Graf dwudzielny

Wygenerowano przez Doxygen 1.9.6

Rozdział 1

Graf dwudzielny

1.1 Wstep

Program sprawdza czy graf nieskierowany jest grafem dwudzielnym.

1.2 Pliki testowe

W zalaczonych plikach znajduja sie dwa pliki tekstowe:

1. good.txt - plik z grafem nieskierowanym ktory jest dwudzielny. $\sim$ 400 krawedzi
2. bad.txt - plik z grafem nieskeirowanyk ktory nie jest dwudzielny. $\sim$ 400 krawedzi
3. smallBipartite.txt - mniejszy plik z grafem ktory jest dwudzielny
4. smallNOBipartite.txt- mniejszy plik z grafem nie dwudzielnym

Plik z grafem ma nastepujaca postac:

- Kazda krawedz jest podana w osobnej lini; podane sa dwa wierzcholki ktore laczy krawedz.
- W pliku moga wystapic puste linie.
- W linii moga wystapic nadmiarowe znaki biale.

1.3 Instrukcja

Aby prawidlowo uzyc programu, podczas uruchamiania z linni polecen powinno uzyc sie nastepujacych przelacznikow:

1. -i plik wejsciowy z krawedziami grafu
2. -o plik wyjsciowy z wynikami

Przyklad: -i good.txt -o output.txt

Autor

Maciej Fajlhauer

Rozdział 2

Indeks plików

2.1 Lista plików

Tutaj znajduje się lista wszystkich plików z ich krótkimi opisami:

Rozdział 3

Dokumentacja plików

3.1 Dokumentacja pliku function.cpp

```
#include "header.h"
```

Funkcje

- void DFS (int a, map< int, int > &color, map< int, list< int > > &graf, map< int, bool > &visited, bool &isBipartite)

  Zmienna realizujaca algorytm przeszukania w glab (DFS, ang.Depth First Search).
  Metoda badania grafu oparta na algorytmie DFS polega na badaniu wszystkich krawedzi wychodzacych z podanego wierzcholka. W tym przypadku zawsze zaczynamy od wierzcholka 0. Po zbadaniu wszystkich krawedzi wychodzacych z danego wierzcholka algorytm powraca do wierzcholka, z ktorego dany wierzcholek zostal odwiedzony. Algorytm DFS jest algorytmem rekurencyjnym.
- void printBipartiteGroups (map< int, int > &color, string &out, map< int, list< int > > &graf)

  Funkcja ma zapisac do pliku wyjsciowego informacje ze graf jest dwudzielny, wypisac liste sasiedztwa dla grafu oraz do ktorych zbiorow naleza konkretne wierzcholki.
- void printNOBipartiteGroups (string &out)
- void wyp (map< int, list< int > > &graf, bool &isBipartite, string &out)

  Funkcja wywoluje funkcje DFS() oraz funkcje printBipartiteGroups() i printNOBipartiteGroups(). W funkcji tworzone sa dwie istotne mapy:
- void addedge (int a, int b, map< int, list< int > > &graf)

  Linia kodu: graf[b].push_back(a); jest obowiazkowa w przypadku grafow nieskierowanych.
- void read (string txt, string out)

  Funkcja otwiera plik, sprawdza poprawnosc argumentow.

3.1.1 Dokumentacja funkcji

3.1.1.1 addedge()

```
void addedge (
            int a,
            int b,
            map< int, list< int > > & graf )
```

Linia kodu: graf[b].push_back(a); jest obowiazkowa w przypadku grafow nieskierowanych.

Funkcja dodaje wierzcholki z pliku wejsciowego do mapy list o nazwie "graf".

3.1.1.2 DFS()

```
void DFS (
            int a,
            map< int, int > & color,
            map< int, list< int > > & graf,
            map< int, bool > & visited,
            bool & isBipartite )
```

Zmienna realizujaca algorytm przeszukania w glab (DFS, ang.Depth First Search).
Metoda badania grafu oparta na algorytmie DFS polega na badaniu wszystkich krawedzi wychodzacych z podanego wierzcholka. W tym przypadku zawsze zaczynamy od wierzcholka 0. Po zbadaniu wszystkich krawedzi wychodzacych z danego wierzcholka algorytm powraca do wierzcholka, z ktorego dany wierzcholek zostal odwiedzony. Algorytm DFS jest algorytmem rekurencyjnym.

