Sprawozdanie z projektu

Kacper Karaś, Jakub Niewadzi

2.05.2022

Specyfikacja Funkcjonalna

1. Cel projektu

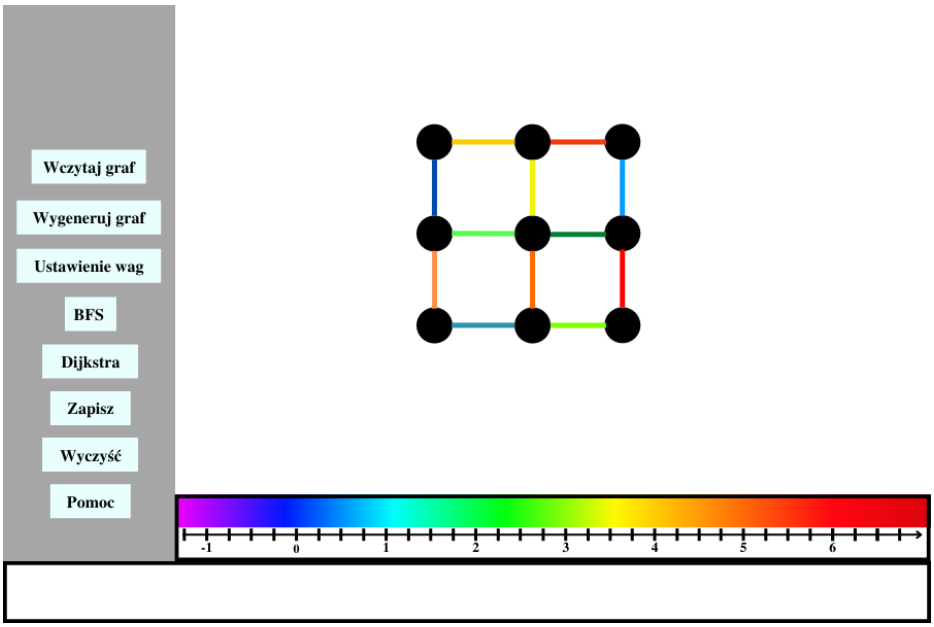
Program ma na celu generowanie, zapisywanie i czytanie grafów oraz powinien posiadać opcje sprawdzenia czy graf jest spójny za pomocą algorytmu BFS i wyznaczenia najkrótszej możliwej ścieżki pomiędzy wybraną parą węzłów, przy pomocy algorytmu Dijkstry.

2. Scenariusz działania programu

Program służy do badań grafu pod względem spójności wykorzystując algorytm BFS oraz znajduje w nim najkrótsze ścieżki za pomocą algorytmu Dijkstry. Program po uruchomieniu wyświetli nam okno interfejsu, gdzie po lewej stronie znajdują się przyciski z odpowiednimi funkcjami jakie może wykonać program. Wciśnięcie odpowiednich przycisków i podanie odpowiednich argumentów umożliwiających nam wykorzystanie programu w przewidzianym przez nas celu (więcej o formacie danych jak i argumentów będzie mowa w dalszej części specyfikacji). Jeżeli program nie znajdzie problemu przy wczytywaniu podanych przez nas argumentów, wyświetli graficzną interpretację grafu w centralnej części interfejsu. Wciśnięcie przycisku BFS lub Dijkstra rozpocznie analizę grafu odpowiednim algorytmem. Zostanie także zmieniona graficzna interpretacja grafu tak aby dobrze odwzorowywała wyniki działania algorytmów. Odpowiednie interpretacje graficzne zostaną szczegółowo opisane w dalszej części specyfikacji.

