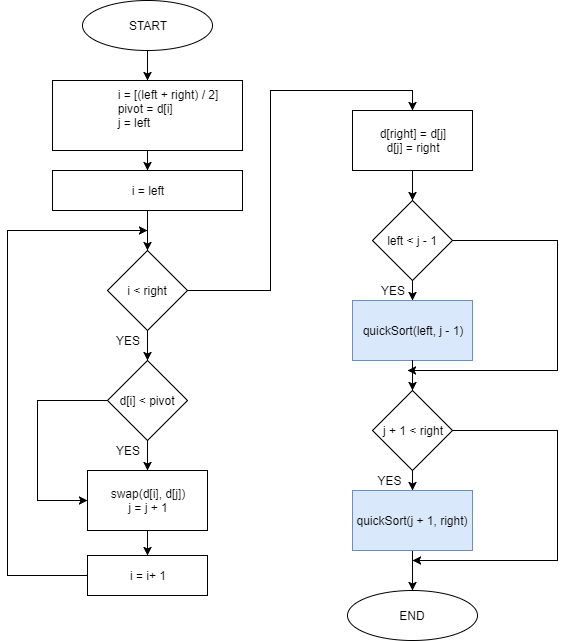
INTERFEJS GRAFICZNY

LOGIKA BIZNESOWA

Aplikacja będzie się składać z interfejsu użytkownika w którym znajdzie się wybór algorytmu sortującego, podanie danych wejsciowych oraz po wykonaniu operacji – wyświetlenie posortowanego zbioru liczb z czasem operacji. Ta warstwa komunikuje się z wartwą logiki biznesowej w której będą zaimplementowane algorytmy sortujące.

**Sortowanie Quicksort:**

**Schemat blokowy**



**Opis działania algorytmu:**

Funkcja sortująca jako argumenty przyjmuję indeks lewego oraz prawego elementu. Następnie z tablicy liczb wprowadzonej od użytkownika wybieramy element leżący na środku dzielonej partycji, po czym go zapisujemy do zmiennej i.

W zmiennej pivot zapisujemy element d[i], a w d[i] zapisujemy ostatni element partycji.

Kolejnym krokiem jest ustawienie zmiennej j na początek partycji, zmienna ta zapamiętuje pozycję podziału partycji.

Następnie w pętli sterowanej zmienną i przeglądamy kolejne elementy od pierwszego do przedostatniego (ostatni mamy zapamiętany jako pivot). Po czym sprawdzamy czy i-ty element jest mniejszy od pivotu, jeśli tak , to trafia on na początek partycji – wymieniamy ze sobą elementy na pozycjach i-tej i j-tej. Po tej operacji przesuwamy punkt przedziałowy partycji j.

Gdy działanie pętli się zakończy, to element j-ty przenosimy na koniec partycji, żeby uzyskać miejsce dla pivotu. Pierwotna partycja została podzielona na dwie partycje:

* Lewą – od pozycji left do j – 1, zawiera elementy mniejsze od piboty
* Prawą – od pozycji j + 1 do right, zawiera elementy większe lub równe od piwotu

Sprawdzamy, czy partycje te obejmują więcej niż jeden element. Jeśli tak, to wywołujemy rekurencyjnie algorytm sortowania przekazując mu granice wyznaczonych partycji. Po powrocie z wywołań rekurencyjnych partycja wyjściowa jest posortowana rosnąco.