Politechnika Śląska
Wydział Informatyki Elektroniki i Informatyki

Podstawy Programowania Komputerów

Mapa (18)

Autor	Mateusz Grabarczyk
Prowadzący	dr. Inż. Adam Gudyś
Rok akademicki	2020/2021
Kierunek	Informatyka
Rodzaj studiów	SSI
Semestr	1
Termin laboratoriów	Poniedziałek, 10:30-12:45 Środa, 8:00-10:15
Sekcja	11
Termin oddania sprawozdania	

---

1 Treść zadania

Napisać program, który umożliwia znalezienie najkrótszej trasy między dwoma miastami. Miasta połączone są drogami o pewnej długości. Drogi są jednokierunkowe. Plik mapy dróg ma następującą postać: W każdej linii podana jest jedna droga:

<miasto początkowe> <miasto końcowe> <odległość>

Program uruchamiany jest z linii poleceń z wykorzystaniem następujących przełączników (kolejność przełączników jest dowolna):

-d plik wejściowy z drogami

-t plik wejściowy z trasami do wyznaczenia

-o plik wynikowy z wyznaczony trasami

Uruchomienie programu bez parametrów powoduje wypisanie krótkiej instrukcji.

2 Analiza zadania

Zagadnienie przedstawia problem znajdywania najkrótszej drogi między drogami podanymi w pliku tekstowym.

2.1 Struktury danych

W programie wykorzystano macierz sąsiedztwa do przechowywania wartości. Macierz sąsiedztwa jest dynamiczną tablicą dwuwymiarową. Taka struktura umożliwia łatwy dostęp do danych.

2.2 Algorytmy

Program znajduje najkrótsze drogi między podanymi miastami poprzez wykorzystanie algorytmu Dijkstry. Wykorzystując zwykłą implementację poprzez zwykłą tablicę, otrzymujemy algorytm o złożoności O(V^2). Algorytm służy do znajdowania najkrótszej ścieżki z pojedynczego źródła w grafie o nieujemnych wagach krawędzi.

3 Specyfikacja zewnętrzna

Program uruchamiany jest z linii poleceń. Należy przekazać do programu nazwy plików: dwóch wejściowych oraz jednego wyjściowego po odpowiednich przełącznikach (odpowiednio –d dla pliku wejsciowego z drogami, -t dla pliku wejściowego z trasami do wyznaczenia i -o plik wyjściowy wynikowy z wyznaczonymi trasami np. :

Program –d drogi.txt -t trasy.txt -o wyniki.txt

Pliki są plikami tekstowymi. Przełączniki mogą być podane w dowolnej kolejności. Urochomienie programu bez przynajmniej jednego parametru powoduje wyświetlenie krótkiej instrukcji : “Podane argumenty sa zle”.

4 Specyfikacja wewnetrzna

Program został zrealizowany zgodnie z paradygmatem strukturalnym.

4.1 Ogólna struktura programu

Na początku sprawdzamy, czy argumenty programu są prawidłowe. Jeśli program nie został wywołany prawidłowo, zostaje wypisana krótka instrukcja. W funkcji glownej tworzony jest vector miast, które są obecne w grafie oraz graf za pomocą tablicy dynamicznej dwuwymiarowej, o wielkości vectora miast. Następnie graf jest wypełniany wartością 0. Funkcja *zapisanie_do_grafu* sprawdza, czy graf udało się wypełnić. Jeśli nie, to program zostaje zakończony za pomocą return –1. Następnie przy użyciu funkcji *pobranie_tras* pobieramy do vectora par (string, string) trasy do wyznaczenia. Jeśli vector będzie pusty, wtedy program zostaje zakończony za pomocą return –1. W przed ostatnim kroku przy użyciu instrukcji warunkowej sprawdzamy, czy oba miasta z każdego elementu vectora par są w vectorze z miastami obecnymi w grafie. Jeśli tak, wtedy wywołujemy funkcję *dijkstra* odpowiednimi argumentami. Funkcja ta znaduje najkrótsze ścieżki do wszystkich pozostałych miast, a następnie zapisuje do pliku wyjściowego z wyznaczonymi trasami daną trasę z miasta startowego do końcowego. Jeśli nie, wtedy do pliku wyjściowego wypisywane są dwa podane miasta oraz krótka instrukcja, że trasy nie da się wyznaczyć. Na samym końcu zwalniamy pamięć grafu.

4.2 Szczegółowy opis typów i funkcji

Szczegółowy opis typów i funkcji jest zawarty w załączniku.

