Politechnika Śląska
Wydział Informatyki, Elektroniki i Informatyki

Programowanie Komputerów 3

Sieć komputerowa

Autor	Krzysztof Małek
Prowadzący	dr. Inż. Jolanta Kawulok
Rok akademicki	2020/2021
Kierunek	informatyka
Rodzaj studiów	SSI
Semestr	3
Termin laboratorium	środa 11:15 | czwartek 8:30
Sekcja	31
Termin oddania sprawozdania	2020-11-09

Zawartość

1. Treść zadania

Napisać program tworzący minimalne połączenia sieci w bloku (minimalne drzewa rozpinające), tak aby każdy z mieszkańców był przyłączony do sieci z wykorzystaniem jak najmniejszej ilości kabli. Program wykorzystuje dwa pliki wejściowe: jeden zawierający dane na temat każdego z mieszkańców oraz jego współrzędne, drugi niemożliwe do zrealizowania połączenia. Połączenia miedzy mieszkańcami zostaną zapisane w pliku wraz z kolejnością i sumą aktualnych odcinków. Program uruchamiany jest z konsoli linii poleceń z wykorzystaniem następujących przełączników:

-coor plik wejściowy z danymi mieszkańców
-tab plik wejściowy z niemożliwymi do zrealizowania połączeniami
-out plik wyjściowy z listą odcinków

2. Analiza zadania

Zagadnienie przedstawia wykorzystanie klas obiektów i algorytmu Kruskala w problemie wyznaczania minimalnych drzew rozpinających.

2.1 Struktury danych

W programie wykorzystano ***klasy*** do łatwego przechowywania danych o każdym z mieszkańców oraz jego współrzędnych. Wykorzystano także ***listę jednokierunkową*** do przechowywania wskaźników na obiekty klasy mieszkańców w celu szybkiego liniowego przejścia przez nie i wyznaczeniu możliwych do zrealizowania połączeń miedzy nimi. Do przechowania niemożliwych do zrealizowania połączeń wykorzystano ***vector par intów*** w celu szybkiego dostania się do określonego miejsca w vectorze i nie błądzeniu po całym jego zakresie.

2.2 Algorytmy

Program tworzy możliwe do zrealizowania połączenia przechodząc przez listę w czasie $O(\frac{n(n+1)}{2})$. Natomiast wyznaczenie najmniejszych odcinków w czasie wraz z posortowaniem $O(n \log n)$

3. Specyfikacja zewnętrzna

Program jest uruchamiany z linii poleceń. Do programu należy przekazać nazwy plików wejściowych i pliku wyjściowego po odpowiednich przełącznikach.

Np.

*.exe –coor mieszkańcy.txt –tab blok_połączenia.txt –out wyjściowy.txt

*.exe –out wyjściowy.txt –tab blok_połączenia.txt –coor mieszkańcy.txt

Złe wykorzystanie przełączników lub brak któregoś z nich sygnalizowane jest komunikatem o błędzie i wyświetleniem pomocy.
"Not enough arguments to run program, it must look like that: -coor name.txt -tab name.txt -out name.txt"

4. Specyfikacja wewnętrzna

4.1 Ogólna struktura programu

W funkcji głównej wywoływana jest funkcja ***checkArguments*** która sprawdza czy program został poprawnie wywołany, jeśli nie zwraca komunikat o błędzie. Następnie wywoływana jest funkcja ***readCoorName***, która wczytuje plik z danymi mieszkańców do bufora, a z niego do listy jednokierunkowej. Następnie ***readTabName***, która wczytuje plik z niemożliwymi połączeniami do bufora, a z niego do *vectora par intów* i sortuje je rosnąco. Następnie wywołana jest funkcja ***fillQueque***, która zapełnia *vector par int par intów* kolejno odległością między mieszkaniami, oraz indeksem pierwszego i drugiego mieszkania, z wykluczeniem połączeń nie możliwych do zrealizowania z vectora *tabVector***.** Następnie wywołuje się funkcja ***makeTheWeeb***, która przechodząc przez kolejkę Q możliwymi do zrealizowania połączeniami, wpisuje do pliku kolejno zrealizowane odcinki wraz z ich kolejnością i sumą długości zrealizowanych połączeń. Na koniec uruchamiana jest funkcja ***eraseList*** która czyści wcześniej stworzoną listę na wskaźniki obiektów *coordinations.*

4.2 Szczegółowy opis typów i funkcji

Szczegółowy opis typów i funkcji zawarty jest w załączniku.

