Politechnika Krakowska
im. Tadeusza Kościuszki
Wydział Fizyki,
Matematyki i Informatyki

Modelowanie Procesów Dyskretnych

Projekt: Zagadnienie szeregowania zadań w systemach przepływowych. Implementacja algorytmu Johnsona dla systemu dwumaszynowego i trójmaszynowego.

Data oddania:
26-11-2016

Studia magisterskie
Kierunek: Informatyka Stosowana
Rok akademicki: 2016/2017

Wykonał: Dominik Freicher
Nr albumu: 120525

SPIS TREŚCI

1. WSTĘP

Celem projektu jest zapoznanie się z zagadnieniem szeregowania zadań w systemach przepływowych oraz implementacja algorytmu Johnsona dla systemu dwumaszynowego oraz trójmaszynowego w dowolnym języku programowania. W niniejszym referacie dokonam charakterystyki algorytmu Johnsona oraz przedstawię technologie, które zastosowane zostaną podczas implementacji wyżej wymienionego algorytmu Johnsona.

2. SZEREGOWANIE ZADAŃ W SYSTEMACH PRZEPŁYWOWYCH

W systemie przepływowym każde zadanie musi przejść przez wszystkie maszyny w ściśle określonym porządku, każde zadanie składa się z *m* operacji. Dla idealnego systemu przepływowego liczba zadań jest równa liczbie maszyn dla każdego zlecenia. W nieidealnym systemie przepływowym kierunek przepływu wszystkich zleceń jest ten sam, ale liczba zadań i użyte maszyny zależą od konkretnego zlecenia. Warto wspomnieć, że system przepływowy nie tylko jest związany z kolejnością wykonywania zleceń na maszynach lecz także z liczbą maszyn, na których realizowane są zlecenia. Systemy przepływowe możemy podzielić więc na systemy przepływowe z dwoma maszynami oraz z liczbą maszyn większą niż dwie. W przypadku systemu z dwoma maszynami, wynik jest realizowany stosując algorytm Johnsona. W przypadku liczby maszyn większej od dwóch, wynik realizowany jest dzięki regule Johnsona.

3. ALGORYTM JOHNSONA

Algorytm Johnsona dotyczy zagadnienia szeregowania zadań. Algorytm ten ma za zadanie ustalić optymalną kolejność wykonywania zadań na dwóch maszynach. W przypadku liczby maszyn większej od dwóch tak jak zostało wspomniane powyżej wynik realizowany jest dzięki regule Johnsona. Algorytm Johnsona charakteryzuje się prostotą i małym zakresem obliczeń numerycznych. Dla algorytmu Johnsona dla dwóch maszyn, określa się ilość zadań oraz dla obu maszyn dla każdego zadania przypisuje się określony czas wykonania zadania. Algorytm taki można przedstawić w trzech krokach.

1. Należy wybrać dowolne zadanie o najkrótszym czasie wykonania jednej z dwóch operacji. Jeżeli najkrótszy czas jest dla pierwszej maszyny wykonać to zadanie jako pierwsze, jeżeli najkrótszy czas jest dla drugiej maszyny wykonać jako ostatnie.

2. Usunąć rozpatrywane zadanie z dalszych rozważań.

3. Powtórzyć kroki nr 1 oraz nr 2 dla wszystkich pozostałych zadań.

Reguła Johnsona z kolei polega na utrzymywaniu tej samej kolejności zleceń na wszystkich maszynach systemu przepływowego. Jest ona najczęściej stosowana dla liczby maszyn >= 3. Algorytm postępowania w takim wypadku możemy zapisać następująco:

1. Wyznacz rozwiązanie otrzymanego problemu 2 - maszynowego za pomocą algorytmu Johnsona.

2. Uszereguj operacje na m-maszynach zgodnie z harmonogramem wyznaczonym w kroku 2.

4. ZASTOSOWANE TECHNOLOGIE

Do implementacji algorytmu Johnsona użyty został język programowania Java w raz z biblioteką Swing, która pozwoliła na stworzenie warstwy graficznego interfejsu użytkownika. Szczegółowa charakterystyka tych dwóch technologii została przedstawiona poniżej.

