Specyfikacja funkcjonalna projektu związanego z algorytmami operującymi na grafach

Adrian Nowosielski
Cezary Skorupski

17.03.2022

Streszczenie

Dokument specyfikacji funkcjonalnej stanowi opis szczegółowy projektu **Grafy** oraz związany z nim zakres prac, funkcje programu, interakcje zachodzące między systemem a użytkownikiem i instrukcje użytkowania.

Spis treści

1 Opis ogólny

1.1 Nazwa programu

Zespół projektowy ustalił, że nazwa wykonywanego projektu będzie nosić nazwę: **"Program operujący na grafach".**

1.2 Poruszany problem

Projekt jest przygotowywany na zajęcia JIMP2 na Politechnice Warszawskiej. Program ma służyć do analizy grafów, które mają specjalną postać-można przedstawić je w postaci siatki, takiej jak papier w kratkę. Skrzyżowania kratek to węzły, a linie pionowe i poziome to krawędzie. Dodatkowo analizowane grafy są ważone, czyli każda krawędź w grafie ma przyporządkowaną jakąś wagę z określonego zakresu.
Program ma mieć następujące funkcjonalności:

- 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.

1.3 Użytkownik docelowy

Program jest dedykowany dla studentów Politechniki Warszawskiej oraz wszystkich innych osób, które są zainteresowane tematyką grafów.

2 Opis funkcjonalności

2.1 Jak korzystać z programu?

W celu skorzystania z programu potrzebny jest sprawny komputer z zainstalowanym systemem typu Linux z kompilatorem języka C. Po pobraniu programu, należy go wypakować, oraz w konsoli, będąc w katalogu programu wpisać make. Po wykonaniu następujących czynności w katalogu programu powinien stworzyć się plik ./grafy.

2.2 Argumenty wywołania programu

Niezbędne do uruchomienia programu:

- W przypadku gdy nie mamy przygotowanego pliku wejściowego z grafem:
  - liczba wierszy- liczba wierszy w generowanym grafie
  - liczba kolumn- liczba kolumn w generowanym grafie
  - dolny przedział wagi- dolny przedział dla losowanych wag krawędzi
  - górny przedział wagi- górny przedział dla losowanych wag krawędzi
- W przypadku, gdy mamy przygotowany plik wejściowy z grafem:
  - plik wejściowy z grafem- plik z grafem o ustalonym formacie

Flagi opcjonalne:

- –**save-**program wywołany z tą flagą zapisze do pliku wejściowego wygenerowany graf przez program. Wygenerowany plik ma określony format grafu(dzięki temu możemy użyć wygenerowanego pliku do uruchomienia programu). Nazwę pliku, do którego ma się zapisać graf podajemy bezpośrednio po fladze –save.
- –**generate-**po uwzględnieniu flagi, użytkownik daje znak programowi, że program sam musi wygenerować plik z grafem.
- –**read-**po uwzględnieniu flagi, użytkownik daje znak programowi, że użytkownik sam ma wygenerowany plik z grafem. Bezpośrednio po fladze –read należy podać plik z grafem.
- –**bfs**-program wywołany z tą flagą musi mieć podany graf do sprawdzenia(musi być zawarta flaga –generate lub –read). W przypadku użycia flagi bfs, użytkownik daje znak programowi, że chce sprawdziyć czy graf, który użytkownik chce sprawdzić jest spójny.
- –**dijkstra**-program wywołany z tą flagą musi mieć podany graf, na którym będzie operować. W przypadku użycia flagi dijkstra, użytkownik daje znak programowi, że chce znaleźć najkrótszą ścieżkę pomiędzy dwoma węzłami.

2.3 Uruchomienie programu

Program został przyogotowany do uruchomienia w systemie Linux. Zalecana jest najnowsza dystrybucja: Ubuntu 20.04.
Uruchomienie programu odbywa się poprzez wpisanie niezbędnych argumentów wywołania programu w terminalu systemu Linux:

- W przypadku, gdy mamy przygotowany plik wejściowy z przygotowanym grafem:
  ./grafy -read [nazwa pliku wejsciowego z grafem] [-bfs(opcjonalnie)] [-dijkstra(opcjonalnie)] [-save(opcjonalnie)]
- W przypadku, gdy nie mamy przygotowanego pliku z grafem:
  ./grafy -generate [liczba wierszy] [liczba kolumn] [przedział dolny dla wagi krawędzi] [przedział górny dla wag krawędzi] [-bfs(opcjonalnie)] [-dijkstra(opcjonalnie)] [-save(opcjonalnie)]

2.4 Możliwości programu

W programie są przewidziane następujące możliwości:

- Sprawdzenie czy podany graf jest spójny, wykorzystując algorytm Breadth First Search.
- Znalezienie najkrótszej ścieżki pomiędzy wybranymi parami węzłów, wykorzystując algorytm Dijkstry.
- Przeczytanie pliku z przygotowywanym przez użytkownika grafem.
- Wygenerowanie przykładowego grafu przez program.