5 Testowanie

Program został przetestowany na różnego rodzaju plikach. Program pomija linie w plikach wejściowych, które są błędnie sformatowane (np. W pliku z drogami będzie o jedno za mało słowo, czy nie będzie odległości między miastami). Jeśli któryś z plików wejściowych będzie pusty, wtedy program zakończy się przy pomocy *return –1;* Jeśli podany plik wynikowy nie istnieje, program automatycznie utworzy plik o takiej nazwie i przekaże do niego wyniki działania programu. Program został sprawdzony pod kątem wycieków pamięci przy użyciu Valgrind.

6 Wnioski

Program do wyznaczania najkrótszej drogi między miastami jest programem prostym, lecz wymaga bardzo dobrej znajmości i zrozumienia algorytmu, na którym się opiera. Najbardziej wymagającym okazało się stworzenie grafu, na którym można w miarę prosto oprzeć algorytm Dijkstry. Nie łatwym zadaniem było też zapisanie wyników do pliku wyjściowego, ponieważ trzeba było użyć rekurencji. Laboratorium pomogło mi w obsłudze różnych struktur danych, których użyłem, przykładowo wektorów par. Ostatnią rzeczą, która była już tylko szukaniem dziury w programie, było wyłapywanie błędów i ich naprawa.

Dodatek

Mateusz Grabarczyk

Szczegółowy opis typów i funkcji

Projekt Mapa

Wygenerowano przez Doxygen 1.8.20

Rozdział 1

Indeks plików

1.1 Lista plików

Tutaj znajduje się lista wszystkich plików z ich krótkimi opisami:

Rozdział 2

Dokumentacja plików

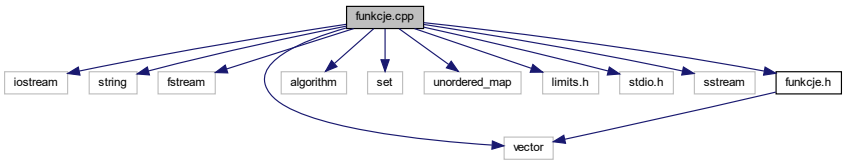
2.1 Dokumentacja pliku drogi.txt

2.2 Dokumentacja pliku DrogiDoWyznaczenia.txt

2.3 Dokumentacja pliku funkcje.cpp

```
#include <iostream>
#include <string>
#include <fstream>
#include <vector>
#include <algorithm>
#include <set>
#include <unordered_map>
#include <limits.h>
#include <stdio.h>
#include <sstream>
#include "funkcje.h"
```

Wykres zależności załączania dla funkcje.cpp:



Funkcje

- void drukuj_sciezke (int ∗p, int ∗∗odl, vector< string > miasta, int m, string nazwa_pliku)

  Drukowanie sciezki rekurencyjnie od miasta startowego do miasta koncowego.
- vector< string > vector_z_miastami (string nazwa_pliku)

  Pobiera miasta obecne w grafie.

- vector< pair< string, string > > pobranie_tras (const string &nazwa_pliku)

  Pobranie tras, ktore trzeba wyznaczyc.
- int zwroc_indeks (vector< string > miasta, string miasto)

  Funkcja zwracajaca indeks miasta w wektorze 'miasta'.
- bool zapisanie_do_grafu (string nazwa_pliku, vector< string > miasta, int **graf)

  Funkcja zapisujaca polaczenia miedzy miastami do grafu.
- int zwroc_najblizsze_miasto (int *odleglosci, bool *sprawdzone, vector< string >miasta)

  Funkcja zwracajaca indeks miasta z najnizsza odlegloscia.
- void dijkstra (string m_start, string m_koniec, vector< string >miasta, int **graf, string nazwa_pliku_wyjsc)

  Glowny algorytm odpowiadajacy za wyznaczenie najkrotszych drog z miasta startowego do reszty miast.

2.3.1 Dokumentacja funkcji

2.3.1.1 dijkstra()

```
void dijkstra (
            string m_start,
            string m_koniec,
            vector< string > miasta,
            int ** graf,
            string nazwa_pliku_wyjsc )
```

Glowny algorytm odpowiadajacy za wyznaczenie najkrotszych drog z miasta startowego do reszty miast.

Parametry

m_start	miasto startowe
m_koniec	miasto koncowe
miasta	wektor miast obecnych w grafie
graf	macierz sasiedztwa (graf)
nazwa_pliku_wyjsc	nazwa pliku wyjsciowego, do ktorego zapiszemy trasy

2.3.1.2 drukuj_sciezke()

```
void drukuj_sciezke (
            int * p,
            int ** odl,
            vector< string > miasta,
            int m,
            string nazwa_pliku )
```

Drukowanie sciezki rekurencyjnie od miasta startowego do miasta koncowego.