3. Interfejs programu

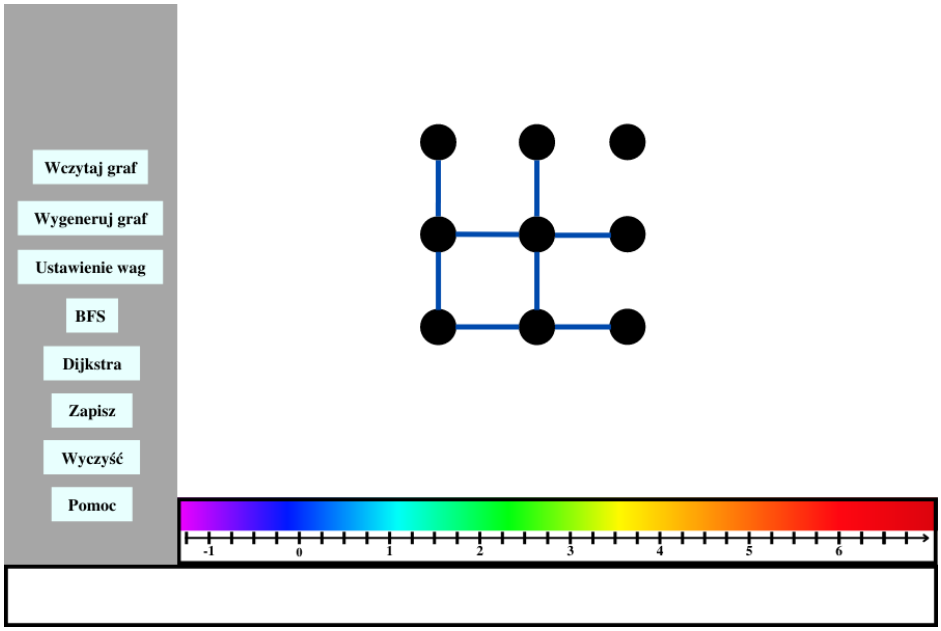
Interfejs programu po uruchomieniu programu i wgraniu pliku zawierającego graf:



Prototyp interfejsu

Kolory krawędzi oznaczają wagę jaka pomiędzy nimi występuje. W przytoczonym przykładzie wagi pomiędzy poszczególnymi węzłami są identyczne, ale program obsługuje również przypadki gdy te wartości są różne chociaż w tedy widoczność poszczególnych krawędzi może być utrudniona.

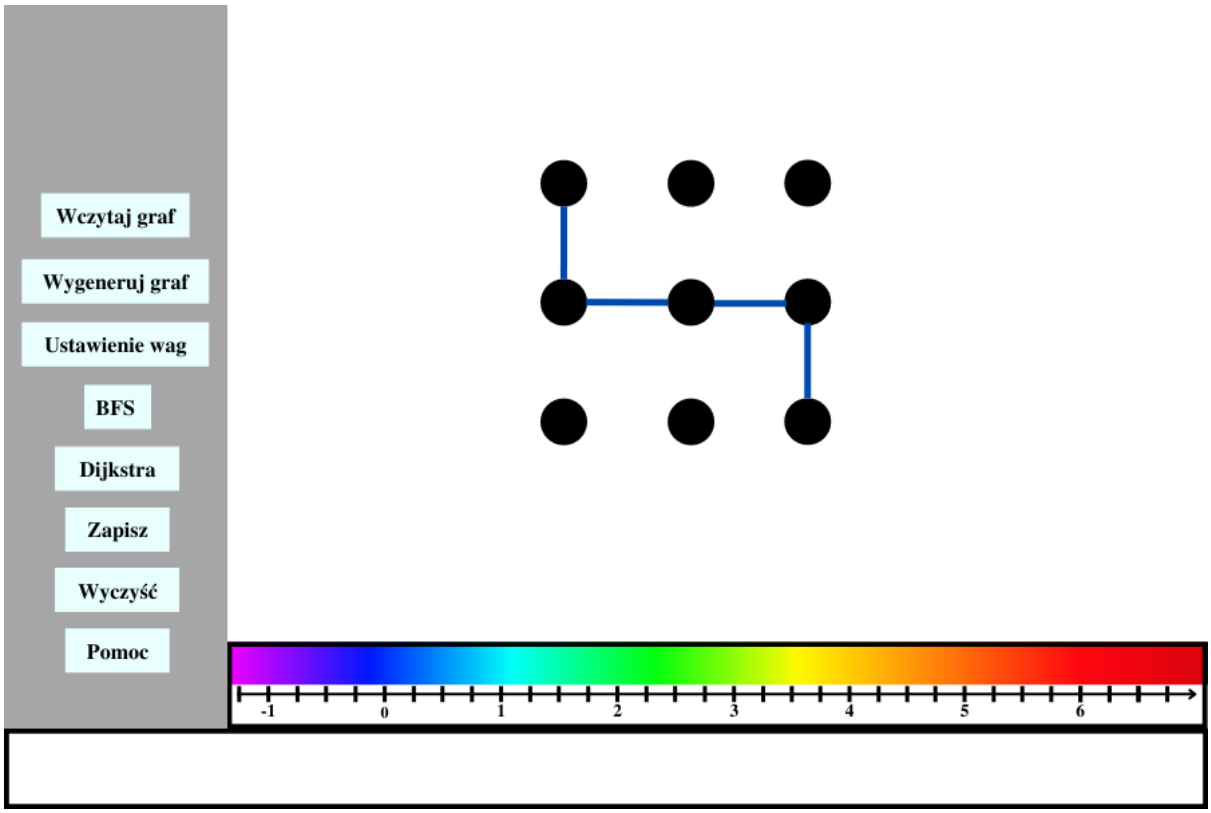
Interfejs programu po uruchomieniu programu i wgraniu pliku zawierającego graf i zanalizowania go algorytmem BFS:



