Spis treści

1 Opis ogólny

1.1 Nazwa programu

SUGP -The Shortest Undirected Graph Path (Poszukiwanie najkrótszej ścieżki w grafie nieskierowanych.)

1.2 Przeznaczenie dokumentu

Dokument stanowi sprawozdanie z projektu *"SUGP"*, z wyszczególnionym opisem aktualnego stanu projektu, zmian względem specyfikacji oraz wnioskami.

1.3 Cel programu

Celem programu jest znalezienie najkrótszej ścieżki w grafie nieskierowanym. Graf będzie wczytywany z pliku o ustalonej strukturze lub generowany na podstawie zadanych parametrów wejściowych. Generowany graf będzie zapisywany w postaci pliku.

Program powinien zostać zwizualizowany za pomocą interfejsu graficznego. Interfejs graficzny będzie spełniał następujące funkcjonalności: wizualizacja grafu, wybór za pomocą myszki węzłów, między którymi zostanie poprowadzona najkrótsza ścieżka, wyświetlenie najkrótszej ścieżki.

1.4 Scenariusz działania programu

Program po uruchomieniu wyświetla menu główne z możliwością wyboru trzech opcji :

1. Wyznacznanie ścieżki z pliku
   (a) Użytkownik wybiera opcje 1.
   (b) Użytkownik zostaje poproszony o podanie wierzchołka startowego i końcowego.
   (c) Program wyświetla najkrótszą znalezioną ścieżkę oraz jej długość.
2. Tworzenie grafu i wyznaczenie w nim ścieżki
   (a) Użytkownik wybiera opcję 2.
   (b) Użytkownik podaje liczbę wierszy i kolumn,następnie jest proszony o podanie zakresu zmienności wag dla generowanych krawędzi.
   (c) Użytkownik podaje wierzchołek startowy i końcowy dla znalezienia najkrótszej ścieżki.
   (d) Program tworzy graf i wyszukuje najkrótszą ścieżkę między wybranymy wierzchołkami.
   (e) Program zapisuje wygenerowany graf do pliku graph.txt.

(f) Program wyświetla ścieżkę oraz jej długość.

3. Zakończenie pracy programu

   (a) Użytkownik wybiera opcję 3, jeżeli nie chce dalej wchodzić w interakcje z programem.

   (b) Program kończy działanie.

Opis szczegółowy oraz przykładowe dane zostały opisane w p.3.

2 Budowa Programu

2.1 Opis klas

Main

Jedyna klasa, z którą w bezpośrednią interakcję wchodzi użytkownik. Odpowiada za sterowanie przekazywaniem informacji o wciśniętych klawiszach do kolejnych klas.

Metody klasy **Main**:

1. *public static void main(String[] args)* - Metoda główna programu. Od jej wywołania rozpoczyna się działanie programu.

Panel

Klasa odpowiedzialna za implementację oraz obsługę graficznego interfejsu użytkownika. Zaimplementowana została tutaj również obsługa zdarzeń związanych z interakcją użytkownika z wykorzystaniem myszy.

Metody klasy **Panel**:

1. *public static void draw(Graph g, int start, int finish)* - Metoda pozwalająca na wyrysowanie grafu na Panelu w celu wizualizacji najkrótszej drogi. Jako argumenty przyjmuje obiekt klasy Graph oraz węzły grafu - startowy oraz docelowy (końcowy).

Algorithms

Klasa zawiera implementacje algorytmów wykorzystanych w programie, czyli Algorytm BFS oraz Algorytm Dijkstry.

Metody klasy **Algorithms**:

1. *public boolean BFS(Graph graph, int start)* - Metoda pozwalająca na sprawdzenie spójności grafu. W przypadku, gdy graf jest spójny - zwraca *true*, w przeciwnym razie - *false*. Jako argumenty wejściowe przyjmuje obiekt klasy Graph oraz węzeł początkowy.

2. *public int[] dijkstra(Graph graph, int start, int finish)* - Metoda zawiera implementację algorytmu Dijkstry (poszukiwanie najkrótszej ścieżki w grafie między dwoma zadanymi wierzchołkami). Zwraca tablicę wierzchołków przez które należy przejść aby koszt drogi był najmniejszy. Jako argumenty wejściowe przyjmuje obiekt klasy Graph, numery węzłów - startowego i końcowego.

