Politechnika Śląska
Wydział Informatyki, Elektroniki i Informatyki

Podstawy Programowania Komputerów

Cykle

autor	Jakub Gazda
prowadzący	dr inż. Bożena Wieczorek
rok akademicki	2021/2022
kierunek	Informatyka
rodzaj studiów	SSI
semestr	1
termin laboratorium	środa, 10:15 – 11:45
sekcja	5
termin oddania sprawozdania	2021-01-24

1. Treść zadania

Napisać program do wyznaczania cykli w grafie skierowanym. W pliku wejściowym podane są krawędzie pewnego grafu. Są one zapisane w następujący sposób:
`<wierzchołek_początkowy> -> <wierzchołek_końcowy>`
Poszczególne krawędzie są rozdzielone przecinkami. Przykładowy plik wejściowy:
`2 -> 3, 1 -> 3, 3 -> 3, 3 -> 2`

W pliku wynikowym zostają zapisane znalezione cykle (każdy cykl w osobnej linii) lub informacja, że w grafie nie występują cykle. Program uruchamiany jest z linii poleceń z wykorzystaniem następujących przełączników:

`-g` plik wejściowy z grafem

`-c` plik wyjściowy ze znalezionymi cyklami

Uruchomienie programu bez parametrów powoduje wypisanie krótkiej informacji o tym, jak użyć programu.

2. Analiza zadania

Zagadnienie przedstawia problem przeszukiwania grafów skierowanych oraz znajdowania w nich cykli.

2.1. Struktury danych

Struktury danych użyte w programie do reprezentacji grafu skierowanego to mapa wierzchołków oraz mapa krawędzi. W obu przypadkach klucze są typu **int** i reprezentują konkretny wierzchołek. Wartości są kolejno:

- typu **bool** – wartość informuje, czy dany wierzchołek został odwiedzony podczas przeszukiwania grafu;
- wektorem **int**ów – zawiera on wszystkie wierzchołki do których można dostać się bezpośrednio za pomocą jednej krawędzi z wierzchołka danego kluczem (z uwzględnieniem kierunku krawędzi).

Mapy pozwalają na szybkie i proste przeszukiwanie grafu.

2.2. Algorytmy

Program wykorzystuje rekurencyjny algorytm przeszukiwania w głąb. Złożoność czasowa tego algorytmu to $O(|V| + |E|)$[1], gdzie V – wierzchołki, E – krawędzie.

3. Specyfikacja zewnętrzna

Program uruchamiany jest z linii poleceń. Do programu należy przekazać nazwy pliku wejściowego i wyjściowego po odpowiednich przełącznikach podanych w dowolnej kolejności (odpowiednio **-g** dla pliku wejściowego oraz **-c** dla pliku wyjściowego), np.

program.exe -g input.txt -c output.txt

Uruchomienie programu bez parametrów powoduje wyświetlenie krótkiej instrukcji. Uruchomienie programu z błędnymi przełącznikami powoduje wyświetlenie komunikatu: **Zostały podane nieprawidłowe parametry!**

Plik z danymi wejściowymi zawiera krawędzie grafu zapisane w taki, sposób: **<wierzchołek> -> <wierzchołek>**. Kolejne krawędzie grafu są oddzielone przecinkiem. Program pomija nadmiarowe spacje i znaki nowej linii. Jest jednak podatny na „sklejanie" drugiego wierzchołka krawędzi z strzałką oraz pierwszego wierzchołka z przecinkiem, np.:

->3 lub **,21**.

W takim wypadku program wypisuje krótką instrukcję z żądanym formatem danych wejściowych. To samo wypisze również, jeśli plik zawiera nieobsługiwane znaki lub jest pusty.