Prototyp interfejsu

Kolor niebieski (waga 0) oznacza tu istniejące połączenie zaś nie istniejące połączenia nie zostały narysowane.

Interfejs programu po uruchomieniu programu i wgraniu pliku zawierającego graf i zanalizowania go algorytmem Djikstry:



Prototyp interfejsu

Połączenia koloru niebieski (waga 0) oznaczają tu najkrótszą drogę pomiędzy dwoma wierzchołkami. Reszta połączeń nie została narysowana, gdyż inne ścieżeki nie stanowią zainteresowania użytkownika .

Opis odpowiednich przycisków Interfejsu:

- **Wczytaj graf** – po wciśnięciu przycisku wyświetli się nam okno wyboru pliku z komputera, z którego program będzie odczytywał graf
- **Wygeneruj graf** – po wciśnięciu przycisku wyświetli się okno z zapytaniem jakich wymiarów chcemy wygenerować graf, a następnie generuje go z odpowiednimi wagami na ekranie i jeżeli podaliśmy plik do zapisu zapisze go również tam
- **Ustawienie wag** – po wciśnięciu przycisku wyświetli nam się okno, w którym należy podać wagi jakie chcemy, żeby miał generowany przez nas graf (program po uruchomieniu ma ustawione wagi 0 – 10)
- **Zapisz** – po wyciśnięciu przycisku wyświetli się nam okno wyboru pliku z komputera, w którym chcemy zapisać generowany graf
- **Wyczyść** - po wyciśnięciu przycisku wyświetla ponownie interfejs z podstawowym wyglądem grafu
- **Pomoc** - po wyciśnięciu przycisku wyświetla pomoc w jaki sposób prawidłowo korzystać z programu

4. Odpowiedni format argumentów

Podczas używania programu użytkownik będzie podawał pewną ilość argumentów, które powinny być w odpowiednim formacie:

- Program powinien mieć dostęp do pliku, z którego będzie czytał graf
- Rozmiary generowanego grafu powinny być liczbami naturalnymi dodatnimi
- Wagi podane przez użytkownika mogą być zmiennoprzecinkowe, ale muszą być nieujemne
- Gdy użytkownik poda plik do zapisu, który nie istnieje plik o takiej nazwie zostanie utworzony

5. Dane wejściowe

Program zakłada możliwość przekazania własnego pliku przechowującego informacje o ilości kolumn i wierszy w grafie oraz opisuje ilość krawędzi w każdym punkcie oraz ich wagi. Wagi krawędzi, gdy nie zostanie podany argument wejściowy mówiący inaczej, to liczby rzeczywiste z przedziału 0 – 10.

Plik ten powinien być sformatowany następująco:

- Dwoma pierwszymi liczbami powinno być liczba kolumn i wierszy grafu
- Następnie powinny pojawić się opisy wierzchołków, które występują w grafie (tu należy zaznaczyć, że numeracje zaczynamy od 0 i numerujemy w dół poszczególnych kolumn). Opisy wierzchołków powinny rozpoczynać się od pierwszego i kończyć na ostatnim, dlatego nie ma potrzeby zaznaczać jaki wierzchołek aktualnie opisujemy.
- Opis wierzchołka powinien składać się z co najwyżej 4 bloków zawierających numer sąsiadującego wierzchołka oraz wagę krawędzi między aktualnie wymienionym wierzchołkiem, a aktualnie opisywanym. Numer wierzchołka i waga powinny być od siebie oddzielone wyłącznie znakami spacji.