Graph

Przechowuje wszystkie informacje o grafie oraz odpowiada za jego wyświetlenie na ekranie, jego budowę i zapis do pliku.

Metody klasy **Graph**:

1. *public void print()* - Metoda pozwala na wypisanie wierzchołków.
2. *public Graph(int rows, int cols)* - konstruktor klasy Graph. Jako argumenty przyjmuje liczbę wierszy i kolumn macierzy sąsiedztwa grafu.
3. *public Graph(int rows, int cols, ArrayList<Node>[] field)* - konstruktor klasy Graph wywoływany w przypadku podania przez użytkownika zmiennej field.
4. *public void printGraphToFile(String filename)* - metoda pozwala na zapis grafu do pliku. Jako argument wejściowy przyjmuje ścieżkę do pliku (lub jego nazwę).

Generator

Generuje wagi dla krawędzi grafu oraz sam graf.

Metody klasy **Generator**:

1. *public double generateValue(double left, double right)* - Metoda pozwalająca na generowanie losowych wag krawędzi z przedziału podanego jako argumenty wejściowe (left oraz right).
2. *public Graph generateGraph(int rows, int cols, double left, double right)* - Metoda pozwalająca na generowanie grafu. Jako argumenty przyjmuje liczbę wierszy oraz kolumn (rows, cols) oraz zakres zmienności wag (left, right) generowanych z wykorzystniem funkcji generateValue() opisanej powyżej.

Node

Jednostka reprezentująca połączenie wierzchołka i krawędzi prowadzącej do niego.

Metody klasy **Node**:

1. *public Node()* - Konstruktor klasy Node.
2. *public Node(int destination, double weight)* - Konstruktor klasy Node wraz z arguentami - destination (numer wierzchołka do którego możemy trafić z bieżącego) oraz weight (dystans do bieżącego wierzchołka).

Reader

Odpowiada za prawidłowy odczyt grafu z pliku wejściowego. Zawiera również funkcję pozwalającą na oczyszczanie odczytywanych linii z niepożądanych znaków.

Metody klasy **Reader**:

1. *private String normaliseStr(String now)* - Metoda pozwala na oczyszczenie ciągu znaków z niepożądanych elementów. Jako argument wejściowy przyjmuje ciąg znaków now, a zwraca oczyszczony ciąg znaków.
2. *public Graph readGraphFromFile(String filename)* - Metoda pozwala na odczyt grafu z pliku. Jako argument przyjmuje ścieżkę do pliku, a zwraca obiekt typu Graph.

2.2 Wykorzystane algorytmy

W projekcie zostały wykorzystane Algorytm Przeszukiwania Grafu Wszerz (BFS) dla sprawdzenia spójności grafu oraz Algorytm Dijkstry do znalezienia najkrótszych ścieżek między wierzchołakmi grafu.

2.3 Dane wejściowe

Istnieją trzy możliwe warianty danych wejściowych:

1. Nazwa pliku(*.txt) zawierającego strukturę grafu, dwa parametry(int) określające wierzchołek startowy i końcowy.
2. Zestaw parametrów wejściowych na podstawie których zostanie wygenerowany graf o krawędziach ważonych w wybranym zakresie.
3. Zestaw parametrów wejściowych na podstawie których zostanie wygenerowany graf o krawędziach ważonych w zakresie <0,1>.

Ad. 1 W przypadku wczytywania grafu dane wejściowe zawarte będą w pliku tekstowym (*.txt), którego nazwę zostanie podana jako argument wywołania.

Struktura pliku:

- Pierwszy wiersz pliku zawierać będzie kolejno: liczbę kolumn i liczbę wierszy grafu.

- Kolejne wiersze pliku dotyczyły będą kolejno wierzchołków od 0 do n(int), gdzie n(int) to liczba wierzchołków pomniejszona o 1 (numeracja wierzchołków zaczyna się od 0). Każdy z wierszy zawierać będzie listę par (numer wierzchołka(int) :wagę krawędzi(double)), gdzie separatorem jest „:", a liczby zmiennoprzecinkowe wyrażające wagi zapisywane są przy pomocy znaku kropki (zamiast przecinka).

