# Tablica asocjacyjna

Wygenerowano przez Doxygen 1.8.4

Wt, 13 maj 2014 22:32:33

# Spis treści

# 1 Graf nieskierowany z wagami

## 1.1 Opis programu

Program definiuje strukturę danych do reprezentacji obiektów, pomiędzy którymi występują różne zależności. Graf składa się z w liczby wierzchołków oraz k liczby krawędzi - w przypadku implementowanego grafu (nieskierowanego) wierzchołki można łączyć w obie strony.

## 1.2 Autor

Program wykonala: Agnieszka Wisniewska, nr albumu: 200 466

# 2 Indeks klas

## 2.1 Lista klas

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



# 3 Indeks plików

## 3.1 Lista plików

Tutaj znajduje się lista wszystkich plików z ich krótkimi opisami:

# 4 Dokumentacja klas

## 4.1 Dokumentacja klasy Benchmark

Definicja klasy Benchmark.

`#include <benchmark.hh>`

### Metody publiczne

- double benchmark (int noIteration, Implementation Type)

  *Funkcja mierzaca czas wykonywania algorytmu przeszukiwania grafu.*
- void SampleGraph (const unsigned int VertCount)

  *Funkcja wypełniając graf testowymi danymi.*
- unsigned int GetEdges ()

  *Funkcja pobierająca ilość krawędzi w grafie.*

### Metody prywatne

- void calculate (Implementation Type)

### Atrybuty prywatne

- Graph< int > benchGraph
- unsigned int Edges

### 4.1.1 Opis szczegółowy

Definicja klasy Benchmark.

Klasa służy do pomiaru czasu wykonywania algorytmu dla wybranych implementacji.

### 4.1.2 Dokumentacja funkcji składowych

#### 4.1.2.1 double Benchmark::benchmark ( int *noIteration,* Implementation *Type* )

Funkcja mierzaca czas wykonywania algorytmu przeszukiwania grafu.

**Parametry**

| | |
|---|---|
| *noIteration* | liczba powtórzeń algorytmu |
| *Type* | rodzaj przeszukiwania |

#### 4.1.2.2 void Benchmark::calculate ( Implementation *Type* ) [private]

#### 4.1.2.3 unsigned int Benchmark::GetEdges ( ) [inline]

Funkcja pobierająca ilość krawędzi w grafie.

**Zwraca**

liczba krawędzi w grafie

#### 4.1.2.4 void Benchmark::SampleGraph ( const unsigned int *VertCount* )

Funkcja wypełniając graf testowymi danymi.

**Parametry**

| | |
|---|---|
| *VertCount* | ilość wierzchołków w grafie |

### 4.1.3 Dokumentacja atrybutów składowych

#### 4.1.3.1 Graph<int> Benchmark::benchGraph [private]

#### 4.1.3.2 unsigned int Benchmark::Edges [private]

Dokumentacja dla tej klasy została wygenerowana z plików:

- benchmark.hh
- benchmark.cpp

## 4.2 Dokumentacja struktury Graph< Type >::Edge

`#include <graph.hh>`

**Metody publiczne**

- Edge (const Type newEnd, const int newWeight)

**Atrybuty publiczne**

- Type SecEnd
- int Weight

### 4.2.1 Dokumentacja konstruktora i destruktora

#### 4.2.1.1 template<typename Type> Graph< Type >::Edge::Edge ( const Type *newEnd,* const int *newWeight* ) [inline]

### 4.2.2 Dokumentacja atrybutów składowych

#### 4.2.2.1 template<typename Type> Type Graph< Type >::Edge::SecEnd

#### 4.2.2.2 template<typename Type> int Graph< Type >::Edge::Weight

Dokumentacja dla tej struktury została wygenerowana z pliku:

- graph.hh

## 4.3 Dokumentacja szablonu klasy Graph< Type >

class Graph Jest to klasa definiująca graf nieskierowany z wagą pozwalająca na wykonywaniu wybranych funkcji.

`#include <graph.hh>`

**Komponenty**

- struct Edge

**Typy publiczne**

- typedef std::vector< Edge > EdgeS

**Metody publiczne**

- void AddVert (const Type &vert)

  *Funkcja dodająca nowy wierzchołek.*
- void RemoveVert (const Type &vert)

  *Funkcja usuwająca wybrany wierzchołek.*
- void AddEdge (const Type &vert1, const Type &vert2, const int Weight=0)

  *Funkcja dodająca nową krawędź*
- void RemoveEdge (const Type &vert1, const Type &vert2)

  *Funkcja usuwająca daną krawędź*
- bool IfConnected (const Type &vert1, const Type &vert2)

  *Funkcja sprawdzająca, czy wierzchołki są ze sobą bezpośrednio połączone.*
- EdgeS Neighbors (const Type &vert)

  *Funkcja znajdująca sąsiednie wierzchołki.*

**Atrybuty prywatne**

- std::map< Type, EdgeS > graph

#### 4.3.1 Opis szczegółowy

**template<typename Type>class Graph< Type >**

class Graph Jest to klasa definiująca graf nieskierowany z wagą pozwalająca na wykonywaniu wybranych funkcji.

