

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. Prace odbywać się będą w cyklach cotygodniowych.

|  |  |  |
| --- | --- | --- |
| 1 | 21.10 - 28.10 | Rozpoczęcie prac nad projektem, Ustalenie wymagań, Wybór technologii |
| 2 | 29.10 - 4.11 | Opracowanie założeń, Decyzja wyboru prezentowanych algorytmów |
| 3 | 5.11 - 11.11 | Projektowanie aplikacji, Implementacja algorytmów |
| 4 | 12.11 - 18.11 | Zaprojektowanie GUI, Napisanie 1. referatu |
| 5 | 19.11 - 25.11 | Testowanie aplikacji |
| 6 | 26.11 - 2.12 | Korekcja błędów, wprowadzanie ewentualnych zmian/rozszerzeń |
| 7 | 3.12 - 9.12 | Praca nad ostateczną wersją programu, Napisanie 2. referatu |
| 8 | 10.12 - 16.12 | Ostateczne przeredagowanie dokumentacji projektowej. |

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 język Java zapewnia skalowalność i wydajność programów napisanych z jego użyciem.

**Wymagania**

|  |  |  |
| --- | --- | --- |
| W1 | QoS | Aplikacja musi działać niezawodnie na różnych konfiguracjach sprzętowych |
| W2 | Sortowanie liczb | Funkcja, która powinna zwracać tablicę posortowanych elementów oraz czas jaki był potrzebny do przeprowadzenia operacji |
| W3 | Wprowadzenie danych | Podanie liczb, które mają być poddane sortowaniu |
| W4 | Wybór algorytmu | Możliwość wyboru algorytmu, który zostanie użyty do sortowania. |
| W5 | Graficzny interfejs | Prosty w obsłudze interfejs umożliwiający korzystanie z aplikacji |

**Architektura**

INTERFEJS GRAFICZNY

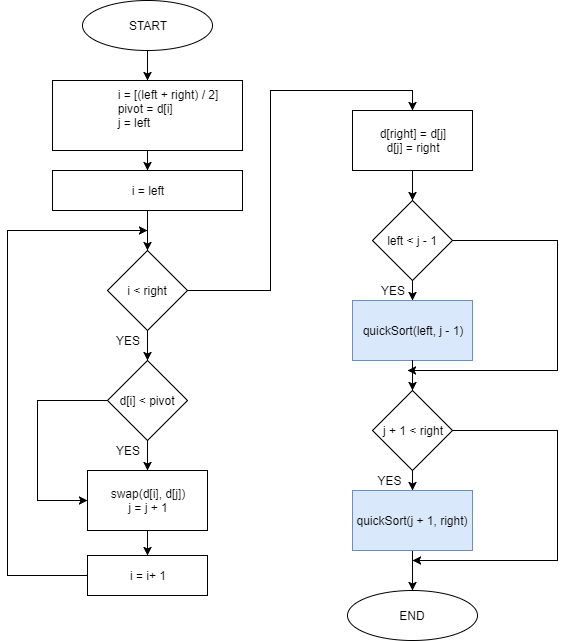
LOGIKA BIZNESOWA

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

**Zastosowane algorytmy**

**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.

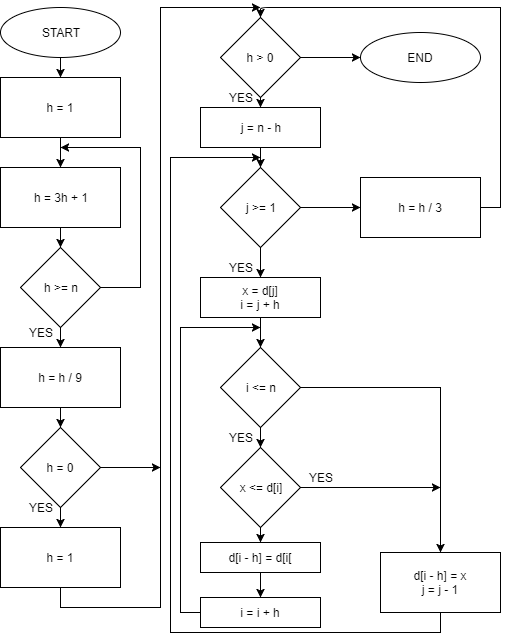
Gdy działanie pętli się zakończy, to element j-ty przenosimy na koniec partycji, żeby uzyskać miejsce dla pivotu. Pierwotna partycja została podzielona na dwie partycje:

* 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.

**Sortowanie Shella:**

**Schemat blokowy**

****

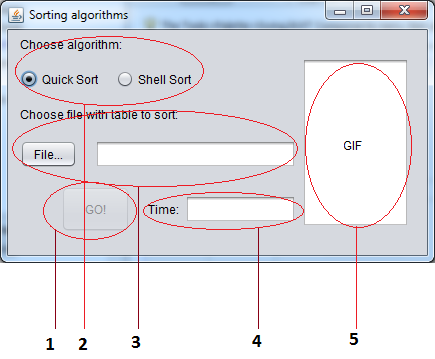
**Opis działania algorytmu:**

Na początku algorytmu wyznaczamy wartość początkowego odstępu h.

Po wyznaczeniu h rozpoczynamy pętlę warunkową. Pętla ta jest wykonywana dotąd, aż odstęp h przyjmie wartość 0. Wtedy kończymy algorytm, zbiór będzie posortowany.

Wewnątrz pętli umieszczony jest algorytm sortowania przez wstawianie, który dokonuje sortowania elementów poszczególnych podzbiorów wyznaczonych przez odstęp h. Po zakończeniu sortowania podzbiorów odstęp h jest zmniejszany i następuje powrót na początek pętli warunkowej.

**Warstwa GUI**



1 - Przycisk uruchamiający wybrany algorytm sortujący na podanym zbiorze danych. Przycisk uaktywnia się po spełnieniu dwóch warunków: Wyborze jednego z algorytmów oraz wskazaniu pliku z danymi do posortowania.

2 - Przyciski typu Radio służące do wyboru jednego algorytmu.

3 - Panel wyboru pliku z danymi do posortowania.

4 - Okienko z czasem działania algorytmu.

5 - Animacja GIF obrazująca sposób działania wybranego algorytmu.