Politechnika Śląska

Wydział Automatyki, Elektroniki i Informatyki

**Programowanie**

**Komputerów**

Temat projektu: (21) Odległość między miastami 2

|  |  |
| --- | --- |
| Autor: | Ignacy Byrdziak |
| Prowadzący: | dr hab. inż. Karolina Nurzyńska |
| Rok akademicki: | 2019/2020 |
| Kierunek: | Informatyka |
| Rodzaj studiów: | SSI |
| Semestr: | 2 |
| Termin oddania: | 2020-09-18 |

# Treść zadania

Napisać program obliczający najkrótszą trasę między podanym miastem, a wszystkimi innymi podanymi w pliku miastami, przy użyciu algorytmu Dijkstry. Program działa zależnie od zadanych przez użytkownika parametrów. Działanie programu polega na wczytywaniu miast i odległości z pliku wejściowego, po otrzymaniu wszystkich dystansów, oblicza on najkrótszą możliwą drogę między podanym przez użytkownika miastem, a wszystkimi pozostałymi, a następnie zależnie od podanym parametrów, zapisuje wynik to pliku lub wypisuje w konsoli. Program wykorzystuje jeden plik wejściowy, w którym znajdują się dwa miasta rozdzielone białym znakiem, a jako trzeci element wiersza znajduje się tam odległość między dwoma miastami. Przykładowy plik danych ma postać:

Andrychow Bielsko 4  
Andrychow Halcnow 8  
Bielsko Cieszyn 8  
Bielsko Halcnow 11  
Cieszyn Dobczyce 7  
Cieszyn Ilownica 2  
Cieszyn Frankfurt 4  
Dobczyce Elk 9  
Dobczyce Frankfurt 14  
Elk Frankfurt 10  
Frankfurt Gniezno 2  
Gniezno Halcnow 1  
Gniezno Ilownica 6  
Halcnow Ilownica 7

# Algorytmy

Algorytm Dijkstry: Mając dany graf z wyróżnionym wierzchołkiem (źródłem) algorytm znajduje odległości od źródła do wszystkich pozostałych wierzchołków. Algorytm Dijkstry jest przykładem algorytmu zachłannego. Złożoność obliczeniowa algorytmu:

Złożoność obliczeniowa algorytmu Dijkstry zależy od liczby V wierzchołków i E krawędzi grafu. O rzędzie złożoności decyduje implementacja kolejki priorytetowej:

* wykorzystując „naiwną” implementację poprzez zwykłą tablicę, otrzymujemy algorytm o złożoności O(V^{2})
* w implementacji kolejki poprzez kopiec, złożoność wynosi O(E\log V)

Algorytm Dijkstry nie działa, jeśli w grafie występują krawędzie z ujemnymi wagami – w tym wypadku używa się wolniejszego, lecz bardziej ogólnego algorytmu Bellmana-Forda. Jeśli graf nie jest ważony (wszystkie wagi mają wielkość 1), zamiast algorytmu Dijkstry wystarczy algorytm przeszukiwania grafu wszerz.

**Specyfikacja zewnętrzna**

Program jest uruchamiany z listy poleceń z możliwymi przełącznikami:

-i plik z danymi wejściowymi

-o plik wyjściowy z wypisanymi miastami i ich odległością do zadanego przez użytkownika miasta

-p wypisanie rozwiązania do konsoli

-h panel pomocy

Jeśli nie zostaną wywołane odpowiednie parametry lub odbędzie się to w zły sposób, wyświetli się panel pomocy. Pliki są plikami tekstowymi, ale mogą mieć dowolne rozszerzenie (lub go nie mieć).

Przykład podania parametrów:

-i citiesInput.txt -o citiesOutput.txt -p –h

**Specyfikacja wewnętrzna**

Program został zrealizowany zgodnie z paradygmatem strukturalnym.

W programie rozdzielono interfejs (komunikacje z użytkownikiem) od logiki

aplikacji (obliczania ścieżek).

**Ogólna struktura programu**

Jeśli zostaną podane odpowiednie argumenty, to program najpierw wczytuje dane z pliku wejściowego, przy pomocy bufora, potem literka po literce zapisuje je w odpowiedniej zmiennej „c”. Kiedy napotka na biały znak, zapisuje nowe miasto do listy przy pomocy funkcji **AddNewCity** (po wcześniejszym sprawdzeniu, czy takie miasto już nie występuje w liście przy pomocy funkcji **FindingDuplicateCities**), a co trzecie słowo z pliku, zapisuje te dane do listy dystansów podobnymi funkcjami. Następnie program tworzy graf przy pomocy funkcji **CreateGraph**, a potem dodaje do niego krawędzie funkcją **AddEdge**. Tak stworzony graf wykorzystuje do użycia na nim algorytmu Dijkstry, przy pomocy funkcji Dijkstra (lub **DijkstraOutput** – jeśli plik ma zostać zapisany do pliku) oraz pomniejszych podfunkcji. Program zapisuje rozwiązanie do pliku albo konsoli, a następnie zwalnia zaalokowaną pamięć.

**Szczegółowy opis typów i funkcji**

Szczegółowy opis typów i funkcji zawarty jest w dokumentacji doxygen otwieranym przez plik index w folderze html w paczce ze sprawozdaniem.

**Testowanie**

Program został przetestowany na różnego rodzaju plikach. Wpisanie złej nazwy pliku lub istnienie pliku wyjściowego (co powodowałoby jego nadpisanie) generuje odpowiedni komunikat.

Program został sprawdzony pod katem wycieków pamięci.

**Wnioski**

Program do znajdywania najktrótszej ścieżki pomiędzy miastami nie jest programem skomplikowanym, ale wymaga przemyślanego przechowywania danych oraz struktur oraz rozpatrzenia wielu wyjątków oraz jego implementacja w języku C nie była dla mnie łatwym zadaniem. Udało mi się zrealizować cel projektu. Największe problemy sprawiło mi implementacja komunikacji z użytkownikiem za pomocą parametrów oraz realizacja samego algorytmu Dijkstry. Projekt nauczył mnie implementacji dynamicznych struktur danych, ich dealokowania oraz operowania na plikach zewnętrznych przy użyciu konsoli.