Rysunek 1 Zastosowane technologie

4.1 JĘZYK PROGRAMOWANIA JAVA

Do implementacji warstwy logiki użyty został język programowania Java. Jest językiem programowania zorientowanym obiektowo, cechującym się mocnym typowaniem. Zaprojektowany i stworzony został przez firmę Sun Microsystem. Głównym projektantem języka Java był James Gosling. Pierwsza wersja standardu opublikowana została w roku 1995. Z pojęciem języka Java nierozłącznie wiąże się termin wirtualnej maszyny (ang. Java Virtual Machine), czyli środowiskiem uruchomieniowym dla języka Java. JVM odpowiada za wykonywanie kodu bajtowego Javy. Warto wspomnieć, że Java występuje w trzech wersjach zaprojektowanych pod odmienne platformy:

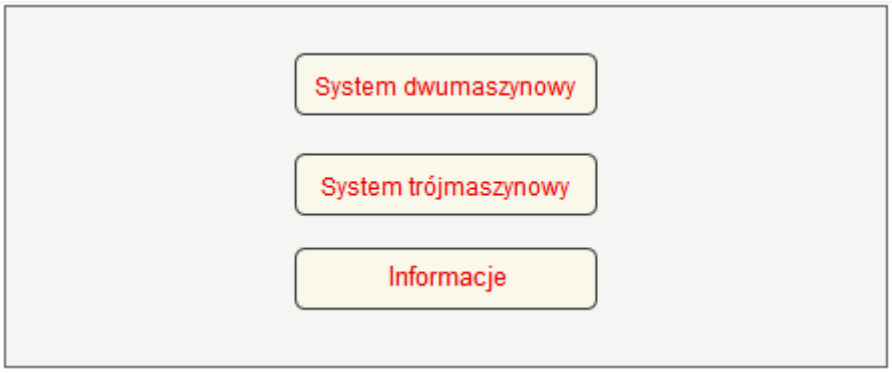
- JavaSE (ang. Java Standard Edition) - standardowo odmiana języka Java.
- JavaEE (ang. Java Enterprise Edition) - wersja korporacyjna języka pozwalająca na tworzenie wielowarstwowych aplikacji serwerowych.
- JavaME (ang. Java Mobile Edition) - wersja języka Java przeznaczona do zastosowania z wykorzystaniem urządzeń mobilnych.

4.2 BIBLIOTEKA GRAFICZNA SWING

Do implementacji warstwy interfejsu graficznego użytkownika wykorzystana została biblioteka Swing. Biblioteki Swing jest oficjalnym zestawem narzędzi Java GUI (Interfejs Graficzny Urządzenia), wydanym przez firmę Sun Microsystems. Posiada bogaty zestaw widżetów. Od podstawowych widżetów takich jak przyciski, etykiety, paski przewijania do zaawansowanych widżetów, takich jak drzewa i tabele. Pozwala tworzyć interfejsy graficzne niezależne od konkretnej platformy sprzętowej, co z kolei czyni ją mniej podatną na błędy związane z konkretną platformą sprzętową. Warto nadmienić, że Swing jest częścią JFC, Java Foundation Classes. Służy jako zbiór pakietów do tworzenia w pełni funkcjonalnych aplikacji desktopowych.

