Specyfikacja programu ***grafexe***

v1.0

Sereda Krystian
Skarżyński Konrad

Link do repozytorium:

https://github.com/Konrad-Krystian-Jimp2/Projekt2

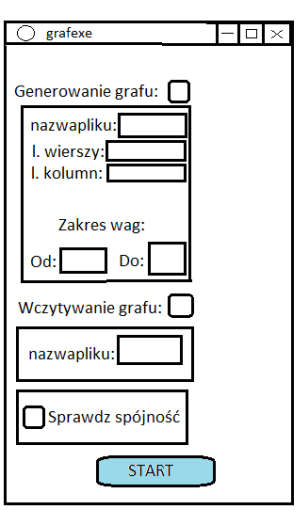
Cel projektu

Program *grafexe* wyznacza pomiędzy określonymi parami węzłów najkrótszą ścieżkę wykorzystując algorytm Dijkstry. Program umożliwia generowanie lub wczytanie grafu, sprawdzanie spójności grafu, poprzez wykorzystanie algorytmu BFS, oraz wyznaczanie najkrótszej ścieżki pomiędzy dowolną liczbą par węzłów.

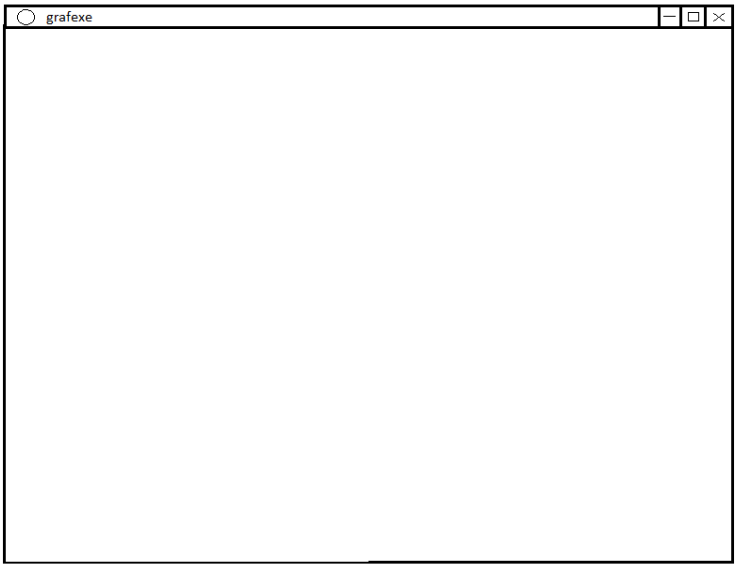
Scenariusze działania

1. Generowanie grafu spójnego lub losowego;
2. Wczytywanie grafu;
3. Sprawdzanie spójności grafu;
4. Znajdowanie najkrótszej ścieżki między węzłami;
5. Wyświetlenie informacji o działaniu programu;