5. Testowanie

Program został przetestowany na różnego rodzaju plikach. Pliki zawierające kombinacje różnych ułożeń współrzędnych i losowo wybranych danych. Pliki puste powodują zgłoszenie błędu o podaniu pustych plików wejściowych. Natomiast podanie nieistniejącego pliku wyjściowego nie powoduje powstawania błędu a jego utworzenie. Program został sprawdzony pod kątem wycieków pamięci.

6 Wnioski

Program Sieć komputerowa jest prostym intuicyjnie programem, który jednak wymaga odpowiedniego wykorzystania struktur danych i ich możliwości, aby program działał w miarę dobrej optymalizacji czasowej i nie powodował błędów przy postawaniu minimalnego drzewa rozpinającego. Najwięcej problemów stwarzało tworzenie zbiorów, które zapobiegały powstawaniu pętli. Należało dobrze zrozumieć gdzie jaki indeks mieszkania należy się pojawić aby nie powstało za dużo krawędzi w grafie.

Literatura

[1] https://stackoverflow.com/
[2] http://www.cplusplus.com/
[3] https://www.geeksforgeeks.org/

Załącznik
Szczegółowy opis typów i funkcji

Sieć komputerowa

Wygenerowano przez Doxygen 1.8.17

Rozdział 1

Indeks hierarchiczny

1.1 Hierarchia klas

Ta lista dziedziczenia posortowana jest z grubsza, choć nie całkowicie, alfabetycznie:

Rozdział 2

Indeks klas

2.1 Lista klas

Tutaj znajdują się klasy, struktury, unie i interfejsy wraz z ich krótkimi opisami:

Rozdział 3

Indeks plików

3.1 Lista plików

Tutaj znajduje się lista wszystkich udokumentowanych plików z ich krótkimi opisami:

Rozdział 4

Dokumentacja klas

4.1 Dokumentacja klasy coordinates

`#include <structs.h>`

Diagram dziedziczenia dla coordinates

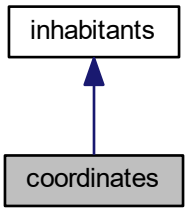
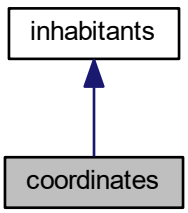


Diagram współpracy dla coordinates:



Metody publiczne

- int sendV ()
- int sendW ()

4.1.1 Opis szczegółowy

Klasa przechowująca współrzędne mieszkańców - pochodna klasy inhabitants

4.1.2 Dokumentacja funkcji składowych

4.1.2.1 sendV()

```
int coordinates::sendV ( )  [inline]
```

< metoda zwracająca współrzędną v

4.1.2.2 sendW()

```
int coordinates::sendW ( )  [inline]
```

< metoda zwracająca współrzędną w

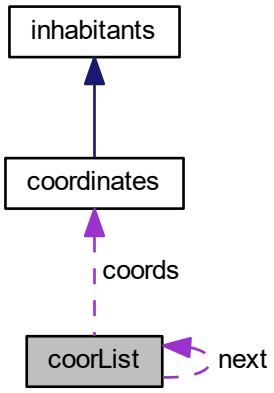
Dokumentacja dla tej klasy została wygenerowana z pliku:

- structs.h

4.2 Dokumentacja struktury coorList

```
#include <structs.h>
```

Diagram współpracy dla coorList:



Atrybuty publiczne

- coordinates * coords

  wskaźnik na obiekt coordinations
- coorList * next

  wskaźnik na następny element listy

4.2.1 Opis szczegółowy

Lista przechowujaca wskaźniki na obiekty coordinations

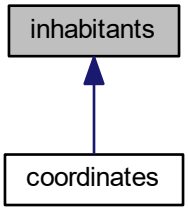
Dokumentacja dla tej struktury została wygenerowana z pliku:

- structs.h

4.3 Dokumentacja klasy inhabitants

```
#include <structs.h>
```

Diagram dziedziczenia dla inhabitants



4.3.1 Opis szczegółowy

Klasa przechowująca index, oraz nazwisko mieszkańców

Dokumentacja dla tej klasy została wygenerowana z pliku:

- structs.h

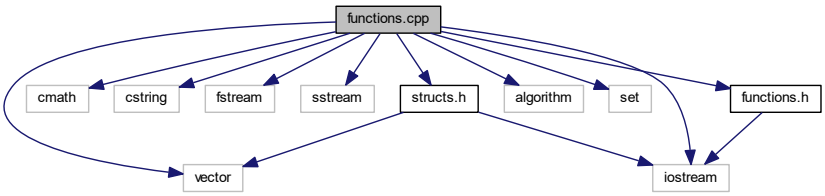
Rozdział 5

Dokumentacja plików

5.1 Dokumentacja pliku functions.cpp

```
#include <iostream>
#include <cmath>
#include <cstring>
#include <fstream>
#include <sstream>
#include <vector>
#include <algorithm>
#include <set>
#include "structs.h"
#include "functions.h"
```

Wykres zależności załączania dla functions.cpp:



Funkcje

- int checkArguments (int argc, char ∗∗&argv, std::string &coorName, std::string &tabName, std::string &out↩ Name)
- void readCoorName (const std::string &coorName, coorList ∗&head)
- void addToCoorList (coorList ∗&head, int _index, std::string _surname, int _v, int _w)
- void readTabName (const std::string &tabName, std::vector< std::pair< int, int >> &tabVector)
- void addToTabVector (std::vector< std::pair< int, int >> &tabVector, int _x, int _y)
- void fillQueque (std::vector< std::pair< int, std::pair< int, int > > > &Q, coorList ∗cHead, std::vector< std↩ ::pair< int, int >> tabVector)
- int howFar (coordinates ∗mainElement, coordinates ∗secondaryElement)
- void makeTheWeeb (std::vector< std::set< int > > &sets, std::vector< std::pair< int, std::pair< int, int > > > &Q, const std::string &outName)
- void eraseList (coorList ∗cHead)

5.1.1 Dokumentacja funkcji

5.1.1.1 addToCoorList()

```
void addToCoorList (
            coorList *& head,
            int _index,
            std::string _surname,
            int _v,
            int _w )
```

Funkcja tworząca obiekt coordinates i dodajaca go do listy

Parametry

head	głowa listy
_index	dana index do utworzenia obiektu
_surname	nazwisko
_v	współrzędna v
_w	współrzędna w

Zwraca

void

5.1.1.2 addToTabVector()

```
void addToTabVector (
            std::vector< std::pair< int, int >> & tabVector,
            int _x,
            int _y )
```

Funkcja dodająca element do vectora z niemożliwymi połączeniami

Parametry

tabVector	vector par intów
_x	mieszkanie 1
_y	mieszkanie 2

Zwraca

void

5.1.1.3 checkArguments()

```
int checkArguments (
            int argc,
            char **& argv,
            std::string & coorName,
            std::string & tabName,
            std::string & outName )
```

Funkcja sprawdza argumenty podane w konsoli przy uruchomieniu programu

Parametry

argc	ilość wyrazów pobrana z konsoli
argv	wyrazy pobrane z konsoli
coorName	nazwa pliku ze współrzędnymi
tabName	nazwa pliku z niemożliwymi połączeniami
outName	nazwa pliku wyjściowego

Zwraca

int

5.1.1.4 eraseList()

```
void eraseList (
            coorList * cHead )
```

Funkcja zwalniająca pamięć z listy wskaźników na obiekty

Parametry

cHead	wskaźnik na kolejne elementy listy

Zwraca

void

5.1.1.5 fillQueque()

```
void fillQueque (
            std::vector< std::pair< int, std::pair< int, int > > > & Q,
            coorList * cHead,
            std::vector< std::pair< int, int >> tabVector )
```

Funkcja zapełniająca i sortująca kolejkę Q wykorzystywaną przez algorytm kruskala

Parametry

Q	vector par int par intów reprezentujący odległość i mieszkania między którymi możemy zrobić połączenie
cHead	głowa listy na kolejne obiekty coordinates

Zwraca

void

5.1.1.6 howFar()

```
int howFar (
            coordinates * mainElement,
            coordinates * secondaryElement )
```

Funkcja licząca odległość między mieszkaniami idąc po prostych

Parametry

mainElement	wskaźnik na obiekt 1 mieszkania
secondaryElement	wskaźnik na obiekt 2 mieszkania

