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 projektu

Celem projektu było napisanie programu, który:

- Potrafi wygenerować graf o zadanej liczbie kolumn i wierszy węzłów i wagach krawędzi losowanych w zadanym zakresie wartości.
- Potrafi zapisać taki graf do pliku o ustalonym formacie.
- Potrafi przeczytać z plku o ustalonym formacie taki graf.
- Potrafi sprawdzić, czy dany graf jest spójny (algorytm przeszukiwanie grafu wszerz - BFS).
- Potrafi znaleźć w tym grafie najkrótsze ścieżki pomiędzy wybranymi parami węzłów, wykorzystując Algorytm Dijkstry.

1.4 Scenariusz działania programu

Program na podstawie podanych argumentów wejściowych ustali czy należy generować graf czy wczytać go z pliku tekstowego.

- Jeżeli dostaje 2 liczby (int) reprezentujące liczbę wierszy i liczbę kolumn lub 4 liczby (dwie pierwsze(int) - liczba wierszy i liczba kolumn, dwie drugie(double) - zakres wartości do wygenerowania wag krawędzi), to
    1. Program generuje graf ważony o wagach w zakresie podanym przez użytkownika, lub, jeżeli zakres nie zostanie podany, program generuje wagi w domyślnym zakresie <0,1>.
    2. Program się pyta o wierzchołek startowy i końcowy do znajdowania najkrótszej ścieżki między nimi:

       ```
       Input start and destinition vertices:
       ```

    3. Użytkownik wprowadza numery wierzchołków okreslające początek i konieć ścieżki.
    4. Jeżeli wszystkie dane wejściowe zostały podane poprawnie, to

  - Program zapisuje wygenerowany graf do pliku graph.txt.
  - Program sprawdza spójność grafu, w przypadku grafu spójnego kontynuje działanie, a w przypadku grafu niespójnego wyświetla komunikat o błędzie.
  - Program znajduje najkrótszą ścieżkę i wyświetla ją:

    `the minimum distance between vertices x1 and x2 is - distance`

    gdzie x1 - wierzchołek startowy, x2 - wierzchołek końcowy, distance - najkrótsza ścieżka.

5. W przypadku podania niepoprawnych danych program zwraca komunikat o błędzie.

- Jeżeli program dostaje plik tekstowy (.txt) i 2 liczby(int) określające wierzchołek początkowy i wierzchołek końcowy do znajdowania ścieżki - przechodzi do następnego kroku.

  1. Program sprawdza spójność grafu, w przypadku grafu spójnego kontynuje działanie, a w przypadku grafu niespójnego wyświetla komunikat o błędzie.
  2. Program wyszukuje najkrótsze ścieżki pomiędzy wybranymi punktami i wyświetla ją z takim samym komunikatem jak powyżej:

     `the minimum distance between vertices x1 and x2 is - distance`

2 Budowa programu

2.1 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.

2.2 Dane wejściowe

Istnieją trzy możliwe warianty danych wejściowych:

1. Nazwa pliku(*.txt) zawierającego strukturę grafu, dwa parametry(int) określające wierzchołek startowy i końcowy.

   Wywołanie:

   `./out -nazwa_pliku(*.txt) -nr_wierzcholka_start(int) -nr_wierczholka_koniec(int)`

2. Zestaw parametrów wejściowych na podstawie których zostanie wygenerowany graf o krawędziach ważonych w wybranym zakresie.

   Wywołanie:

```
./out -liczba_wierszy(int) -liczba_kolumn(int) -lewa_granica_zakresu(double)
        - prawa_granica_zakresu(double)
```

3. Zestaw parametrów wejściowych na podstawie których zostanie wygenerowany graf o krawędziach ważonych w zakresie <0,1>.

   Wywołanie:

   ```
   ./out -liczba_wierszy(int) -liczba_kolumn(int)
   ```

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

2.3 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

2.4 Struktura plików projektu

Program zawiera cztery moduły: *buffer*, *graph*, *node*, *queue* oraz plik *main.c*, zawierający główną funkcję main całego programu oraz. Każdy z modułów składa się z dwóch plików: .h oraz .c, o odpowiedniej nazwie. W każdym z modułów plik .h zawiera prototypy funkcji, ewentualne definicje struktur i stałe. Większość obsługi błędów znajduje się bezpośrednio w funkcji main, z tego względu, że gdyby zostały przeniesione do zewnętrznych funkcji, niezbędne byłoby przekazywanie dużej ilości parametrów – byłoby to niezbyt czytelne. Drugą możliwością byłoby zadeklarowanie owych zmiennych jako globalne, jednak duża ilość zmiennych globalnych nie jest dobrą praktyką.

