|  |  |  |
| --- | --- | --- |
| **SPRAWOZDANIE** | | **Data wykonania:**  03.03.2022r – 31.03.2022r |
| **Tytuł Projektu:** | **Wykonał:** | **Sprawdził:** |
| *Operowanie na grafach oraz ich generowanie – program GraphSolver* | *Wojciech Wasilewski*  *Wiktor Zaremba* |  |

**Sprawozdanie z wykonania programu GraphSolver**

[**1. Opis zagadnienia i programu** 2](#_Toc99662825)

[1.1 Zasada działania algorytmu BFS 3](#_Toc99662826)

[1.2 Zasada działania algorytmu Dijkstra 4](#_Toc99662827)

[1.3 Operowanie na plikach wejściowych/wyjściowych 5](#_Toc99662828)

[2. Opis wywołania programu 6](#_Toc99662829)

[3. Testy rozwiązania dla różnych danych wejściowych 7](#_Toc99662830)

[3.1 Generacja graf 10x5 wraz z BFS i Dijkstra 8](#_Toc99662831)

[3.2 Odczyt graf 10x5 wraz z BFS 8](#_Toc99662832)

[3.3 Odczyt graf 10x5 wraz z BFS i Dijkstra 8](#_Toc99662833)

[3.4 Generacja graf 30x15 wraz z BFS i Dijkstra 9](#_Toc99662834)

[3.6 Generacja graf 100x300 wraz z BFS i Dijkstra 12](#_Toc99662835)

[4. Wnioski i spostrzeżenia o błędach 13](#_Toc99662836)

[4.1 Wykrycie błędnego argumentu 13](#_Toc99662837)

[4.2 Wykrycie błędnego wariantu 13](#_Toc99662838)

[4.3 Wykrycie błędnej flagi 13](#_Toc99662839)

[4.4 Błędny format pliku 13](#_Toc99662840)

[4.5 Wykrycie nieistniejącego pliku 15](#_Toc99662841)

# 1. Opis zagadnienia i programu

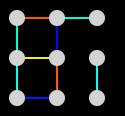
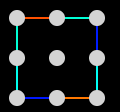
Grafy są zbiorami wierzchołków, które połączone są krawędziami. W naszym przypadku każda krawędź ma do siebie przypisaną wagę. Nasz projekt głównie zajmował się grafami, które można przedstawić jako prostokąt ułożony z wierzchołków, w którym pomiędzy dwoma sąsiadującymi wierzchołkami znajduje się krawędź:

Obraz zawierający obiekt na zewnątrz

Opis wygenerowany automatycznie

Kolory na rysunku oznaczają wagi krawędzi (ciemnoniebieski – minimum, ciemnoczerwony – maksimum).

Program, którym zajmowaliśmy się w ramach tego projektu potrafi wygenerować taki graf o zadanej liczbie wierszy i kolumn oraz przedziale, z którego losowo dobierane są wagi krawędzi. Umożliwia również zarówno jego zapis do pliku o zadanym formacie, jak i możliwość odczytu. Po generacji lub odczytaniu grafu z pliku program potrafi sprawdzić jego spójność. Graf jest spójny, gdy istnieje w nim droga (czyli ciąg krawędzi) pomiędzy dwoma dowolnymi wierzchołkami. W skrócie, gdy nie ma odosobnionego wierzchołka lub grupy wierzchołków (bez połączenia z resztą grafu). By to sprawdzić, program korzysta z algorytmu BFS (przeszukiwanie wszerz).

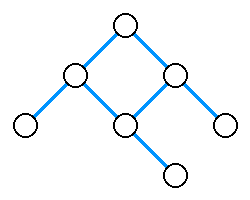


Przykłady grafów niespójnych

# Zasada działania algorytmu BFS

Algorytm breadth-first search, w skrócie BFS służy do przeszukiwania grafu wszerz. Opiera się na przeglądaniu wierzchołków w grafie w kolejności od żródłowego, wgłąb grafu. Przebieg algorytmu:

* *Oznacz wszystkie wierzchołki grafu jako nieodwiedzone.*
* *Odwiedź wierzchołek początkowy, dodaj go do kolejki A.*
* *Dopóki kolejka A nie jest pusta:*
  + *Weź pierwszy wierzchołek z kolejki, usuwając go z niej.*
  + *Odwiedź nieodwiedzone sąsiednie wierzchołki tego wierzchołka, dodaj je do kolejki A.*



Złożoność czasowa algorytmu podczas początkowej części jego działania to O(v), gdzie v to wierzchołek. Oznaczamy w ten sposób każdy wierzchołek w grafie. Następnie, liczba relacji sąsiedztwa dla krawędzi e to O(e), wiec ogólna złożoność algorytmu dla grafu skierowanego to *O(v+e),* dla grafu nieskierowanego wynosiłaby *O(v+2e).*

Informacje o wierzchołkach są przechowywane w strukturze podanej poniżej, zaś wierzchołki znajdować się będą w tablicy wskaźników na te struktury.