Parametry

p	lista poprzednikow
odl	graf z odleglosciami
miasta	vector miast w grafie
m	miasto koncowe
nazwa_pliku	nazwa pliku wyjsciowego, do ktorego zapisujemy dana trase

2.3.1.3 pobranie_tras()

```
vector<pair<string, string> > pobranie_tras (
            const string & nazwa_pliku )
```

Pobranie tras, ktore trzeba wyznaczyc.

Parametry

nazwa_pliku	nazwa pliku z trasami, ktore trzeba wyznaczyc

Zwraca

wektor par string string, ktorego pierwszym elementem pary jest miasto startowe, a drugim koncowe

2.3.1.4 vector_z_miastami()

```
vector<string> vector_z_miastami (
            string nazwa_pliku )
```

Pobiera miasta obecne w grafie.

Parametry

nazwa_pliku	nazwa pliku z drogami

Zwraca

set z miastami obecnymi w grafie

2.3.1.5 zapisanie_do_grafu()

```
bool zapisanie_do_grafu (
            string nazwa_pliku,
```

```
            vector< string > miasta,
            int ** graf )
```

Funkcja zapisujaca polaczenia miedzy miastami do grafu.

Parametry

nazwa_pliku	nazwa pliku tekstowego z miastami
miasta	vector miast obecnych w grafie
graf	graf prezentujacy polaczenia miedzy miastami

Zwraca

true or false, w zaleznosci, czy plik jest pusty czy nie

2.3.1.6 zwroc_indeks()

```
int zwroc_indeks (
            vector< string > miasta,
            string miasto )
```

Funkcja zwracajaca indeks miasta w wektorze 'miasta'.

Parametry

miasta	wektor z miastami obecnymi w grafie
miasto	miasto, ktorego indeks chcemy otrzymac

Zwraca

indeks miasta w wektorze 'miasta'

2.3.1.7 zwroc_najblizsze_miasto()

```
int zwroc_najblizsze_miasto (
            int * odleglosci,
            bool * sprawdzone,
            vector< string > miasta )
```

Funkcja zwracajaca indeks miasta z najnizsza odlegloscia.

Parametry

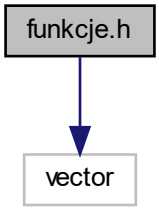
odleglosci	tablica zawierajaca odleglosci
sprawdzone	tablica bool zawierajaca wierzcholki (sprawdza czy sa odwiedzone czy tez nie)
miasta	vector miast (sluzy tutaj do wyznaczenia ilosci wierzcholkow w grafie)

Zwraca

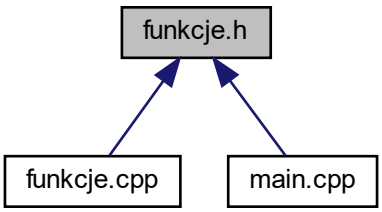
zwraca indeks miasta z najnizsza odlegloscia (waga)

2.4 Dokumentacja pliku funkcje.h

`#include <vector>`
Wykres zależności załączania dla funkcje.h:



Ten wykres pokazuje, które pliki bezpośrednio lub pośrednio załączają ten plik:



Funkcje

- vector< string > vector_z_miastami (string nazwa_pliku)

  Pobiera miasta obecne w grafie.
- vector< pair< string, string > > pobranie_tras (const string &nazwa_pliku)

  Pobranie tras, ktore trzeba wyznaczyc.
- int zwroc_indeks (vector< string > miasta, string miasto)

  Funkcja zwracajaca indeks miasta w wektorze 'miasta'.
- bool zapisanie_do_grafu (string nazwa_pliku, vector< string > miasta, int **graf)

  Funkcja zapisujaca polaczenia miedzy miastami do grafu.
- int zwroc_najblizsze_miasto (int *odleglosci, bool *sprawdzone, vector< string >miasta)

  Funkcja zwracajaca indeks miasta z najnizsza odlegloscia.

- void dijkstra (string m_start, string m_koniec, vector< string >miasta, int **graf, string nazwa_pliku_wyjsc)

  Glowny algorytm odpowiadajacy za wyznaczenie najkrotszych drog z miasta startowego do reszty miast.
- void drukuj_sciezke (int *p, int **odl, vector< string > miasta, int m, string nazwa_pliku)

  Drukowanie sciezki rekurencyjnie od miasta startowego do miasta koncowego.