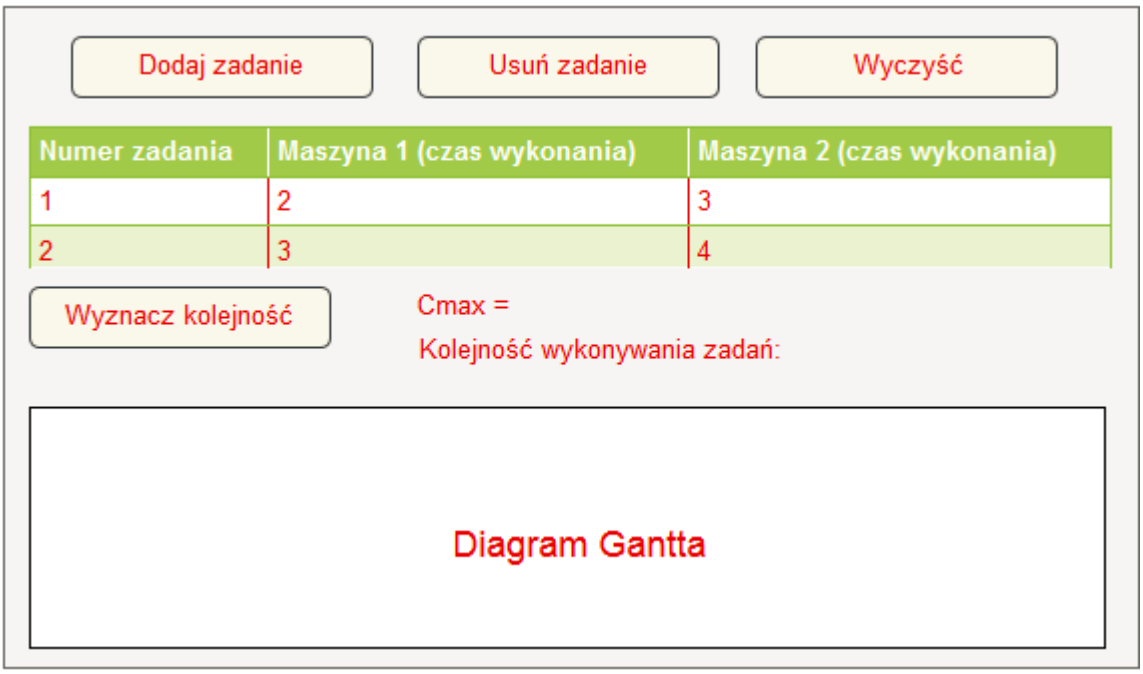
5. KONCEPT INTERFEJSU UŻYTKOWNIKA

Interfejs użytkownika będzie się składał z trzech widoków pozwalających na nawigację pomiędzy funkcjonalnościami. Widok startowy będzie zawierał dwa przyciski umożliwiające wybór trybu pracy, a mianowicie uruchomienia obliczeń dla systemu dwumaszynowego lub trójmaszynowego. Koncepcja ekranu startowego zaprezentowana została poniżej.



Rysunek 2 Koncept widoku startowego

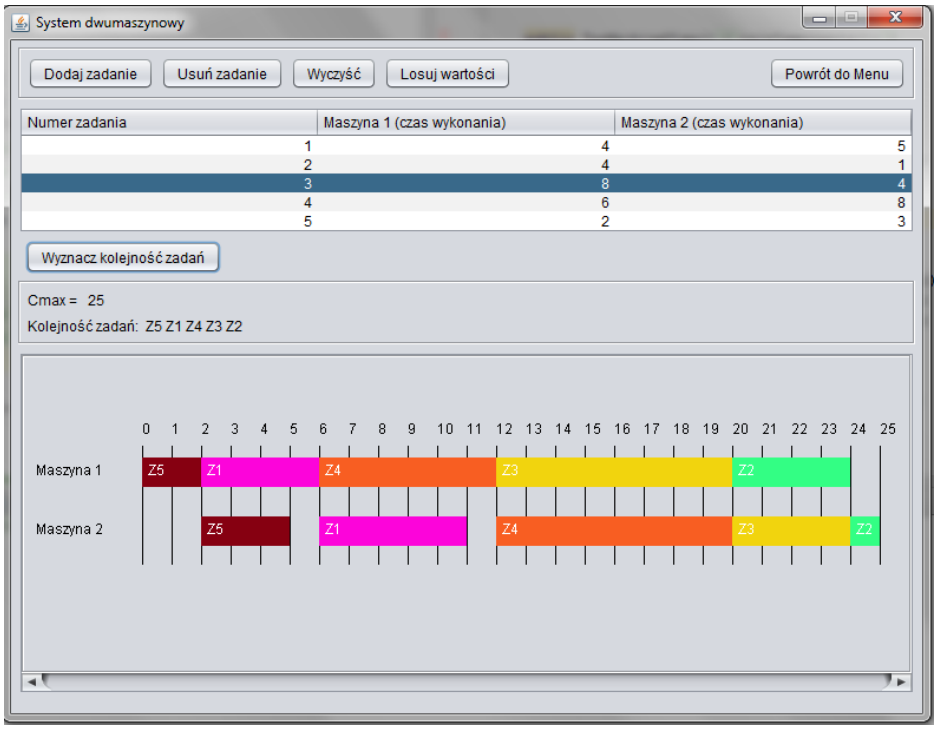
Kolejnymi widokami będą widoki szczegółowe dla systemu dwumaszynowego oraz trójmaszynowego. Każdy z widoków będzie się składał z formatki umożliwiającej dodanie nowych zadań oraz ich czasów. Na samej górze widoku znajdzie się menu nawigacyjne, które będzie zawierało operacje takie jak czyszczenie formatki, dodanie nowego zadania oraz uruchomienie obliczeń. Rezultat będzie zawierał kolejność wykonywania zadań, wartość funkcji kryterialnej Cmax oraz diagram Gantta, który będzie przedstawiał uszeregowanie zadań. Koncept ekranu szczegółowego zaprezentowany został poniżej.