Obraz zawierający tekst

Opis wygenerowany automatycznie

# Zasada działania algorytmu Dijkstra

Program również potrafi znaleźć najkrótszą drogę pomiędzy parą wierzchołków w danym grafie. Najkrótszą, czyli o najmniejszej sumie wag poszczególnych krawędzi. W tym celu korzysta z algorytmu Dijkstry, który umożliwia znalezienie najkrótszej drogi z danego wierzchołka do każdego innego (w tym wybranego).

Podczas działania algorytmu dla wierzchołków zostają wyznaczone dwie wartości: koszt dotarcia do danego wierzchołka oraz poprzedni wierzchołek na ścieżce. Dla wszystkich wierzchołków koszt dotarcia wynosi nieskończoność, oprócz wierzchołka w którym się znajdujemy, gdzie koszt wynosi 0. Wierzchołki z nieskończoną droga znajdują się w zbiorze A, są to tzw. wierzchołki nieprzejrzane. Schemat algorytmu:

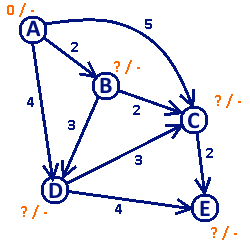
* *Dopóki zbiór A nie jest pusty:*
  + *Weź ze zbioru A wierzchołek o najmniejszym koszcie dotarcia. Ustaw go jako v i usuń ze zbioru A.*
  + *Dla wszystkich krawędzi z wierzchołka (krawędź oznaczamy jako k) wykonaj czynności:*
    - *Oznacz wierzchołek znajdujący się na drugim końcu krawędzi k jako u.*
    - *Jeśli koszt dotarcia do wierzchołka u z wierzchołka v przez krawędź k jest mniejszy od aktualnego kosztu dotarcia do wierzchołka u, to:*
      * *Przypisz kosztowi dotarcia do wierzchołka u koszt dotarcia do wierzchołka v powiększony o wagę krawędzi k.*
      * *Ustaw wierzchołek v jako poprzednik wierzchołka u.*

Złożoność czasowa w przypadku algorytmu Dijkstra wykorzystującego kopiec do przechowywania zbioru A wynosi *O(elog10n),* gdzie n to liczba wierzchołków, e liczba krawędzi.

Informacje o kopcu zaimplementowanym wraz z algorytmem są przechowywane w strukturze, natomiast kopce znajdują się we wskaźnikach na struktury.

Obraz zawierający tekst

Opis wygenerowany automatycznie



# Operowanie na plikach wejściowych/wyjściowych

Program potrafi operować na plikach (odczyt i zapis), więc ustalony został dokładny format takiego pliku.

Obraz zawierający tekst

Opis wygenerowany automatycznie

Przykład pliku tekstowego z zapisanym w nim grafem

W pierwszej linijce takiego pliku znajdują się kolejno liczba wierszy i liczba kolumn danego grafu. Ta informacja jest potrzebna m.in. do ustalenia wielkości grafu (liczby wierzchołków). W każdym kolejnym wierszu znajdują się informacje o połączonych wierzchołkach i wagach krawędzi do nich prowadzących (w linijce 2. – połączenia wierzchołka nr. 0, w linijce 3. – połączenia wierzchołka nr. 1 itd.). Pierwszą liczbą jest numer wierzchołka, z którym połączony jest aktualnie czytany wierzchołek. Następnie jest przerwa (spacja), „:” oraz waga krawędzi prowadząca z czytanego wierzchołka do wierzchołka, którego numer przeczytaliśmy przed chwilą.

Waga powinna zawierać maksymalnie 16 cyfr (bez „.”), ponieważ w przeciwnym razie zostanie zaokrąglona. Wynika to z limitu narzuconego przez typ zmiennej double. Po przeczytaniu wagi zostaje ona zapisana wraz z połączonym wierzchołkiem, musi nastąpić minimum jedna przerwa ‘’ ‘’ i znowu zostaje podany numer wierzchołka, który jest połączony z aktualnie sprawdzanym. Format ten jest powtarzany dla każdego wierzchołka w grafie.

Jak widać wyżej, wagi jednej krawędzi mogą się różnić między sobą (np. krawędź od 0 do 1 jest równa 0.8864916775696521, a w drugą stronę już 0.4630166785185348). Program potrafi zarówno wygenerować i odczytać taki graf, jak i taki, w którym każda krawędź ma tylko jedną wagę, oraz na nich operować.