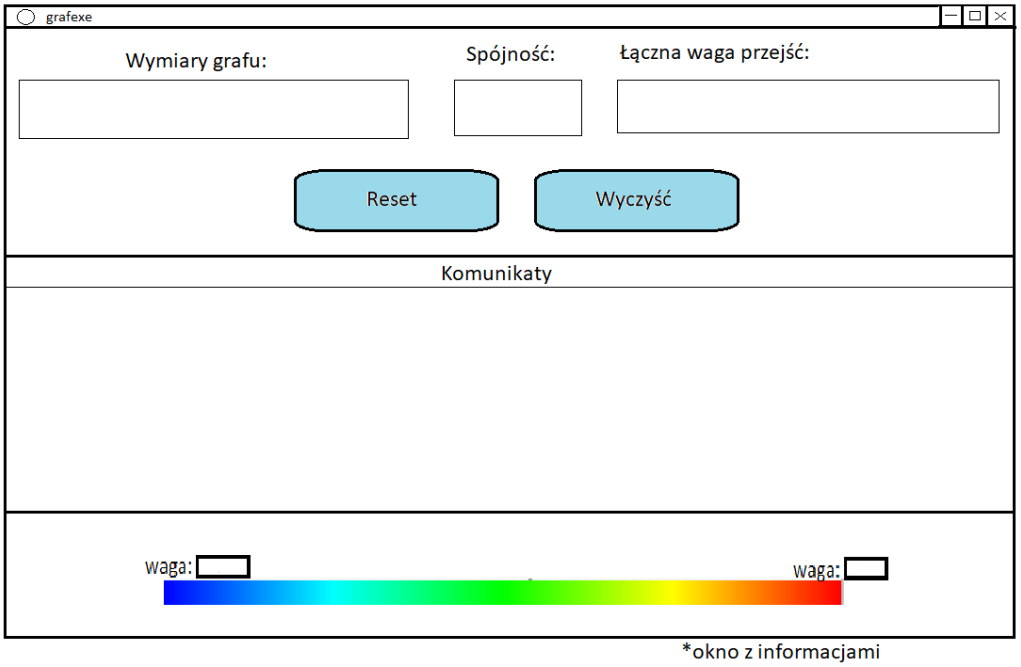
Wygląd aplikacji



*okno początkowe



*okno z grafem

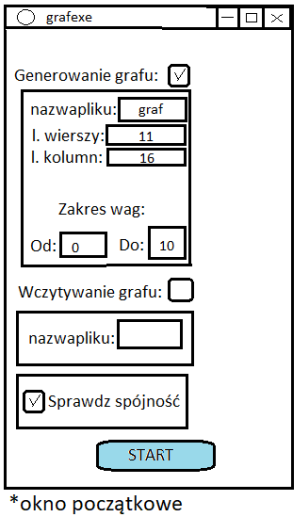


*okno z informacjami

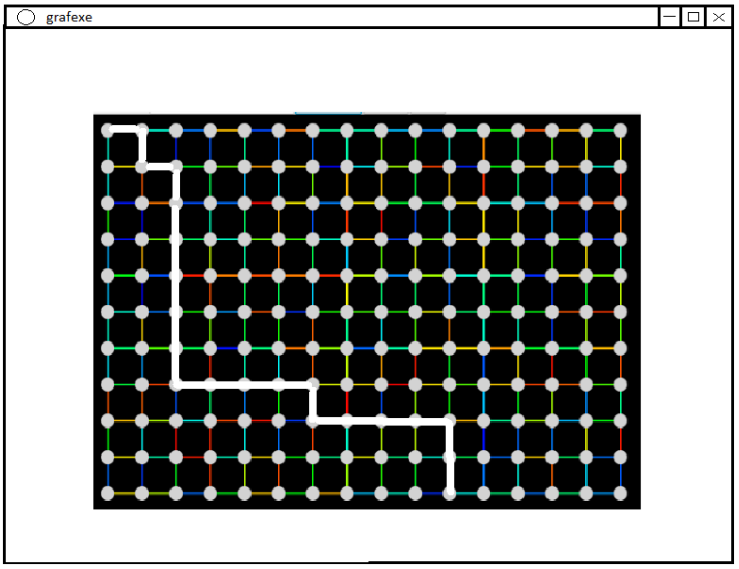
Wielkość i proporcje okien są poglądowe
Wygląd okien aplikacji bez uzupełnionych danych