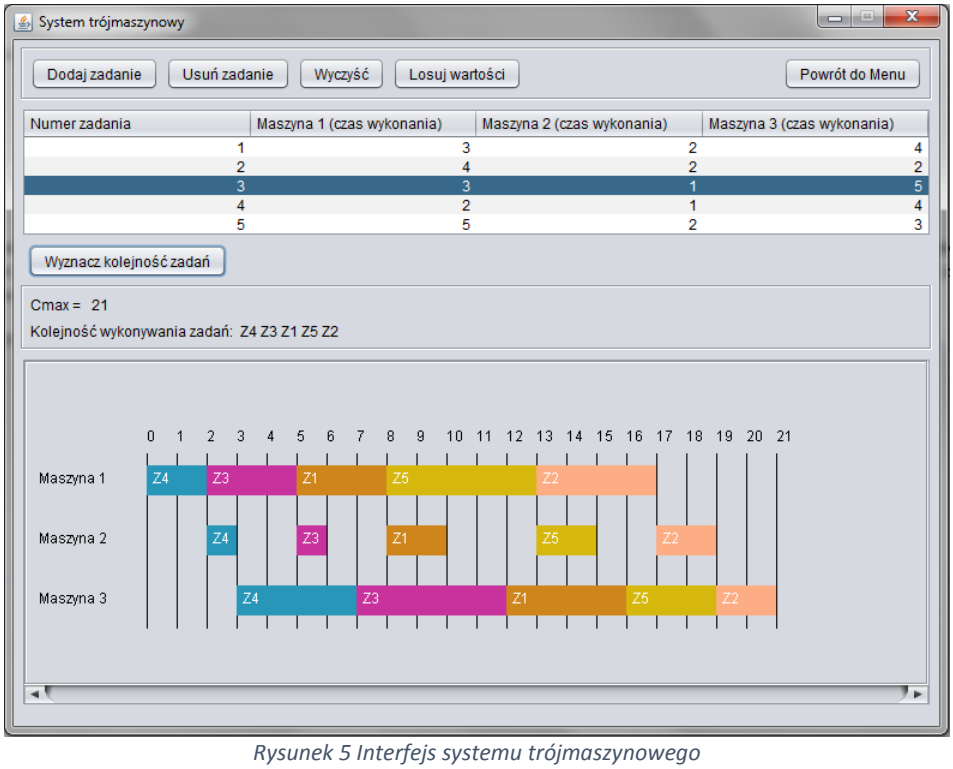
Rysunek 3 Koncept widoku szczegółowego

6. GRAFICZNY INTERFEJS UŻYTKOWNIKA

Interfejs użytkownika składa się z trzech widoków. Pierwszy widok to wybór trybu działania programu, kolejno widoki szczegółowe z użyciem dwóch maszyn oraz z użyciem trzech maszyn. Widok szczegółowy trybu pracy podzielony jest na dwie części. Pierwsza z nich służy do wprowadzania danych, natomiast druga część prezentuje wynik działania algorytmu. Graficzny interfejs użytkownika aplikacji prezentuje Rysunek 1. Użytkownik ma możliwość dodawania nowych zadań oraz ich usuwania. Poszczególne zadania wyświetlane są w tabeli, która składa się z trzech kolumn (dla trybu dwóch maszyn): numer zadania, maszyna 1, maszyna 2 oraz odpowiednio czterech kolumn dla trybu trzech maszyn: numer zadania, maszyna 1, maszyna 2, maszyna 3. Kolumny odnoszące się do maszyn określają czas wykonywania poszczególnych zadań na danej maszynie. Użytkownik ma możliwość wprowadzania dla konkretnego zadania, czasu wykonania zadania na maszynie. Aplikacja pozwala również na losowanie czasów wykonywania zadań na obu maszynach. Przycisk „czyść" pozwala na wyczyszczenie zawartości tabeli. Przycisk „wyznacz kolejność" rozpoczyna proces wykonania algorytmu Johnsona. W rezultacie aplikacja zwraca użytkownikowi informacje o optymalnej kolejności zadań, informacje o wartości funkcji kryterialnej Cmax oraz diagram Gantta, który w sposób graficzny prezentuje uszeregowanie zadań na dwóch lub trzech maszynach.



Rysunek 4 Interfejs systemu dwumaszynowego



Rysunek 5 Interfejs systemu trójmaszynowego

7. IMPLEMENTACJA

Interfejs graficzny stworzony został przy pomocy biblioteki Swing. Projekt interfejsu wykonany został w edytorze narzędzia Intellij Idea. W aplikacja wykorzystano komponenty biblioteki Swing takie jak: JFrame, JPanel, JButton, JTable, JComponent.