Parametry

graf	Mapa list zawierajaca zadany graf.
visited	Mapa zawierajaca informacje czy konkretny wierzcholek zostal juz odwiedzony. Jako klucz przyjmuje wartosci typu 'int', ktore okreslaja wierzcholki zas jako wartosci przechowywane pod danym kluczem zmienne typu bool.
color	Mapa przechowujaca informacje o kolorze jak zostal przydzielony kazdemu wierzcholkowi.
isBipartite	Zmienna logiczna przechowujaca informacje czy zadany graf nieskierowany jest grafem dwudzielnym.
a	Wierzcholek

Zwraca

Zwraca true w zmiennej isBipartite jezeli graf jest dwudzielny. W przeciwnym razie zwraca wartosc false.

3.1.1.3 printBipartiteGroups()

```
void printBipartiteGroups (
            map< int, int > & color,
            string & out,
            map< int, list< int > > & graf )
```

Funkcja ma zapisac do pliku wyjsciowego informacje ze graf jest dwudzielny, wypisac liste sasiedztwa dla grafu oraz do ktorych zbiorow naleza konkretne wierzcholki.

Parametry

color	Mapa przechowujaca informacje o kolorze jak zostal przydzielony kazdemu wierzcholkowi podczas dzialania algorytmu DFS.
out	Zmienna typu string zawierajaca nazwe pliku wyjsciowego.
graf	Mapa list zawierajaca zadany graf.

3.1.1.4 printNOBipartiteGroups()

```
void printNOBipartiteGroups (
            string & out )
```

3.1.1.5 read()

```
void read (
            string txt,
            string out )
```

Funkcja otwiera plik, sprawdza poprawnosc argumentow.

Parametry

txt	Zmienna zawierajaca nazwe pliku ktory trzeba otworzyc
out	Zmienna zawierajaca nazwe pliku do ktorego trzeba zapisac wynik koncowy

3.1.1.6 wyp()

```
void wyp (
            map< int, list< int > > & graf,
            bool & isBipartite,
            string & out )
```

Funkcja wywoluje funkcje DFS() oraz funkcje printBipartiteGroups() i printNOBipartiteGroups(). W funkcji tworzone sa dwie istotne mapy:

1. map<int, int>color - przechowuje informacje o kolorze jak zostal przydzielony kazdemu wierzcholkowi podczas dzialania algorytmu DFS.
2. map<int, bool>visited - zawiera informacje czy konkretny wierzcholek zostal juz odwiedzony. Jako klucz przyjmuje wartosci typu 'int', ktore okreslaja wierzcholki zas jako wartosci przechowywane pod danym kluczem zmienne typu bool.

Parametry

graf	Mapa list zawierajaca i przechowujaca zadany graf.
isBipartite	Zmienna logiczna przechowujaca informacje czy zadany graf nieskierowany jest grafem dwudzielnym.
out	Zmienna zawierajaca nazwe pliku do ktorego trzeba zapisac wynik koncowy

3.2 Dokumentacja pliku header.h

```
#include <iostream>
#include <fstream>
#include <map>
#include <list>
#include <vector>
```

Funkcje

- void DFS (int a, map< int, int > &color, map< int, list< int > > &graf, map< int, bool > &visited, bool &isBipartite)

  Zmienna realizujaca algorytm przeszukania w glab (DFS, ang.Depth First Search).
  Metoda badania grafu oparta na algorytmie DFS polega na badaniu wszystkich krawedzi wychodzacych z podanego wierzcholka. W tym przypadku zawsze zaczynamy od wierzcholka 0. Po zbadaniu wszystkich krawedzi wychodzacych z danego wierzcholka algorytm powraca do wierzcholka, z ktorego dany wierzcholek zostal odwiedzony. Algorytm DFS jest algorytmem rekurencyjnym.
- void printBipartiteGroups (map< int, int > &color, string &out, map< int, list< int > > &graf)