Przykładowe uruchomienie programu

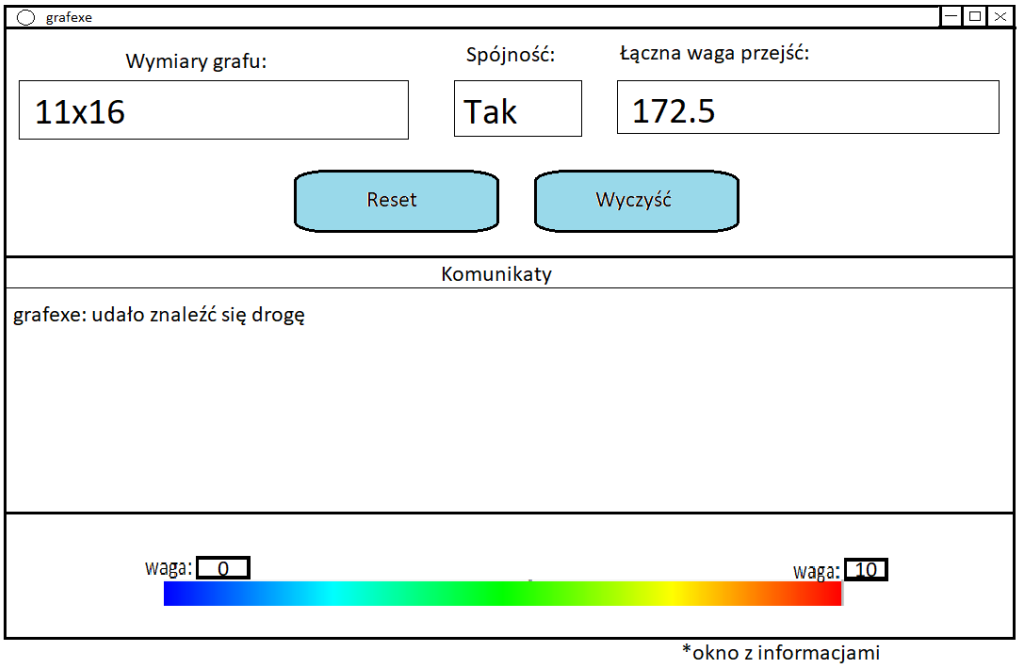
Wygenerowanie spójnego grafu o 11 wierszach, 16 kolumnach i wagach z zakresu od 0 do 10. W wyniku kliknięcia myszką wierzchołka 0, a następnie 170 program rysuje najkrótszą ścieżkę między tymi wierzchołkami, wypisuje wymiary grafu; łączną wagę przejść wynoszącą 172,5. Wyświetla informację, że graf jest spójny (“TAK”) oraz komunikat, że udało się znaleźć drogę między wybranymi wierzchołkami.



*okno początkowe



*okno z grafem



*okno z informacjami

Dane wejściowe

- **nazwapliku(generowanie)** - Nazwa generowanego przez program grafu, użytkownik nadaje ją jako jednoczłonowy wyraz, w którym występować mogą symbole z alfabetu łacińskiego, cyfry oraz symbole specjalne. Długość nazwy zawiera się w 64 znakach.
- **l.wierszy** - pole tekstowe służące do podania ilości wierszy w generowanym grafie. Liczba ta powinna być naturalna większa od jeden.
- **l.kolumn** - pole tekstowe służące do podania ilości kolumn w generowanym grafie. Liczba ta powinna być naturalna większa od jeden.
- **zakres wag** - poprzez pola tekstowe "od" oraz "do" podajemy kolejno początek oraz koniec przedziału z którego losowane będą wagi w generowanym grafie. Liczba "do" powinna być większa lub równa od liczby "od".
- **nazwapliku(wczytywanie)** -Nazwa wczytywanego przez program grafu, użytkownik nadaje ją jako jednoczłonowy wyraz, w którym występować mogą symbole z alfabetu łacińskiego, cyfry oraz symbole specjalne. Długość nazwy zawiera się w 64 znakach.
- **sprawdź spójność** - pole wyboru, które zaznaczamy bądź odznaczamy. Zaznaczone pozwoli sprawdzić spójność generowanego lub wczytanego grafu.
- **generowanie grafu** - pole wyboru, które zaznaczamy bądź odznaczamy. Zaznaczenie równoważne jest na zdecydowanie się na jedną z dwóch opcji które dostarcza program (między generowaniem i wczytywaniem) w tym przypadku jest to generowanie.
- **wczytywanie grafu** - pole wyboru, które zaznaczamy bądź odznaczamy. Zaznaczenie równoważne jest na zdecydowanie się na jedną z dwóch opcji które dostarcza program (między generowaniem i wczytywaniem) w tym przypadku jest to wczytanie grafu.