Struktura programu składa się z klas, w których wykonana została implementacja warstwy graficznego interfejsu użytkownika oraz logiki aplikacji. W projekcie ożemy wyróżnić następujące klasy:

- `MenuView.java` – implementacja GUI głównego menu.
- `TwoMachineView.java` – implementacja GUI systemu 2 maszynowego.
- `ThreeMachineView.java` – implementacja GUI systemu 3 maszynowego.
- `ThreeMachineChartComponent.java` – implementacja GUI wykresu.
- `TwoMachineChartComponent.java` – implementacja GUI wykresu.
- `Item.java` – pojedynczy obiekt wykresu.

Metody obsługujące akcje przycisków w widokach szczegółowych to kolejno:

- `addNewTask()` – dodanie nowego zadania do listy,
- `clearTasks()` – czyszczenie listy zadań,
- `removeTask()` – usunięcie zadania z listy,
- `randomValues()` – wylosowanie czasów dla zadań,
- `backToMenu()` – powrót do menu,
- `calculate()` – uruchomienie obliczeń algorytmu.

Wyżej wymienione metody mają swe zastosowanie dla widoków szczegółowych systemu dwumaszynowego oraz trójmaszynowego.

Szczegółowy algorytm dla systemu dwumaszynowego został opisany w rozdziale trzecim. Poniżej przedstawiony został sposób implementacji algorytmu w aplikacji:

```
List<Job> firstList = new ArrayList<>();
List<Job> secondList = new ArrayList<>();


for (int i = 0; i < table.getRowCount(); i++) {

    int firstMachineValue = (int) table.getModel().getValueAt(i, 1);
    int secondMachineValue = (int) table.getModel().getValueAt(i, 2);

    if (firstMachineValue <= secondMachineValue) {
        firstList.add(new Job("Z" + (i + 1), firstMachineValue));
    } else if (firstMachineValue > secondMachineValue) {
        secondList.add(new Job("Z" + (i + 1), secondMachineValue));
    }
}

Collections.sort(firstList);
Collections.sort(secondList, Collections.reverseOrder());
firstList.addAll(secondList);
```

Algorytm taki można przedstawić w trzech krokach.

1. Należy wybrać dowolne zadanie o najkrótszym czasie wykonania jednej z dwóch operacji. Jeżeli najkrótszy czas jest dla pierwszej maszyny wykonać to zadanie jako pierwsze, jeżeli najkrótszy czas jest dla drugiej maszyny wykonać jako ostatnie.

2. Usunąć rozpatrywane zadanie z dalszych rozważań.

3. Powtórzyć kroki nr 1 oraz nr 2 dla wszystkich pozostałych zadań.

Szczegółowy algorytm dla systemu trójmaszynowego został opisany w rozdziale trzecim. Poniżej przedstawiony został sposób implementacji algorytmu w aplikacji:

```
List<Job> firstList = new ArrayList<>();
List<Job> secondList = new ArrayList<>();

List<Integer> firstValuesList = new ArrayList<>();
List<Integer> secondValuesList = new ArrayList<>();

for (int i = 0; i < table.getRowCount(); i++) {

    int firstMachineValue = (int) table.getModel().getValueAt(i, 1);
    int secondMachineValue = (int) table.getModel().getValueAt(i, 2);
    int thirdMachineValue = (int) table.getModel().getValueAt(i, 3);

    firstValuesList.add(firstMachineValue + secondMachineValue);
    secondValuesList.add(secondMachineValue + thirdMachineValue);
}

for (int i = 0; i < firstValuesList.size(); i++) {
    if (firstValuesList.get(i) <= secondValuesList.get(i)) {
        firstList.add(new Job("Z" + (i + 1), firstValuesList.get(i)));
    } else if (firstValuesList.get(i) > secondValuesList.get(i)) {
        secondList.add(new Job("Z" + (i + 1), secondValuesList.get(i)));
    }
}

Collections.sort(firstList);
Collections.sort(secondList, Collections.reverseOrder());

firstList.addAll(secondList);
```

Algorytm postępowania w takim wypadku możemy zapisać następująco:

1. Wyznacz rozwiązanie otrzymanego problemu 2 - maszynowego za pomocą algorytmu Johnsona.

2. Uszereguj operacje na m-maszynach zgodnie z harmonogramem wyznaczonym w kroku 2.

Zaimplementowane zostały również testy jednostkowe, które dają możliwość testowania algorytmów systemów dwumaszynowego oraz trzymaszynowego w oderwaniu od warstwy graficznej aplikacji. Implementacja testów przedstawiona została poniżej.