Jeśli podczas otwierania plików wystąpi błąd, program poinformuje o tym wypisując: **Wystapil blad podczas otwierania pliku!**

Plik wyjściowy zawiera informację o braku cykli lub znalezione cykle (każdy w osobnej linii) zapisane w formacie:

<wierzchołek> -> <wierzchołek> -> … -> <wierzchołek>

4. Specyfikacja wewnętrzna

4.1. Ogólna struktura programu

Program rozpoczyna się od sprawdzenia poprawności podanych parametrów (w innym przypadku jest zamykany) i otwarcia plików podanych przez użytkownika. Następnie wywoływana jest funkcja **make_digraph**, która sczytuje dane z pliku wejściowego i wypełnia nimi strukturę typu **Digraph**. Jeśli podczas jej działania wystąpił błąd funkcja zwraca wartość **false**, a program oraz obsługiwane pliki są zamykane. W innym przypadku wywoływana jest funkcja **search_cycles**. Wywołuje ona kolejną funkcję - **DFS**, której uprzednio przygotowuje potrzebne dane oraz sprawdza zwracaną przez nią wartość. Funkcja **DFS** przeszukuje graf z użyciem rekurencyjnego algorytmu przeszukiwania w głąb. Znalezione cykle zapisywane są w wektorze przekazywanym jako parametr, a sama funkcja nic nie zwraca. Jeśli wektor cykli jest pusty (żaden cykl nie został znaleziony) **search_cycles** zwraca wartość **false**, a program informuje o braku cykli. W innym przypadku zwracane jest **true**, a następnie

znalezione cykle wypisywane są w pliku wyjściowym. Na końcu zamykane są obsługiwane pliki oraz program.

4.2. Szczegółowy opis typów i funkcji

Szczegółowy opis typów i funkcji znajduje się w załączniku.

5. Testowanie

Program został przetestowany dla nieistniejących plików. Jeśli dotyczy to pliku wejściowego, zwraca on informację o błędzie odczytu plików, a jeśli wyjściowego to plik taki jest tworzony. Jeśli plik wejściowy jest pusty, zawiera nieobsługiwane ciągi znaków/znaki (inne niż `->` oraz `,`) lub nie zawiera liczb program wypisuje informację o złym formacie danych wejściowych. Program działa dla plików wejściowych z znakami nowej linii oraz nadmiarowymi spacjami. Największy graf dla którego sprawdzana była poprawność danych wyjściowych posiadał 18 wierzchołków i 42 cykle. Program sprawdzany był również na grafach, w których występuje pętla.

6. Wnioski

Najtrudniejsza w napisaniu programu okazała się funkcja **DFS**. Wynikało to z rekurencyjnej natury algorytmu w niej użytego. Skomplikowało to również sposób w jaki niektóre parametry musiały być przekazywane do funkcji.

Nauczyłem się korzystać z generatora dokumentacji, wywoływać i obsługiwać program z parametrami oraz korzystać z map w C++, których nigdy wcześniej nie używałem.

Literatura

[1] https://en.wikipedia.org/wiki/Depth-first_search

Dodatek
Szczegółowy opis typów i funkcji

Projekt PPK

Generated by Doxygen 1.9.3

Chapter 1

Class Index

1.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 2

File Index

2.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 3

Class Documentation

3.1 Digraph Struct Reference

Struktura Digraph reprezentuje graf skierowany.

`#include <functions.h>`

Public Attributes

- std::map< int, bool > **mVertices**

  Mapa o kluczu typu int i wartości typu bool. Klucz reprezentuje wierzchołek grafu, a wartości są przydatne w późniejszym szukaniu cykli.
- std::map< int, std::vector< int > > **mArcs**

  Mapa o kluczu typu int i wartości będącej wektorem intów. Klucz reprezentuje wierzchołek grafu, a wartość przechowuje wszystkie wierzchołki do których można dostać się z wierzchołka danego kluczem.

3.1.1 Detailed Description

Struktura Digraph reprezentuje graf skierowany.

Struktura Digraph posiada dwie składowe reprezentujące kolejno wierzchołki i krawędzie grafu.

The documentation for this struct was generated from the following file:

- functions.h

Chapter 4

File Documentation

4.1 functions.h File Reference

Classes

- struct Digraph

  Struktura Digraph reprezentuje graf skierowany.