Zwraca

int

5.1.1.7 makeTheWeeb()

```
void makeTheWeeb (
            std::vector< std::set< int > > & sets,
            std::vector< std::pair< int, std::pair< int, int > > > & Q,
            const std::string & outName )
```

Funkcja wpisująca do pliku kolejne połączenia między mieszkaniami wraz z sumaryczną długością kabli przy każdym nowym połączeniu

Parametry

sets	vector zbiorów mieszkań dzięki którym eliminujemy pętle
Q	vector par int par intów reprezentujący odległość i mieszkania między którymi możemy zrobić połączenie
outName	nazwa pliku wyjściowego

Zwraca

void

5.1.1.8 readCoorName()

```
void readCoorName (
            const std::string & coorName,
            coorList *& head )
```

Funkcja wczytująca plik ze współrzędnymi do listy

Parametry

coorName	nazwa pliku
head	głowa listy na współrzędne mieszkańców (początkowo nullptr)

Zwraca

void

5.1.1.9 readTabName()

```
void readTabName (
            const std::string & tabName,
            std::vector< std::pair< int, int >> & tabVector )
```

Fubnkcja wczytująca plik z niemożliwymi połączeniami wpisująca je do vectora par intów i sortująca go malejąca

Parametry

tabName	nazwa pliku z niemożliwymi połączeniami
tabVector	vector z parą intów niemożliwych do połączenia mieszkań

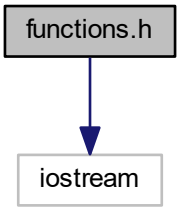
Zwraca

void

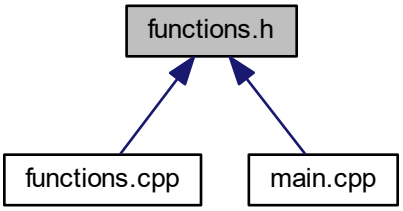
5.2 Dokumentacja pliku functions.h

```
#include <iostream>
```

Wykres zależności załączania dla functions.h:



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



Funkcje

- int checkArguments (int argc, char **&argv, std::string &coorName, std::string &tabName, std::string &out↩ Name)
- void readCoorName (const std::string &coorName, coorList *&head)
- void addToCoorList (coorList *&head, int _index, std::string _surname, int _v, int _w)
- void readTabName (const std::string &tabName, std::vector< std::pair< int, int >> &tabVector)
- void addToTabVector (std::vector< std::pair< int, int >> &tabVector, int _x, int _y)
- void fillQueque (std::vector< std::pair< int, std::pair< int, int > > > &Q, coorList *cHead, std::vector< std↩ ::pair< int, int >> tabVector)
- int howFar (coordinates *mainElement, coordinates *secondaryElement)
- void makeTheWeeb (std::vector< std::set< int > > &sets, std::vector< std::pair< int, std::pair< int, int > > > &Q, const std::string &outName)
- void eraseList (coorList *cHead)

5.2.1 Dokumentacja funkcji

5.2.1.1 addToCoorList()

```
void addToCoorList (
            coorList *& head,
            int _index,
            std::string _surname,
            int _v,
            int _w )
```

Funkcja tworząca obiekt coordinates i dodajaca go do listy

Parametry

head	głowa listy
_index	dana index do utworzenia obiektu
_surname	nazwisko
_v	współrzędna v
_w	współrzędna w

Zwraca

void

5.2.1.2 addToTabVector()

```
void addToTabVector (
            std::vector< std::pair< int, int >> & tabVector,
            int _x,
            int _y )
```

Funkcja dodająca element do vectora z niemożliwymi połączeniami

Parametry

tabVector	vector par intów
_x	mieszkanie 1
_y	mieszkanie 2

Zwraca

void

5.2.1.3 checkArguments()

```
int checkArguments (
            int argc,
```

```
            char **& argv,
            std::string & coorName,
            std::string & tabName,
            std::string & outName )
```

Funkcja sprawdza argumenty podane w konsoli przy uruchomieniu programu

Parametry

argc	ilość wyrazów pobrana z konsoli
argv	wyrazy pobrane z konsoli
coorName	nazwa pliku ze współrzędnymi
tabName	nazwa pliku z niemożliwymi połączeniami
outName	nazwa pliku wyjściowego