Przykładowo:

```
7 4
1 :0.8864916775696521  4 :0.2187532451857941
5 :0.2637754478952221  2 :0.6445273453144537  0 :0.4630166785185348
6 :0.8650384424149676  3 :0.42932761976709255  1 :0.6024952385895536
7 :0.5702072705027322  2 :0.86456124269257
8 :0.9452864187437506  0 :0.8961825862332892  5 :0.9299058855442358
1 :0.5956443807073741  9 :0.31509645530519625  6 :0.40326574227480094
10 :0.7910000224849713  7 :0.7017066711437372  2 :0.20056970253149548
6 :0.9338390704123928  3 :0.797053444490967  11 :0.7191822139832875
4 :0.7500681437013168  12 :0.5486221194511974  9 :0.25413610146892474
13 :0.8647843756083231  5 :0.8896910556803207  8 :0.4952122733888106
14 :0.5997502519024634  6 :0.5800735782304424  9 :0.7796297161425758
15 :0.3166804339669712  10 :0.14817882621967496  7 :0.8363991936747263
13 :0.5380334165340379  16 :0.8450927265651617  8 :0.5238810833905587
17 :0.5983997022381085  9 :0.7870744571266874  12 :0.738310558943156
10 :0.8801737147065481  15 :0.6153113201667844  18 :0.2663754517229303
19 :0.9069409600272764  11 :0.7381164412958352  14 :0.5723418590602954
20 :0.1541384547533948  17 :0.3985282545552262  12 :0.29468967639003735
21 :0.7576872377752496  13 :0.4858285745038984  16 :0.28762266137392745
17 :0.6628790185051667  22 :0.9203623808816617  14 :0.8394013782615275
18 :0.6976948178131532  15 :0.4893608558927002  23 :0.5604145612239925
24 :0.8901867253885717  21 :0.561967244435089  16 :0.35835658210649646
17 :0.8438726714274797  20 :0.3311114339467634  25 :0.7968809594947989
21 :0.6354858042070723  23 :0.33441278736675584  18 :0.43027465583738667
27 :0.8914256412658524  22 :0.8708278171237049  19 :0.4478162295166256
20 :0.35178269705930043  25 :0.2054048551310126
21 :0.6830700124292063  24 :0.3148089827888376  26 :0.5449034876557145
27 :0.2104213229517653  22 :0.8159939689806697  25 :0.4989269533310492
26 :0.44272335750313074  23 :0.4353604625664018
```

Ad. 3 W przypadku generowania grafu na podstawie parametrów wejściowych powinny zostać podane następujące dane w odpowiedniej kolejności:

- Liczba wierszy grafu(int)
- Liczbę kolumn grafu(int)
- Zakres zmienności losowanych wag – 2 liczby(double) rzeczywiste nieujemne w porządku rosnącym (wartości będą losowane w zakresie od pierwszej liczby do drugiej)

Jeżeli nie zostanie podany zakres zmienności losowanych wag przyjęty zostanie przedział domyślny - <0,1>.

Przykładowo:

- 7 4 0 2 – podanie takich argumentów wejściowych oznacza wygenerowanie grafu o 7 wierszach, 4 kolumnach i wylosowanych wagach w przedziale <0,2>
- 2 3 – podanie takich argumentów wejściowych oznacza wygenerowanie grafu o 2 wierszach, 3 kolumnach oraz domyślnym przedziale wartości <0,1>

Dane powinny zostać podane w odpowiednim formacie. Żeby program był w stanie znaleźć rozwiązanie należy podać liczby w zakresie:

- Liczbę wierzchołków labiryntu (int) -liczba całkowita od 2 do 10000
- Liczbę krawędzi(int) - liczba całkowita od 1 do 10000
- Lewa granica zakresu(double) (w przypadku losowania wag krawędzi) - od 0 do 10000
- Prawa granica zakresu(double) (w przypadku losowania wag krawędzi) - od 0 do 10000; Przy tym lewa granica >= prawa granica

2.4 Dane wyjściowe

1. Zapis grafu w przypadku jego generacji

   Dane wyjściowe zapisywane są w takim samym formacie jak dane wejściowe w przypadku wczytywania grafu – plik tekstowy (graph.txt).