Zasada działania programu

Po uruchomieniu programu grafexe wyświetlone zostanie *okno początkowe, które odpowiada za sterowanie programem. Wybieramy tutaj jedną z dwóch opcji generowanie lub wczytywanie grafu a następnie wypełniamy odpowiednie rubryki. Po podaniu właściwych informacji i zaznaczeniu odpowiednich okienek oraz kliknięciu przycisku "Start" pojawią się dwa okno tj. *okno z grafem oraz *okno z informacjami. W oknie z grafem zilustrowany zostanie graf oraz połączenia między węzłami w odpowiednich kolorach odpowiadającym wagom. Użytkownik zaznaczyć będzie mógł odpowiednio początek oraz koniec szukanej ścieżki poprzez kliknięcie lewym przyciskiem myszy na interesujące go wierzchołki. Wówczas powstanie najkrótsza ścieżka zilustrowana na grafie (przykład w dołączonym zdjęciu). W oknie z informacjami zostaną wyświetlone informacje na temat prezentowanego grafu takie jak wymiary, łączna waga przejść, spójność oraz zilustrowany przedział wagowy. Ponadto program poinformuje nas o przebiegu działania programu w miejscu komunikatów. W przeciwnym przypadku, gdy podane zostaną błędne dane lub ich nieodpowiednia ilość wyświetlone zostanie okno z informacjami oraz odpowiednie informacje odnośnie błędów w miejscu komunikatów.

Przycisk "Reset" pozwala nam uruchomić program od początku i wprowadzić nowe dane.
Przycisk "Wyczyść" usuwa powstałe ścieżki.

Teoria

Algorytm Dijkstry - algorytm służący do wyznaczania najkrótszych ścieżek w grafie. Wyznacza on najkrótsze ścieżki z jednego wierzchołka (wierzchołka źródłowego) do pozostałych wierzchołków, przy założeniu, że wagi krawędzi grafu nie są ujemne.

Zasada działania: Tworzymy dwa zbiory wierzchołków Q i S. Zainicjowany zbiór Q zawiera wszystkie wierzchołki grafu, a zbiór S jest pusty. Dla wszystkich wierzchołków u grafu za wyjątkiem startowego v ustawiamy koszt dojścia d(u) na nieskończoność. Koszt dojścia d(v) zerujemy. Dodatkowo ustawiamy poprzednik p(u) każdego wierzchołka u grafu na niezdefiniowany (liczba ujemna). Poprzednie wierzchołki będą wyznaczały w kierunku odwrotnym najkrótsze ścieżki od wierzchołków u do wierzchołka startowego v. Dopóki zbiór Q zawiera wierzchołki, wykonujemy następujące czynności:

- Wybieramy ze zbioru Q wierzchołek u o najmniejszym koszcie dojścia d(u).
- Wybrany wierzchołek u przenosimy ze zbioru Q do zbioru S.
- Dla każdego sąsiada w wierzchołka u, który jest wciąż w zbiorze Q, sprawdzamy, czy
- d(w) > d(u) + waga krawędzi u–w.
- Jeśli tak, to należy wyznaczyć nowy koszt dojścia do wierzchołka w jako:
- d(w) ← d(u) + waga krawędzi u–w.
- Następnie wierzchołek u czynimy poprzednikiem w:
- p(w) ← u.

BFS - algorytm przeszukiwania grafów wszerz, wykorzystujący kolejkę FIFO (First in first out). Przechodzenie grafu rozpoczyna się od wybranego wierzchołka *s* i polega na sprawdzeniu wszystkich osiągalnych z niego wierzchołków. Wynikiem działania algorytmu jest drzewo przeszukiwania wszerz o korzeniu w *s*, zawierające wszystkie wierzchołki osiągalne z *s*. Do każdego z tych wierzchołków prowadzi dokładnie jedna ścieżka z *s*, która jest jednocześnie najkrótszą ścieżką w grafie wejściowym.

Źródła:

- https://pl.wikipedia.org/wiki/Przeszukiwanie_wszerz
- https://eduinf.waw.pl/inf/alg/001_search/0126.php
- https://eduinf.waw.pl/inf/alg/001_search/0138.php

Komunikaty oraz błędy

Wystąpienie większości błędów wiąże się z pojawieniem tylko okna z informacjami . Przedstawione komunikaty to schematy komunikatów, które będą wyświetlane w rubryce “komunikaty”, a w miejsce pogrubionych wyrażeń wstawiane będą odpowiednie zmienne. Komunikaty wyświetlane będą z polskimi znakami.