  Funkcja ma zapisac do pliku wyjsciowego informacje ze graf jest dwudzielny, wypisac liste sasiedztwa dla grafu oraz do ktorych zbiorow naleza konkretne wierzcholki.
- void printNOBipartiteGroups (map< int, int > &color, string &out, map< int, list< int > > &graf)

  Funkcja ma za zadanie zapisac do pliku wyjsciowego informacje ze gra nieskierowany nie jest grafem dwudzielnym.
- void wyp (map< int, list< int > > &graf, bool &isBipartite, string &out)

  Funkcja wywoluje funkcje DFS() oraz funkcje printBipartiteGroups() i printNOBipartiteGroups(). W funkcji tworzone sa dwie istotne mapy:
- void addedge (int a, int b, map< int, list< int > > &graf)

  Funkcja dodaje wierzcholki z pliku wejsciowego do mapy list o nazwie "graf".
- void read (string txt, string out)

  Funkcja otwiera plik, sprawdza poprawnosc argumentow.

3.2.1 Dokumentacja funkcji

3.2.1.1 addedge()

```
void addedge (
            int a,
            int b,
            map< int, list< int > > & graf )
```

Funkcja dodaje wierzcholki z pliku wejsciowego do mapy list o nazwie "graf".

Parametry

a	Pierwszy wierzcholek z linii
b	Drugi wierzcholek z linii
graf	Mapa list zawierajaca zadany graf. Zmienna jest dalej przekazywana.

Funkcja dodaje wierzcholki z pliku wejsciowego do mapy list o nazwie "graf".

3.2.1.2 DFS()

```
void DFS (
            int a,
            map< int, int > & color,
            map< int, list< int > > & graf,
            map< int, bool > & visited,
            bool & isBipartite )
```

Zmienna realizujaca algorytm przeszukania w glab (DFS, ang.Depth First Search).
Metoda badania grafu oparta na algorytmie DFS polega na badaniu wszystkich krawedzi wychodzacych z podanego wierzcholka. W tym przypadku zawsze zaczynamy od wierzcholka 0. Po zbadaniu wszystkich krawedzi wychodzacych z danego wierzcholka algorytm powraca do wierzcholka, z ktorego dany wierzcholek zostal odwiedzony. Algorytm DFS jest algorytmem rekurencyjnym.

Parametry

graf	Mapa list zawierajaca zadany graf.
visited	Mapa zawierajaca informacje czy konkretny wierzcholek zostal juz odwiedzony. Jako klucz przyjmuje wartosci typu 'int', ktore okreslaja wierzcholki zas jako wartosci przechowywane pod danym kluczem zmienne typu bool.
color	Mapa przechowujaca informacje o kolorze jak zostal przydzielony kazdemu wierzcholkowi.
isBipartite	Zmienna logiczna przechowujaca informacje czy zadany graf nieskierowany jest grafem dwudzielnym.
a	Wierzcholek

Zwraca

Zwraca true w zmiennej isBipartite jezeli graf jest dwudzielny. W przeciwnym razie zwraca wartosc false.

3.2.1.3 printBipartiteGroups()

```
void printBipartiteGroups (
            map< int, int > & color,
            string & out,
            map< int, list< int > > & graf )
```

Funkcja ma zapisac do pliku wyjsciowego informacje ze graf jest dwudzielny, wypisac liste sasiedztwa dla grafu oraz do ktorych zbiorow naleza konkretne wierzcholki.

Parametry

color	Mapa przechowujaca informacje o kolorze jak zostal przydzielony kazdemu wierzcholkowi podczas dzialania algorytmu DFS.
out	Zmienna typu string zawierajaca nazwe pliku wyjsciowego.
graf	Mapa list zawierajaca zadany graf.

3.2.1.4 printNOBipartiteGroups()

```
void printNOBipartiteGroups (
            map< int, int > & color,
            string & out,
            map< int, list< int > > & graf )
```

Funkcja ma za zadanie zapisac do pliku wyjsciowego informacje ze gra nieskierowany nie jest grafem dwudzielnym.