3 Format danych i przygotowanie plików

3.1 Pojęcia i pola formularza:

- **Graf-**podstawowy obiekt rozważań teorii grafów, struktura matematyczna służąca do przedstawiania i badania relacji między obiektami. W uproszczeniu graf to zbiór wierzchołków, które mogą być połączone krawędziami w taki sposób, że każda krawędź kończy się i zaczyna w którymś z wierzchołków.
- **Węzły-**inna nazwa na wierzchołki w grafie.
- **Krawędź-**krawędź grafu jest to para wyróżnionych wierzchołków grafu, czyli takich, które są ze sobą połączone. W reprezentacji graficznej jest to linia łącząca te wierzchołki.

3.2 Struktura katalogów:

Struktura katalogów w programie nie jest skomplikowana. Po wejściu do katalogu głownego programu znajdziemy w nim 3 katalogi:

- **Dokumenty-**katalog, w którym przechowywane są dokumenty projektowe(specyfikacja funkcjonalna,implementacyjna,sprawozdanie).
- **Kod-**katalog, w którym umieszczony jest cały kod potrzebny do uruchomienia programu.
- **Przyklad-**przykładowe pliki wejściowe i parametry uruchomienia programu.

3.3 Przechowywanie danych w programie:

Podstawowym sposobem przechowywania danych w programie jest lista liniowa. Graf jest przechowywany w tablicy list liniowych. Do przeprowadzenia algorytmu BFS wykorzystywana jest również lista liniowa. W programie wykorzystywana jest również kolejka priorytetowa.

3.4 Dane wejściowe

Program w celu wykonania potrzebuje plik z informacjami o grafie. Plik wejściowy z grafem powinien zawierać linie sformatowane następująco:
1:wiersze kolumny
2:numer wierzchołka :waga krawędzi numer wierzchołka :waga krawędzi.
3:n:Następne wszystkie dalsze linie powinny wyglądać tak samo, jak 2 przedstawiona linia.
Gdzie:

- **wiersze-**liczba całkowita wierszy w grafie
- **kolumny**-liczba całkowita kolumn w grafie
- **numer wierzchołka**-numer wierzchołka, do którego prowadzi krawędź
- **waga krawędzi**-liczba zmiennoprzecinkowa symbolizująca wagę połączenia

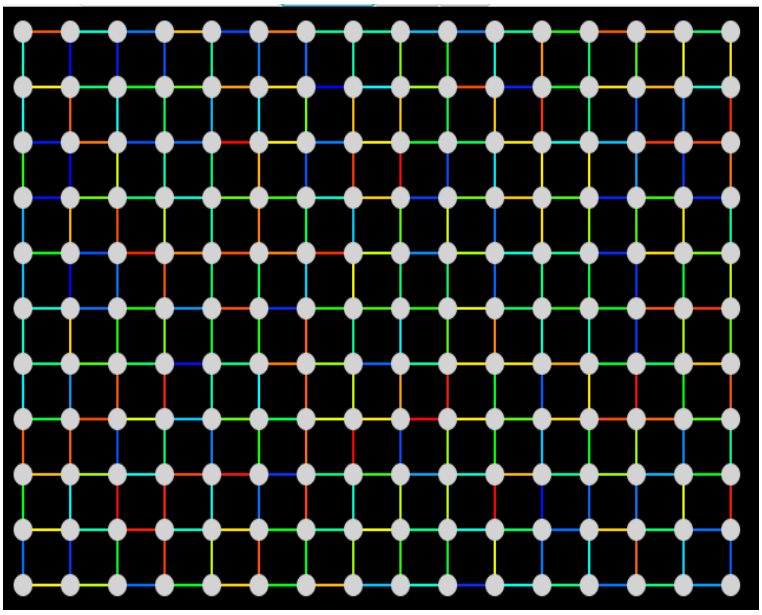
Przykład:
Chcąc wszytać graf w kratkę z dwoma kolumnami oraz dwoma wierszami, plik wejściowy powinien wyglądać następująco:
1: 2 2
2: 1 :0.5956998521442058 2 :0.9274149876681226
3: 0 :0.0636977381369554 3 :0.5678848678096593
4: 4 :0.4409732769434216 3 :0.4378655880865946
5: 1 :0.2069389713960416 2 :0.0232922532704157
Numery wierzchołków, z których wychodzą krawędzie reprezentują numery linii w pliku. Czyli w przykładzie zerowy wierzchołek łączy się z pierwszym oraz drugim. Analogicznie zerowy wierzchołek łączy się z zerowym i pierwszym itd.

3.5 Dane wyjściowe

Wynik działania programu jest zależny od preferencji użytkownika. W trybie grafu losowanego, użytkownik, jako wynik otrzymuje graf, który może zapisać do pliku i na którym może również sprawdzić działanie algorytmów Dijkstry oraz BFS. Wynik działania algorytmów zależnie od wybranego algorytmu daje wyniki w terminalu:

- W przypadku algorytmu BFS program daje odpowiedź w terminalu na pytanie czy podany graf jest spójny.
- W przypadku algorytmu Dijkstry znajduje najktótsze ścieżki między wybranymi przez użytkownika parami węzłów.