   Struktura pliku:

   - Pierwszy wiersz pliku zawiera kolejno: liczbę kolumn(int) i liczbę wierszy(int) grafu.
   - Kolejne wiersze pliku dotyczą kolejno wierzchołków od 0 do n(int), gdzie n(int) to liczba wierzchołków pomniejszona o 1 (numeracja wierzchołków zaczyna się od 0). Każdy z wierszy zawiera listę par (numer wierzchołka(int) :wagę krawędzi(double), gdzie separatorem jest „:”, a liczby zmiennoprzecinkowe wyrażające wagi zapisywane są przy pomocy znaku kropki (zamiast przecinka).

Przykładowo:

```
2 3
0 :0.2637754478952221  1 :0.6445273453144537  3 :0.4630166785185348
0 :0.8650384424149676  2 :0.42932761976709255  4 :0.6024952385895536
1 :0.9452864187437506  5 :0.8961825862332892
0 :0.5956443807073741  4 :0.31509645530519625
3 :0.7910000224849713  5 :0.7017066711437372  1 :0.20056970253149548
4 :0.9338390704123928  1 :0.797053444490967  2 :0.7191822139832875
```

2. Wynik działania programu - znalezienie najkrótszej ścieżki.

   Dane wyjściowe z najkrótszą ścieżką między wybranymi wierzchołkami wyświetlane są na ekranie

2.5 Komunikaty o błędach

Program obsługuje wiele rodzajów błędów - testy zostaną przeprowadzone poprzez uruchomienie programu z odpowiednio przygotowanymi, niepoprawnymi argumentami wywołania i formatowaniem pliku wejściowego, tak aby otrzymać każdy z komunikatów błędów z osobna.

Program powinien wyświetlić następujące komunikaty w razie wystąpienia problemów związanych z działaniem programu:

- Cannot open file – Jeżeli program nie jest w stanie otworzyć pliku, nie może znaleźć pliku o podanej nazwie albo plik nie istnieje.
- Wrong data format in file - Jeśli program nie może poprawnie odczytać danych z pliku wejściowego, podanego przez użytkownika, dane zapisane w złym formacie lub podane są niepoprawne wartości.
- Invalid program invocation arguments - W przypadku, gdy zostały podane złe argumenty lub ich liczba nie jest odpowiednia.
  - Invalid program invocation arguments. expected 2,3 or 4 arguments, but found - Jeżeli podano błędną ilość argumentów.
  - Invalid program invocation arguments. expected only numbers and dots, but founded - Jeżeli zostały podane niepoprawne argumenty.
- Inconsistent graph - Po sprawdzeniu spójności grafu powinien się wyświetlić ten komunikat, jeżeli podany albo wygenerowany graf jest niespójny, wtedy program nie jest w stanie znaleźć najkrótszą ścieżkę.

3 Przykładowe testy

Interfejs użytkownika - aby wybrać opcję należy wpisać odpowiedni numer z instrukcji użytkownika.

```
run:
*********************************
          Program SUGP
Autorzy:
Lidia Lachman, Daria Vasilchyk
*********************************

Wybierz opcje z posrod podanych:
1 - Wyznaczanie sciezki z pliku
2 - Tworzenie grafu i wyznaczenie w nim sciezki
3 - Wyjscie z programu

Wybierz opcje:
```

Rysunek 1: Interfejs użytkownika

Testowanie opcji nr 1 - Wyznaczanie najkrótszej ścieżki grafu wczytanego z pliku graph.txt między punktami podanymi przez użytkownika.

```
3 4
1 :0.9798 4 :0.3136
0 :0.9798 2 :0.1793 5 :0.6135
1 :0.1793 3 :0.0232 6 :0.3246
2 :0.0232 7 :0.5909
5 :0.5838 0 :0.3136 8 :0.3623
4 :0.5838 6 :0.497 1 :0.6135 9 :0.1909
5 :0.497 7 :0.2064 2 :0.3246 10 :0.8596
6 :0.2064 3 :0.5909 11 :0.3361
9 :0.3811 4 :0.3623
8 :0.3811 10 :0.7627 5 :0.1909
9 :0.7627 11 :0.8214 6 :0.8596
10 :0.8214 7 :0.3361
```