- Podanie liczby kolumn lub wierszy w postaci liczby większej od jeden i niecałkowitej lub w postaci liczby nie większej niż jeden:
  “*Wprowadzono niepoprawną liczbę kolumn lub wierszy.*”

- Podanie liczby określającej początek przedziału, z którego losowane są wagi przejść w grafie większej niż liczby określającej koniec tego przedziału:
  "*Podany przedział wag nie jest poprawny.* "

- Podanie nazwy pliku nieistniejącego:
  "*Nie udało się otworzyć pliku z grafem :* ***plik.***"

- Nieodpowiednie wprowadzenie danych (nieodpowiedni ich format np. litery w rubryce z liczbą kolumn):
  "*Sprawdź poprawność wprowadzanych danych*."

- Brak części lub wszystkich danych wejściowych:
  "*Nie wprowadzono wszystkich wymaganych danych.*"

- Nieodpowiednio zapisany graf w odczytywanym pliku:
  "*Zły format pliku z grafem.*"

- Wprowadzenie nieodpowiedniej liczby (mniejszej niż zero) określającej jeden z końców przedziału, z którego losowane zostaną wagi:
  "*Wprowadzono niewłaściwą liczbę wyznaczającą przedział, z którego losowane będą wagi.*"

Struktury danych

Wykorzystane zostaną tablice, kolejka priorytetowa w postaci drzewa binarnego chociażby w dijkstrze oraz lista sąsiedztwa przy generowaniu i przechowywaniu grafu. Utworzona zostaną klasy odpowiedzialne za poszczególne funkcjonalności programu:

- BFS - klasa odpowiedzialna w głównej mierze za wykonywanie algorytmu BFS. Posiada ona funkcję *BFS,* która zwraca tablicę zawierającą informacje o najkrótszej ścieżce między wybranymi wierzchołkami (pomija wagi przejść) oraz koszty przejść do każdego wierzchołka (jedno przejście równe jest wadze 1). Ponadto jest ona w stanie określić spójność grafu.

- dijkstra - klasa odpowiedzialna za wykonanie algorytmu Dijkstry oraz konstruowanie ścieżki między wybranymi wierzchołkami. Posiada ona funkcję *dijkstra,* która wykonuje wcześniej wspomniany algorytm i zwraca tablicę, która zawiera informacje o najkrótszej ścieżce między wybranymi wierzchołkami oraz koszty przejść do każdego wierzchołka. Ponadto zawiera funkcję showPath, która konstruuje ścieżkę między wybranymi wierzchołkami.

- generator - klasa odpowiedzialna za tworzenie grafu. Zawiera funkcję grafgen i zwraca wygenerowany graf jako tablicę.

- main - klasa sterująca i wywołujące odpowiednie funkcję w zależności od wyboru użytkownika. Zwraca ona informację o poprawności wykonania się programu.

- czytacz - klasa odpowiedzialna za odczytywanie grafu w odpowiednim formacie z pliku tekstowego. Posiada ona funkcję readFromFile, która z dostarczonego pliku odczytuje wymiary grafu oraz informację o przejściach i ich ewentualnych wagach, następnie zwracając te informacje.

- zapisywacz - klasa odpowiedzialna za zapisywanie grafu w odpowiednim formacje do pliku tekstowego. Posiada ona funkcję writeToFile, która tworzy plik o podanej nazwie oraz zapisuje wygenerowany graf do danego pliku. Funkcja nie zwraca żadnych informacji.

*każda z klas dodatkowo zgłasza adekwatny wyjątek w przypadku niepowodzeń.

Testy

W trakcie tworzenia programu tworzone będą przeprowadzane testy jednostkowe wyżej wymienionych funkcji z odpowiednim komentarzem. Wyniki testów końcowych GUI zostaną przedstawione w formie screenów z wynikiem działania programu i odpowiednim komentarzem o poprawności.