Parametry

out	Zmienna typu string zawierajaca nazwe pliku wyjsciowego.

3.2.1.5 read()

```
void read (
            string txt,
            string out )
```

Funkcja otwiera plik, sprawdza poprawnosc argumentow.

Parametry

txt	Zmienna zawierajaca nazwe pliku ktory trzeba otworzyc
out	Zmienna zawierajaca nazwe pliku do ktorego trzeba zapisac wynik koncowy

3.2.1.6 wyp()

```
void wyp (
            map< int, list< int > > & graf,
            bool & isBipartite,
            string & out )
```

Funkcja wywoluje funkcje DFS() oraz funkcje printBipartiteGroups() i printNOBipartiteGroups(). W funkcji tworzone sa dwie istotne mapy:

1. map<int, int>color - przechowuje informacje o kolorze jak zostal przydzielony kazdemu wierzcholkowi podczas dzialania algorytmu DFS.
2. map<int, bool>visited - zawiera informacje czy konkretny wierzcholek zostal juz odwiedzony. Jako klucz przyjmuje wartosci typu 'int', ktore okreslaja wierzcholki zas jako wartosci przechowywane pod danym kluczem zmienne typu bool.

Parametry

graf	Mapa list zawierajaca i przechowujaca zadany graf.
isBipartite	Zmienna logiczna przechowujaca informacje czy zadany graf nieskierowany jest grafem dwudzielnym.
out	Zmienna zawierajaca nazwe pliku do ktorego trzeba zapisac wynik koncowy

3.3 header.h

Idź do dokumentacji tego pliku.

```
00001 #pragma once
00002 #include <iostream>
00003 #include <fstream>
00004 #include <map>
00005 #include <list>
00006 #include <vector>
00007 using namespace std;
00008
00024 void DFS(int a, map<int, int>& color, map<int, list<int»& graf, map<int, bool>& visited, bool&
      isBipartite);
00025
00034 void printBipartiteGroups(map<int, int>& color, string& out, map<int, list<int»& graf);
00035
00043 void printNOBipartiteGroups(map<int, int>& color, string& out, map<int, list<int»& graf);
00044
00057 void wyp(map<int, list<int»& graf, bool& isBipartite, string& out);
00058
00067 void addedge(int a, int b, map<int,list<int»& graf);
00068
00075 void read(string txt,string out);
```

3.4 Dokumentacja pliku main.cpp

```
#include <iostream>
#include <fstream>
#include <stdio.h>
#include <string>
#include <vector>
#include <list>
#include <map>
#include <unordered_set>
#include "header.h"
```

Funkcje

- int main (int argc, char ∗argv[ ])

  Funkcja main() przyjmuje lacznie 4 argumenty (dwa przelaczniki wraz z ich wartosciami) wpisane podczas uruchamiania programu z linii polecen. Najpierw nastepuje odczyt wyzej wymienionych przelacznikow oraz ich wartosci, a nastepnie sprawdzana jest porpawnosc ich wprowadzenia. W przypadku bledu, funkcja wypisuje na ekranie instrukcje dla uzytkownika wraz z przykladem porpawnego uzycia przelacznikow. Jezeli wszystko zostalo poprawnie wpisane, funkcja main() wywoluje funkcje read().

3.4.1 Dokumentacja funkcji

3.4.1.1 main()

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

Funkcja main() przyjmuje lacznie 4 argumenty (dwa przelaczniki wraz z ich wartosciami) wpisane podczas uruchamiania programu z linii polecen. Najpierw nastepuje odczyt wyzej wymienionych przelacznikow oraz ich wartosci, a nastepnie sprawdzana jest porpawnosc ich wprowadzenia. W przypadku bledu, funkcja wypisuje na ekranie instrukcje dla uzytkownika wraz z przykladem porpawnego uzycia przelacznikow. Jezeli wszystko zostalo poprawnie wpisane, funkcja main() wywoluje funkcje read().