# 2. Opis wywołania programu

Program GraphSolver jest kompilowany przy pomocy polecenia make w wierszu poleceń, następnie w celu wyświetlenia instrukcji działania należy go uruchomić poleceniem ./graf. Wyświetli nam się poniższe okno:

Obraz zawierający tekst

Opis wygenerowany automatycznie

Użycie z generacja grafu: ./graf -g wysokosc szerokosc min max wariant [-s nazwa\_pliku] [-b] [-d poczatek koniec]

Generacja grafu o podanej wysokości, szerokości, z krawędziami o wagach z danego przedziale liczb double, z możliwością wariantu 0 (krawędź ma tylko jedna wagę) lub 1 (krawędź może mieć różne wagi w dwie strony

-s nazwa\_pliku opcjonalna flaga pozwalająca zapisać wygenerowany graf do pliku nazwa\_pliku, która należy podać przed pozostałymi flagami

Użycie z czytaniem grafu z pliku: ./graf -r nazwa\_pliku [-b] [-d poczatek koniec]

Przeczytanie grafu z pliku nazwa\_pliku

Opcjonalne flagi w obu przypadkach:

-b sprawdzenie spójności grafu

-d poczatek koniec znalezienie najkrótszej ścieżki pomiędzy wierzchołkami poczatek i koniec

# 3. Testy rozwiązania dla różnych danych wejściowych

3.1 Generacja graf 10x5 wraz z BFS i DijkstraObraz zawierający tekst

Opis wygenerowany automatycznie

3.2 Odczyt graf 10x5 wraz z BFSObraz zawierający tekst, urządzenie, pomarańczowy, pomiar

Opis wygenerowany automatycznie

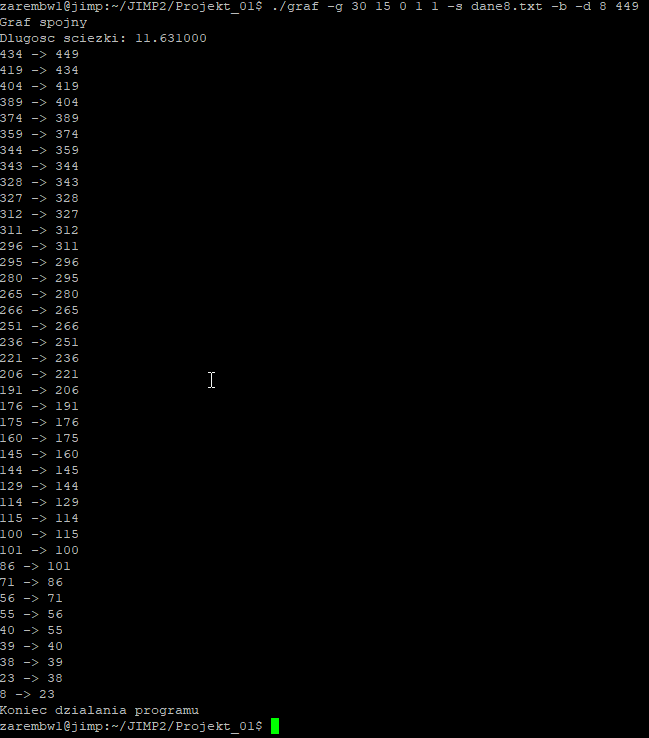
# 3.3 Odczyt graf 10x5 wraz z BFS i Dijkstra

Obraz zawierający tekst

Opis wygenerowany automatycznie

# 3.4 Generacja graf 30x15 wraz z BFS i Dijkstra

Obraz zawierający tekst, urządzenie, wskaźnik, pomiar

Opis wygenerowany automatycznie

Obraz zawierający tekst

Opis wygenerowany automatycznie

3.5 Generacja graf 50x100 wraz z BFS i Dijkstra

­­­Obraz zawierający tekst

Opis wygenerowany automatycznie

# 3.6 Generacja graf 100x300 wraz z BFS i Dijkstra

Obraz zawierający tekst

Opis wygenerowany automatycznie

# 4. Wnioski i spostrzeżenia o błędach

Program poprawnie reaguje na błędnie wprowadzone dane zarówno przy wprowadzaniu flag i danych w wierszu poleceń, jak i skutecznie wykrywa niepoprawne formaty plików wejściowych.

# 4.1 Wykrycie błędnego argumentu

Powinna znajdować w argumencie piątym flaga „-d” zamiast numeru wierzchołka

Obraz zawierający tekst, urządzenie, zegarek, pomiar

Opis wygenerowany automatycznie

# 4.2 Wykrycie błędnego wariantu

Program wykrył użycie błędnego wariantu, dozwolone tylko 0 lub 1 (krawędzie z takimi samymi bądź różnymi wagami w obie strony)



# 4.3 Wykrycie błędnej flagi

Obraz zawierający tekst, wskaźnik, urządzenie

Opis wygenerowany automatycznie

# 4.4 Błędny format pliku

Program wykrył, że w pliku dane3.txt znajduje się niepoprawny format pliku, w linijce z rozmiarem grafu znalazły się litery „xD”.



Obraz zawierający tekst

Opis wygenerowany automatycznie

# 4.5 Wykrycie nieistniejącego pliku

Obraz zawierający tekst

Opis wygenerowany automatycznie

# 4.6 Wykrycie błędnego przedziału losowania



Program nie obsługuje wag ujemnych

Nasz program posiada ograniczenie rozmiaru grafu w postaci technicznego ograniczenia typu danej int, która wynosi 2147483647 wierzchołków, jest to graf o wymiarach ok 45000x45000, zająłby on około 200GB, a więc szansa na to że użytkownik zechce wygenerować tak ogromny graf jest niska.