Rysunek 2: Przykładowy plik tekstowy graph.txt

Aby wybrać opcję nr 1, użytkownik powinien wpisać 1 i zatwierdzić klikając *Enter*. Następnie użytkownik powinien działać według poleceń wyświetlanych przez program. Najpierw program poprosi o podanie wierzchołka startowego i końcowego do wyznaczenia najkrótszej ścieżki. Później program wyświetli najkrótszą ścieżke - punkty i ich wagi, a następnie jej długość.

```
Wybierz opcje:  1
Opcja 1
Podaj wierzcholek startowy:  0
Podaj wierzcholek koncowy:  3



wierzcholek: 0
waga: 1.1823
wierzcholek: 1
waga: 0.20249999999999999
wierzcholek: 2
waga: 0.0232
wierzcholek: 3
waga: 0.0

Dlugosc najkrotszej sciezki: 1.408

Wybierz opcje:
```

Rysunek 3: Opcja 1

Testowanie opcji nr 2 - Tworzenie grafu z zadanymi przez uzytkownika parametrami oraz wyznaczenie dla niego najkrótszej ścieżki między podanymi punktami.

Aby wybrać opcję nr 2, użytkownik powinien wpisać 2 i zatwierdzić klikając *Enter*. Następnie użytkownik powinien działać według poleceń wyświetlanych przez program. Najpierw program poprosi o podanie liczby wierszy i kolumn oraz zakresu zmienności wag, aby mógł wygenerować graf. Potem program poprosi o podanie wierzchołka startowego i końcowego do wyznaczenia najkrótszej ścieżki. Później program wyświetli najkrótszą ścieżke - punkty i ich wagi, a następnie jej długość.

```
Wybierz opcje:  2
Opcja 2
Podaj liczbe wierszy:  8
Podaj liczbe kolumn:  12
Podaj zakres zmiennosci wag:  0 10
Gerneruje graf..
Wygenerowano pomyslnie!

Podaj wierzcholek startowy:  2
Podaj wierzcholek koncowy:  15



wierzcholek: 2
waga: 7.9066
wierzcholek: 3
waga: 7.2674
wierzcholek: 15
waga: 0.0

Dlugosc najkrotszej sciezki: 15.174

Wybierz opcje:
```

Rysunek 4: Opcja 2

Testowanie opcji nr 3 - Zakończenie pracy programu.

```
Wybierz opcje:  3
BUILD SUCCESSFUL (total time: 1 minute 56 seconds)
```

Rysunek 5: Opcja 3

4 Zmiany względem specyfikacji

Założeniem projektu od samego początku powstawania było zachowanie ścisłości między pisanym programem, a specyfikacją funkcjonalna i implementacyjną. Proces powstawania elementów projektu spowodował brak interfejsu graficznego.

5 Podsumowanie projektu

Projekt *"SUGP"* został zrobiony zgodnie z częścią wymagań, a dokładniej zostały zaimplementowana możliwość wygenerowania grafu na podstawie danych wejściowych, zapis i odczyt grafu z pliku o ustalonym formacie, sprawdzenie grafu na spójność oraz najważniejsze - znalezenie najkrótszej ścieżki w grafie między wybranymi wierzchołkami. W ramach niego powstał działający program *"SUGP"*, specyfikacja funkcjonalna, specyfikacja implementacyjną oraz sprawozdanie końcowe. Przygotowane rozwiązanie posiada prostą obsługę,posiada try różne możliwości uruchomienia w zalezności od oczekiwanych danych wyjściowych. Program *"SUGP"* został przetestowany pod kątem różnych błędów i nietypowych plików wejściowych. W konsekwencji jego działanie powinno być w znaczącej liczbie przypadków zgodne z oczekiwaniami.

Bibliografia

[1] `https://eduinf.waw.pl/inf/alg/001_search/0126.php`

[2] `https://eduinf.waw.pl/inf/alg/001_search/0138.php`

[3] `https://pl.wikipedia.org/wiki/Przeszukiwanie_wszerz`