Zwraca

int

5.2.1.4 eraseList()

```
void eraseList (
            coorList * cHead )
```

Funkcja zwalniająca pamięć z listy wskaźników na obiekty

Parametry

cHead	wskaźnik na kolejne elementy listy

Zwraca

void

5.2.1.5 fillQueque()

```
void fillQueque (
            std::vector< std::pair< int, std::pair< int, int > > > & Q,
            coorList * cHead,
            std::vector< std::pair< int, int >> tabVector )
```

Funkcja zapełniająca i sortująca kolejkę Q wykorzystywaną przez algorytm kruskala

Parametry

Q	vector par int par intów reprezentujący odległość i mieszkania między którymi możemy zrobić połączenie
cHead	głowa listy na kolejne obiekty coordinates

Zwraca

void

5.2.1.6 howFar()

```
int howFar (
            coordinates * mainElement,
            coordinates * secondaryElement )
```

Funkcja licząca odległość między mieszkaniami idąc po prostych

Parametry

mainElement	wskaźnik na obiekt 1 mieszkania
secondaryElement	wskaźnik na obiekt 2 mieszkania

Zwraca

int

5.2.1.7 makeTheWeeb()

```
void makeTheWeeb (
            std::vector< std::set< int > > & sets,
            std::vector< std::pair< int, std::pair< int, int > > > & Q,
            const std::string & outName )
```

Funkcja wpisująca do pliku kolejne połączenia między mieszkaniami wraz z sumaryczną długością kabli przy każdym nowym połączeniu

Parametry

sets	vector zbiorów mieszkań dzięki którym eliminujemy pętle
Q	vector par int par intów reprezentujący odległość i mieszkania między którymi możemy zrobić połączenie
outName	nazwa pliku wyjściowego

Zwraca

void

5.2.1.8 readCoorName()

```
void readCoorName (
            const std::string & coorName,
            coorList *& head )
```

Funkcja wczytująca plik ze współrzędnymi do listy

Parametry

coorName	nazwa pliku
head	głowa listy na współrzędne mieszkańców (początkowo nullptr)

Zwraca

void

5.2.1.9 readTabName()

```
void readTabName (
            const std::string & tabName,
            std::vector< std::pair< int, int >> & tabVector )
```

Fubnkcja wczytująca plik z niemożliwymi połączeniami wpisująca je do vectora par intów i sortująca go malejąca

Parametry

tabName	nazwa pliku z niemożliwymi połączeniami
tabVector	vector z parą intów niemożliwych do połączenia mieszkań

Zwraca

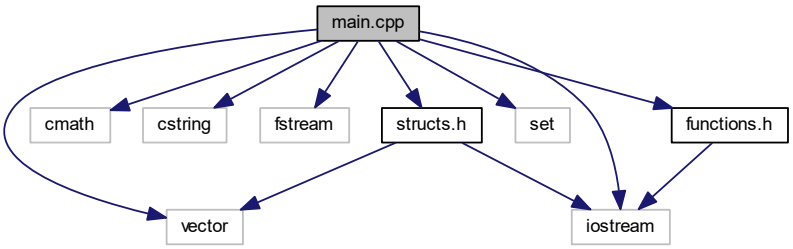
void

5.3 Dokumentacja pliku main.cpp

```
#include <iostream>
#include <cmath>
#include <cstring>
```

```
#include <fstream>
#include <vector>
#include <set>
#include "structs.h"
#include "functions.h"
```

Wykres zależności załączania dla main.cpp:



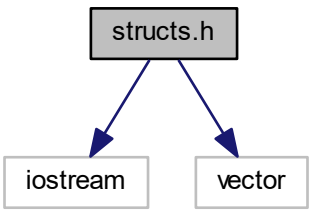
Funkcje

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

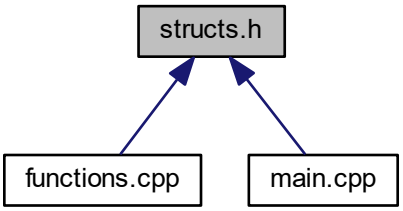
5.4 Dokumentacja pliku structs.h

```
#include <iostream>
#include <vector>
```

Wykres zależności załączania dla structs.h:



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



Komponenty

- class inhabitants
- class coordinates
- struct coorList

Indeks