2.4.1 Dokumentacja funkcji

2.4.1.1 dijkstra()

```
void dijkstra (
            string m_start,
            string m_koniec,
            vector< string > miasta,
            int ** graf,
            string nazwa_pliku_wyjsc )
```

Glowny algorytm odpowiadajacy za wyznaczenie najkrotszych drog z miasta startowego do reszty miast.

Parametry

m_start	miasto startowe
m_koniec	miasto koncowe
miasta	wektor miast obecnych w grafie
graf	macierz sasiedztwa (graf)
nazwa_pliku_wyjsc	nazwa pliku wyjsciowego, do ktorego zapiszemy trasy

2.4.1.2 drukuj_sciezke()

```
void drukuj_sciezke (
            int * p,
            int ** odl,
            vector< string > miasta,
            int m,
            string nazwa_pliku )
```

Drukowanie sciezki rekurencyjnie od miasta startowego do miasta koncowego.

Parametry

p	lista poprzednikow
odl	graf z odleglosciami
miasta	vector miast w grafie
m	miasto koncowe
nazwa_pliku	nazwa pliku wyjsciowego, do ktorego zapisujemy dana trase

2.4.1.3 pobranie_tras()

```
vector<pair<string, string> > pobranie_tras (
            const string & nazwa_pliku )
```

Pobranie tras, ktore trzeba wyznaczyc.

Parametry

nazwa_pliku	nazwa pliku z trasami, ktore trzeba wyznaczyc

Zwraca

wektor par string string, ktorego pierwszym elementem pary jest miasto startowe, a drugim koncowe

2.4.1.4 vector_z_miastami()

```
vector<string> vector_z_miastami (
            string nazwa_pliku )
```

Pobiera miasta obecne w grafie.

Parametry

nazwa_pliku	nazwa pliku z drogami

Zwraca

set z miastami obecnymi w grafie

2.4.1.5 zapisanie_do_grafu()

```
bool zapisanie_do_grafu (
            string nazwa_pliku,
            vector< string > miasta,
            int ** graf )
```

Funkcja zapisujaca polaczenia miedzy miastami do grafu.

Parametry

nazwa_pliku	nazwa pliku tekstowego z miastami
miasta	vector miast obecnych w grafie
graf	graf prezentujacy polaczenia miedzy miastami

Zwraca

true or false, w zaleznosci, czy plik jest pusty czy nie

2.4.1.6 zwroc_indeks()

```
int zwroc_indeks (
            vector< string > miasta,
            string miasto )
```

Funkcja zwracajaca indeks miasta w wektorze 'miasta'.

Parametry

miasta	wektor z miastami obecnymi w grafie
miasto	miasto, ktorego indeks chcemy otrzymac

Zwraca

indeks miasta w wektorze 'miasta'

2.4.1.7 zwroc_najblizsze_miasto()

```
int zwroc_najblizsze_miasto (
            int * odleglosci,
            bool * sprawdzone,
            vector< string > miasta )
```

Funkcja zwracajaca indeks miasta z najnizsza odlegloscia.

Parametry

odleglosci	tablica zawierajaca odleglosci
sprawdzone	tablica bool zawierajaca wierzcholki (sprawdza czy sa odwiedzone czy tez nie)
miasta	vector miast (sluzy tutaj do wyznaczenia ilosci wierzcholkow w grafie)

Zwraca

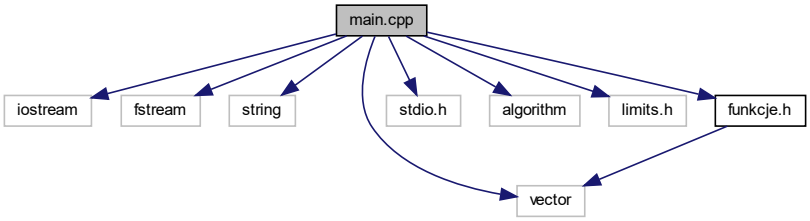
zwraca indeks miasta z najnizsza odlegloscia (waga)

2.5 Dokumentacja pliku main.cpp

```
#include <iostream>
#include <fstream>
```

```
#include <string>
#include <vector>
#include <stdio.h>
#include <algorithm>
#include <limits.h>
#include "funkcje.h"
```

Wykres zależności załączania dla main.cpp:



Funkcje

- int main (int argc, char **argv)

2.5.1 Dokumentacja funkcji

2.5.1.1 main()

```
int main (
            int argc,
            char ** argv )
```

2.6 Dokumentacja pliku zapis.txt

2.7 Dokumentacja pliku zapisane.txt

Indeks