- Moduł buffer zawiera funkcje odpowiadającą za wczytanie zawartości pliku tekstowego oraz za zapis wygenerowanego grafu do pliku tekstowego. Zawiera również definicje kodów błędów oraz pomocnicze struktury, pozwalające na wyświetlanie komunikatów o błędach.
- Moduł graph odpowiada za działania prowadzone bezpośrednio ma grafie, zawiera funkcje do generacji grafu i wag krawędzi, realizuje algorytmy BFS i Dijkstry - funkcje sprawdzającą spójność i wyszukującą najkrótszą ścieżkę.
- Moduł node zawiera strukturę do przechowywania jednostki grafu, pod którą rozumiemy wierzchołek wraz ze wskaźnikiem na następny oraz wagą przylegającej krawędzi.
- Moduł queue dotyczy struktury kolejki i funkcji wykonywanych na niej operacji, a dokładnie dodawania i usuwania elementów.

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.

2.6 Kompilacja

Dla kompilacji został użyty kompilator gcc version 9.2.0 (GCC). Standart języka C w programie - C11. Szczegóły kompilacji zostały zawarte w skrypcie makefile.

3 Przykładowe testy

- Niepoprawne dane

  1. Plik wejściowy nie istnieje:

```
.\out.exe graaaph.txt 4 2
Cannot open file
```

  2. Niepoprawne dane w pliku wejściowym:

```
./out.exe graph2.txt 4 50
Wrong data format in file
```

  3. Niepoprawne argumenty wywołania:

```
./out.exe 3 d
Invalid program invocation arguments. expected only numbers and dots,
but founded - d
```

```
./out.exe d 3
Invalid program invocation arguments. expected only numbers and dots,
but founded - d
```

  4. Zła ilość argumentów wywołania:

```
./out.exe 1 2 3 4 5
Invalid program invocation arguments. expected 2,3 or 4 arguments, but found 5
```

- Poprawne działanie programu

  1. Z plikiem wejściowym

```
./out.exe graph.txt 2 86
the minimum distance between vertices 2 and 86 is - 236.240271
```

  2. Generowanie grafu z zadanym zakresem generowania wag krawędzi, znalezienie ścieżki miedzy wybranymi węzłami:

```
./out.exe 100 100 1 1000
Input start and destinition vertices: 36 725
the minimum distance between vertices 36 and 725 is - 5123.982360
```

  3. Generowanie grafu z domyślnym zakresem generowania wag krawędzi, znalezienie ścieżki miedzy wybranymi węzłami:

```
./out.exe 100 100
Input start and destinition vertices: 56 701
the minimum distance between vertices 56 and 701 is - 20.301248
```

4. Generowanie grafu dla dużej liczby wierszy i kolumn z domyślnym zakresem generowania wag krawędzi, znalezienie ścieżki miedzy wybranymi węzłami:

   ```
   ./out.exe 10000 10000
   Input start and destinition vertices: 15 1028
   ...
   ```

   Dla dużych wartości działanie programu zajmuje znacznie więcej czasu

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 zweryfikował dokładnie wszystkie założone idee i spowodował zmianę następujących punktów w poszczególnych dokumentach:

1. Scenariusz działania:

   - Żeby usprawnić działanie programu i uniknąć konieczności podwójnego uruchomiania programu został zmieniony scenariusz programu w przypadku generowania pliku, teraz program również wyszukuje ścieżkę pomiędzy wybranymi punktami, które są wyznaczane poprzez interakcję z użytkownikiem.

2. Dane wyjściowe:

   - W celu ułatwienia obsługi programu została dodana funkcjonalność odpowiadająca za wyświetlenie komunikatu z zapytaniem do użytkownika o węzeł startowy i końcowy dla znalezienia ścieżki :

     `Input start and destinition vertices:`

   - Najkrótsza ścieżka nie jest zapisywana do pliku, a jest wyświetlana jej długość na ekranie. Został również dodany odpowiedni komunikat:

     `the minimum distance between vertices x1 and x2 is - distance`

     Gdzie x1- wierzchołek startowy, x2 - wierzchołek końcowy, distance - długość najkrótszej ścieżki między nimi.

3. Komunikaty błędów:

   - Nastąpiło rozszerzenie definicji komunikatu błędu dotyczącego błędnych argumentów wywołania. Dodano dwa komunikaty :

     ```
     Invalid program invocation arguments. expected 2,3 or 4 arguments,
     but found

     Invalid program invocation arguments. expected only numbers and dots,
     but founded
     ```

4. Sposób prowadzenia zmian i system kontroli wersji.

   - Dokumentacja była tworzona i edytowana przez online edytor tekstu Overleaf.
   - Zmiany w kodzie były wprowadzane przy uzgodnieniu z partnerem lub projekt był edytowany wspólnie za pośrednictwem komunikacji face-to-face, dlatego prowadzenie projektu w systemie kontroli wersji GIT nie było systematyczne. Zamieszczono jedynie finalną wersję: link do repozytorium GitHub.

5 Podsumowanie projektu

Projekt *"SUGP"* został w całości przygotowany. W ramach niego powstał działający program *"SUGP"*, makefile, 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.