Functions

- bool make_digraph (std::ifstream &iFile, Digraph &digraph)

  Funkcja make_digraph służy do wypełnienia zmiennej typu Digraph danymi z pliku wejściowego.
- bool search_cycles (Digraph &digraph, std::vector< std::vector< int > > &vCycles)

  Funkcja search_cycles przygotowuje potrzebne zmienne dla funkcji DFS, którą następnie wywołuje.
- void DFS (std::map< int, bool > mVertices, std::map< int, std::vector< int > > &mArcs, std::map< int, bool > &mVisited, int nVertex, std::vector< int > &vStack, std::vector< std::vector< int > > &vCycles, int nIteration)

  Funkcja DFS służy do szukanania cykli w grafie skierowanym. Funkcja korzysta z rekurencyjnego algorytmu przeszukiwania w głąb.

4.1.1 Function Documentation

4.1.1.1 DFS()

```
void DFS (
            std::map< int, bool > mVertices,
            std::map< int, std::vector< int > > & mArcs,
            std::map< int, bool > & mVisited,
            int nVertex,
            std::vector< int > & vStack,
            std::vector< std::vector< int > > & vCycles,
            int nIteration )
```

Funkcja DFS służy do szukanania cykli w grafie skierowanym. Funkcja korzysta z rekurencyjnego algorytmu przeszukiwania w głąb.

Parameters

mVertices	Mapa reprezentująca wierzchołki grafu. Wartości informują czy dany wierzchołek był już odwiedzony podczas szukania cykli.
mArcs	Mapa reprezentująca krawędzi grafu.
mVisited	Mapa przechowującą informację, czy dla danego wierzchołka były już szukane cykle.
nVertex	Wierzchołek dla którego wywoływana jest funkcja.
vStack	Wektor używany jako stos, do którego odkładane są kolejne wierzchołki wywołań funkcji. Jeśli jego początkowy i ostatni element są sobie równe, to znajduje się w nim cykl.
vCycles	Wektor wektrów przechowujący znalezione cykle.
nIteration	Licznik iteracji funkcji.

Note

Mapa krawędzi oraz wierzchołków przekazywana jest osobno, a nie jako pojedyncza struktura dla oszczędności pamięci. Mapa krawędzi może zostać przekazana przez referencje, jednak mapa wierzchołków dla poprawnego działania funkcji musi zostać przekazana przez kopię.

4.1.1.2 make_digraph()

```
bool make_digraph (
            std::ifstream & iFile,
            Digraph & digraph )
```

Funkcja make_digraph służy do wypełnienia zmiennej typu Digraph danymi z pliku wejściowego.

Parameters

iFile	Zmienna plikowa z danymi wejściowymi, które będą użyte do wypełnienia zmiennej Digraph.
digraph	Zmienna typu Digraph, która będzie wypełniana danymi.

Returns

Funkcja zwraca wartość false, jeśli dane wejściowe są podane w złym formacie (np. pojawia się niepożądany znak). Funkcja zwraca wartość true, jeśli zmienna digraph została pomyślnie zapełniona danymi.

4.1.1.3 search_cycles()

```
bool search_cycles (
            Digraph & digraph,
            std::vector< std::vector< int > > & vCycles )
```

Funkcja search_cycles przygotowuje potrzebne zmienne dla funkcji DFS, którą następnie wywołuje.

Parameters

digraph	Graf w którym szukane będą cykle.
vCycles	Wektor wektorów intów, w którym przechowywane będą znalezione cykle.

Returns

Funkcja zwraca wartość true, jeśli jakieś cykle zostały znalezione. Funkcja zwraca false jeśli żaden cykl nie został znaleziony.

4.2 functions.h

Go to the documentation of this file.

```
1
2 #ifndef define
3 #define define
4
9 struct Digraph
10 {
11     std::map<int, bool> mVertices;
12     std::map<int, std::vector<int» mArcs;
13 };
14
21 bool make_digraph(std::ifstream& iFile, Digraph& digraph);
22
29 bool search_cycles(Digraph& digraph, std::vector<std::vector<int»& vCycles);
30
42 void DFS(std::map<int, bool> mVertices, std::map<int, std::vector<int»& mArcs, std::map<int, bool>&
       mVisited, int nVertex, std::vector<int>& vStack, std::vector<std::vector<int»& vCycles, int
       nIteration);
43 #endif
```