Przykładowy prawidłowo sformatowany plik:

```
3 3
      1 :1.232          3 :2.233
      0 :1.232          2 :0.485          4 :7.583
      1 :0.485          5 :6.593
      0 :2.233          4 :2.690          6 :9.515
      1 :7.583          3 :2.690          5 :10.000   7 :3.467
      2 :6.593          4 :10.000   8 :3.234
      3 :9.515          7 :7.383
      4 :3.467          6 :7.383          8 :8.487
      5 :3.234          7 :8.487
```

6. Teoria

Graf jest to uporządkowana para {**W, K**}, gdzie W to zbiór n węzłów odpowiednio ponumerowanych np. 0, 1, 2, …, n-1. K jest to zbiór krawędzi, czyli relacji między dwoma węzłami, które w naszym przypadku posiadają wagę (liczbę rzeczywistą).

Graf spójny - graf, dla którego spełniony jest warunek, że dla każdej pary węzłów istnieje ścieżka, która je łączy.

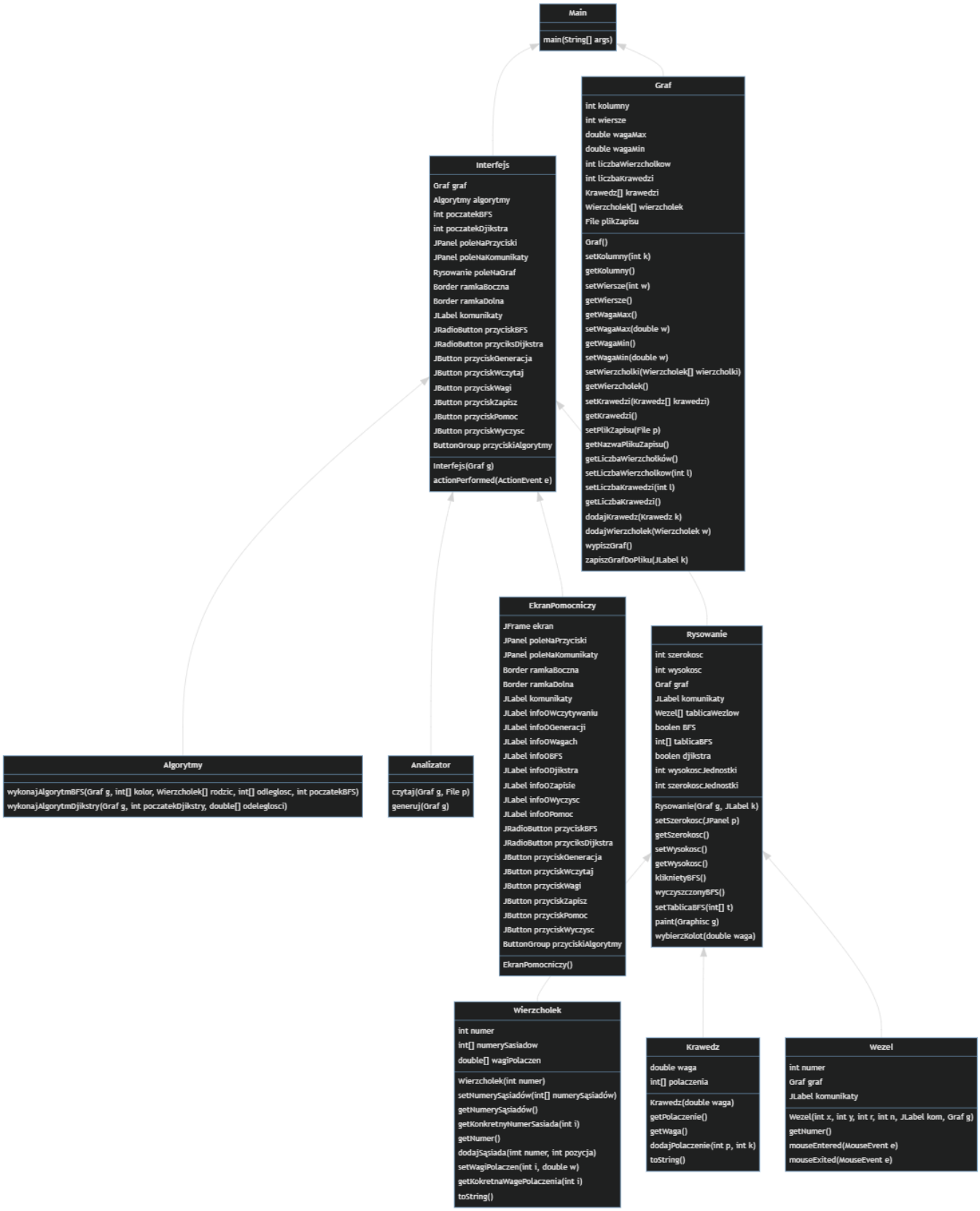
7. Komunikaty błędów

Program będzie kontynuował pracę po wpisaniu błędnych argumentów wtedy, kiedy jest to możliwe.

Oto lista komunikatów o błędach jakie mogą wystąpić podczas próby uruchomienia programu:

Specyfikacja implementacyjna programu.

Klasy w programie:



Poniżej znajduje się opis klas jakie będą używane w projekcie wraz z krótkim jakie metody w sobie zawierają orz jakie argumenty zawiera (jeżeli jakieś zawiera):

- **Main** - zawiera inicjację klasy Graf oraz Interfejs. Posiada metodę main.

- **Graf** - jest to klasa zawierająca wszelkie informacje o grafie analizowanym przez program. Zawiera atrybuty przechowujące takie informacje jak: ilość kolumn, ilość wierszy, maksymalną wagę, minimalną wagę, liczbę wierzchołków, liczbę krawędzi, tablice klas Krawedzi i Wierzcholkow oraz plik, w którym będzie zapisywany graf. Tam, gdzie to potrzebne atrybuty mają napisane gettery i settery. Metody znajdujące się w klasie to: dodajKrawedz, dodajWierzcholek i wypiszGraf. Klasa zawiera również konstruktor, w którym ustawiona są podstawowe wartości wag.