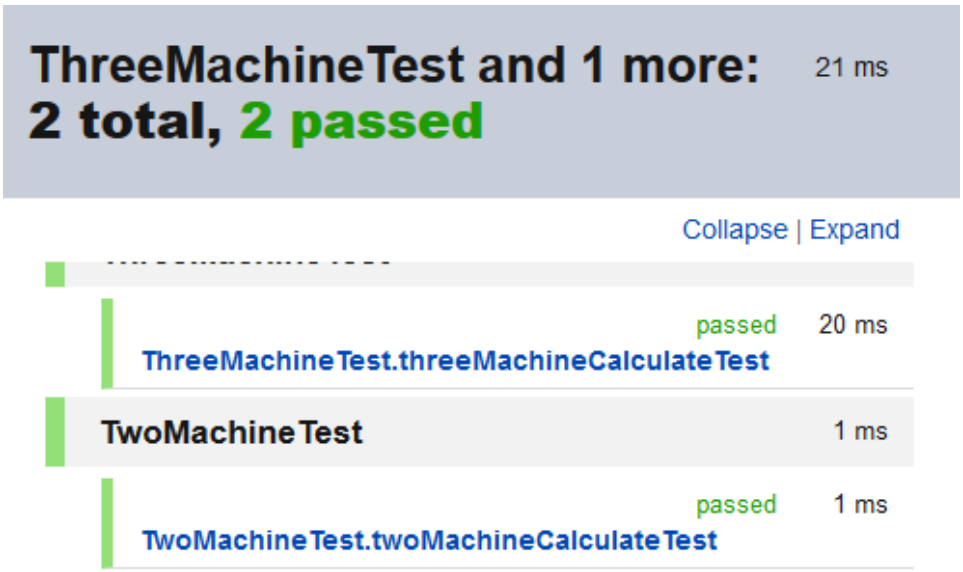
Test jednostkowy dla algorytmu dwumaszynowego:

```
@Test
public void twoMachineCalculateTest() {
    List<Integer> firstValuesList = Arrays.asList(4, 4, 8, 6, 2);
    List<Integer> secondValuesList = Arrays.asList(5, 1, 4, 8, 3);

    String result = twoMachineCalculate(firstValuesList, secondValuesList);
    assertEquals("Z5 Z1 Z4 Z3 Z2 ", result);
}
```

Test jednostkowy dla algorytmu trójmaszynowego:

```
@Test
public void threeMachineCalculateTest() {
    List<Integer> firstValuesList = Arrays.asList(3, 4, 3, 2, 5);
    List<Integer> secondValuesList = Arrays.asList(2, 2, 1, 1, 2);
    List<Integer> threeValuesList = Arrays.asList(4, 2, 5, 4, 3);

    String result = threeMachineCalculate(firstValuesList, secondValuesList, threeValuesList);
    assertEquals("Z4 Z3 Z1 Z5 Z2 ", result);
}
```

Wyniki:



8. WYMAGANIA SPRZĘTOWE

Aplikacja została zaimplementowana w języku Java, do uruchomienia aplikacji wymagane jest środowisko uruchomieniowe Javy. Aplikacja może być uruchomiana z użyciem systemu operacyjnego, na którym jest możliwość zainstalowania środowiska uruchomieniowego Javy.

Za uruchomienie aplikacji odpowiedzialny jest plik o nazwie MPDFREICHER.jar. Uruchomienie aplikacji możliwe jest za pośrednictwem linii komend z użyciem polecenia:

```
java -jar MPDFREICHER.jar
```

9. PODSUMOWANIE

Celem niniejszego projektu było zaimplementowanie algorytmu Johnsona dla systemu dwumaszynowego i trójmaszynowego. Implementacja została wykonana w języku Java z wykorzystaniem biblioteki Swing. Aplikacja pozwala na znajdowanie optymalnej kolejności wykonywania zadań na dwóch oraz trzech maszynach. Program może być wykonywany z użyciem dowolnego systemu operacyjnego, który pozwala na zainstalowanie środowiska uruchomieniowego Javy.

10. BIBLIOGRAFIA

1. Deterministyczne szeregowanie zadań,. http://riad.pk.edu.pl/~ljamroz/nwmpdstac/mpd%20sequencing%20www.pdf (dostęp online 21.11.2016).