Program daje również możliwość czytania własnego grafu w kratkę z przygotowanego pliku. Wyniki wyjściowe algorytmów BFS oraz Dijkstry są analogiczne do trybu losowanego grafu.
Przykładowa wizualizacja wygenerowanego grafu-kratki, w którym wszystkie węzły są ze sobą połączone:



Rysunek 1: Przykładowy graf

4 Scenariusz działania programu

4.1 Scenariusz ogólny

- Tryb podstawowy:
  Wywołanie:
  /.grafy –generate [liczba kolumn] [liczba wierszy] [przedział dolny wag krawędzi] [przedział górny wag krawędzi] –save
  Wynik: Wygenerowanie grafu w kratkę oraz zapisanie wygenerowanego grafu do pliku txt.

- Tryb BFS:
  Wywołanie:
  ./grafy –read [nazwa pliku tekstowego] –bfs
  Wynik:
  Wynikiem programu jest informacja o tym czy podany przez użytkownika graf jest spójny.

- Tryb Dijkstra:
  Wywołanie:
  ./grafy –read [nazwa pliku tekstowego] –dijkstra [wierzchołek A] [wierzchołek B]
  Wynik:
  Wynikiem programu jest informacja o długości najkrótszej ścieżki pomiędzy wybranami przez użytkownika węzłami.

- Tryb ogólny: Oczywiście tryby można ze sobą łączyć. Program umożliwia wygenerowanie grafu oraz sprawdzenie tego, czy wygenerowany graf jest spójny oraz wyświetlenie najkrótszej drogi pomiędzy węzłam. Nie uwzględniamy tego jednak w poszczególnych trybach, bo generowanie grafu i sprawdzanie informacji o długości najkrótszych ścieżki pomiędzy węzłami bez znajomości grafu nie ma sensu.

4.2 Rozszerzony scenariusz działania programu:

4.2.1 Tryb podstawowy:

Możliwe wyniki kończące działanie programu:

Komunikat	Przyczyna
INCORRECT_NUMBER_OF_ARG	Niepoprawna ilość argumentów przy uruchumieniu programu.
UNKNOWN_FLAG	Nieznana flaga.
INPUT_ERR	Program nie może otworzyć pliku z przygotowanym grafem.
INPUT_WK_ERR	Podano ujemne lub zerowe wartości wierszy lub kolumn.
INPUT_FORMAT_ERR	Plik wejściowy jest źle sformatowany.
INPUT_NOT_INT	Liczby wierzchołków w pliku nie są liczbami całkowitymi
WEIGHT_ERR	Podano niepoprawny przedział wag krawędzi.
NO_OUT	Program wywołano z flagą -save, ale nie podano pliku wyjściowego.

Możliwe wyniki kontynujące działanie programu:

Komunikat	Przyczyna
FILENAME_TAKEN	Podany plik tekstowy jest zajęty. Program pyta się o chęć nadpisania pliku.

4.2.2 Tryb BFS:

Możliwe wyniki kończące działanie programu:
Tryb BFS wymaga mieć w pełni wczytany/wygenerowany graf. W momencie gdy program nie dostanie grafu, uwzględniając w programie flagę –bfs otrzymamy błąd INPUT_FORMAT lub inne błędy związane ze złym wczytaniem grafu.

4.2.3 Tryb Dijkstra:

Możliwe wyniki kończące działanie programu:

Komunikat	Przyczyna
INCORRECT_NUMBER_OF_ARG	Niepoprawna ilość argumentów przy uruchumieniu programu.
COHERENT_ERR	Podany graf nie jest spójny.
INADEQUATE_NUMBERS_ERR	Niepoprawnie podane numery węzłów w celu znalezienia najkró†szej ścieżki między nimi.

Reszta błędów w trybie Dijkstra jest adekwatna do błędów w trybie podstawowym. Przykładowo błędy z czytaniem pliku, podawaniem argumentów, nieznajomości przez program flag itp.

5 Testowanie:

5.1 Ogólny przebieg testowania:

Do przetestowania programu, zespół użyje przygotowanych grafów, na których będziemy operować i sprawdzać wyniki. Przykładowo, zespół opracuje sobie 4 przykładowe grafy z przykładowymi wagami krawędzi oraz policzy właściwe wyniki na kartce. Następnie, obliczone już wyniki będziemy testować w programie, pisząc oddzielne od programu testy algorytmów Dijkstry oraz BFS. Zespół nie zamierza testować czytania oraz możliwości zapisu grafów, ponieważ graf jest rzeczą niezbędną do działania algorytmów Dijkstry oraz BFS. Przykładowe grafy, które zostały przez zespół wymyślone udostępnimy użytkownikowi w katalogu przykłady. Do testowania w miarę możliwości będziemy też się starać używać biblioteki Assert.