- **Interfejs –** jest to najważniejsza klasa programu to dzięki niej wyświetlany jest interfejs i obiekty znajdujące się na nim. To również tu znajduje się ActionLisiner służący do wywoływania odpowiednich metod w programie w zależności od akcji użytkownika. Klasa ta zawiera inicjalizację wszystkich obiektów interfejsu oraz w konstruktorze rozmieszcza je odpowiednio w oknie głównym.

- **Krawendz** - jest to klasa zawierająca wszelkie informacje na temat poszczególnych krawędzi w grafie. Posiada argumenty takie jak: waga i tablice polaczenie (zawierającą dwa elementy pierwszy to numer początkowego wierzchołka krawędzi, drugi to numer końcowego wierzchołka. W konstruktorze klasy przypisuje się wagę danej krawędzi. Klasa posiada metodę doddajPolaczenie.

- **Wierzcholek** - jest to klasa zawierająca wszelkie informacje na temat poszczególnych wierzchołków w grafie. Posiada argumenty takie jak numer i tablice zawierającą numery sąsiadów (sąsiedzi przechowywanie są w następującej kolejności: pierwsze miejsce tablicy to górny sąsiad, drugie prawy sąsiad, trzecie dolny i czwarte miejsce lewy). W konstruktorze klasy przypisuje się numer danego wierzchołka oraz tablice sąsiadów wypełnia się wartościami -1. Klasa posiada matody: setNumerySąsiadów, getNumerySąsiadów, getNumer, dodajSąsiada.

- **Analizator –** jest to klasa zawierające metody związane z analizą grafu. Jedną z nich jest metoda czytaj odczytująca z podanego pliku dane na temat grafu i zapisujące je w odpowiednich argumentach klasy Graf. Druga metoda zajmuje się generacją grafu o określonej ilości kolumn i wierzchołków.

- **Algorytmy –** jest to klasa zawierająca metody analizujące graf zapisany w klasie Graf. Klasa posiada dwie metody: pierwszą analizującą graf za pomocą algorytmu BFS i drugą odpowiadającą za analizę grafu algorytmem Djikstry.

- **Rysowanie** – jest to klasa odpowiedzialna za rysowanie odpowiednich rodzajów grafów w centralnym polu w oknie głównym programu.

- **Ekran pomocniczy –** jest to klasa odpowiedzialna za wyświetlenie ekranu pomocniczego, zawierającego nieaktywne elementy normalnego interfejsu programu oraz zawierającego opisy działania elementów interfejsu.

Opis testów jednostkowych