#### 4.3.2 Dokumentacja składowych definicji typu

##### 4.3.2.1 template<typename Type> typedef std::vector<Edge> Graph< Type >::EdgeS

#### 4.3.3 Dokumentacja funkcji składowych

##### 4.3.3.1 template<typename Type> void Graph< Type >::AddEdge ( const Type & *vert1*, const Type & *vert2*, const int *Weight* = 0 )

Funkcja dodająca nową krawędź

**Parametry**

| | |
|---|---|
| *vert1* | współrzędna pierwszego wierzchołka |
| *vert2* | współrzędna drugiego wierzchołka |
| *Weight* | waga krawędzi |

##### 4.3.3.2 template<typename Type> void Graph< Type >::AddVert ( const Type & *vert* )

Funkcja dodająca nowy wierzchołek.

**Parametry**

| | |
|---|---|
| *vert* | wartość dodawanego wierzchołka |

#### 4.3.3.3 template<typename Type> bool Graph< Type >::IfConnected ( const Type & *vert1,* const Type & *vert2* )

Funkcja sprawdzająca, czy wierzchołki są ze sobą bezpośrednio połączone.

**Parametry**

| | |
|---|---|
| *vert1* | współrzędna pierwszego wierzchołka |
| *vert2* | współrzędna drugiego wierzchołka |

**Zwraca**

true jeśli wierzchołki są połączone
false jeśli wierzchołki nie są połączone

#### 4.3.3.4 template<typename Type> Graph< Type >::EdgeS Graph< Type >::Neighbors ( const Type & *vert* )

Funkcja znajdująca sąsiednie wierzchołki.

**Parametry**

| | |
|---|---|
| *vert* | wierzchołek, którego sąsiadów poszukujemy |

**Zwraca**

wektor wierzchołków sąsiadujących

#### 4.3.3.5 template<typename Type> void Graph< Type >::RemoveEdge ( const Type & *vert1,* const Type & *vert2* )

Funkcja usuwająca daną krawędź

**Parametry**

| | |
|---|---|
| *vert1* | współrzędna pierwszego wierzchołka |
| *vert2* | współrzędna drugiego wierzchołka |

#### 4.3.3.6 template<typename Type> void Graph< Type >::RemoveVert ( const Type & *vert* )

Funkcja usuwająca wybrany wierzchołek.

**Parametry**

| | |
|---|---|
| *vert* | wartość usuwanego wierzchołka |

### 4.3.4 Dokumentacja atrybutów składowych

#### 4.3.4.1 template<typename Type> std::map< Type, EdgeS > Graph< Type >::graph [private]

Dokumentacja dla tej klasy została wygenerowana z pliku:

- graph.hh

# 5 Dokumentacja plików

## 5.1 Dokumentacja pliku benchmark.cpp

Plik zawierający funkcję mierzącą czas wykonywania algorytmu.

```
#include <ctime>
#include "search.hh"
#include "benchmark.hh"
#include <cstdlib>
```

### 5.1.1 Opis szczegółowy

Plik zawierający funkcję mierzącą czas wykonywania algorytmu.

## 5.2 Dokumentacja pliku benchmark.hh

Plik zawiera definicje klasy Benchmark oraz typu Implementation.

```
#include "graph.hh"
```

#### Komponenty

- class Benchmark

  *Definicja klasy Benchmark.*

#### Wyliczenia

- enum Implementation { bfs, dfs }

  *Typ danych przechowujacy rodzaj wyszukiwania.*

### 5.2.1 Opis szczegółowy

Plik zawiera definicje klasy Benchmark oraz typu Implementation.

### 5.2.2 Dokumentacja typów wyliczanych

#### 5.2.2.1 enum Implementation

Typ danych przechowujacy rodzaj wyszukiwania.

**Wartości wyliczeń**

***bfs***

***dfs***

## 5.3 Dokumentacja pliku graph.hh

```
#include <vector>
#include <map>
```

#### Komponenty

- class Graph< Type >

  *class Graph Jest to klasa definiująca graf nieskierowany z wagą pozwalająca na wykonywaniu wybranych funkcji.*
- struct Graph< Type >::Edge

### 5.3.1 Opis szczegółowy

Plik zawierający szablon klasy Graph.

## 5.4 Dokumentacja pliku main.cpp

Plik zawierający główną funkcję programu.

```
#include "graph.hh"
#include "benchmark.hh"
#include <iostream>
#include <cstdlib>
#include <ctime>
```

#### Funkcje

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

  *Główna funkcja programu.*

### 5.4.1 Opis szczegółowy

Plik zawierający główną funkcję programu.

### 5.4.2 Dokumentacja funkcji

#### 5.4.2.1 int main ( int *argc,* char ** *argv* )

Główna funkcja programu.

Pozwala na zmierzenie czasu dla wybranego wyszukiwania.

## 5.5 Dokumentacja pliku mainpage.dox

## 5.6 Dokumentacja pliku search.cpp

Plik zawierający funkcje przeszukujące graf wszerz oraz w głąb.

```
#include "search.hh"
#include <map>
#include <queue>
#include <stack>
```

#### Funkcje

- void DFS (Graph< int > *Ptr)

  *Przeszukiwanie grafu w głąb.*

- void BFS (Graph< int > *Ptr)

  *Przeszukiwanie grafu wszerz.*

#### 5.6.1 Opis szczegółowy

Plik zawierający funkcje przeszukujące graf wszerz oraz w głąb.

#### 5.6.2 Dokumentacja funkcji

##### 5.6.2.1 void BFS ( Graph< int > * *Ptr* )

Przeszukiwanie grafu wszerz.

**Parametry**

| | |
|---|---|
| *Ptr* | wskaźnik na przeszukiwany graf |

##### 5.6.2.2 void DFS ( Graph< int > * *Ptr* )

Przeszukiwanie grafu w głąb.

**Parametry**

| | |
|---|---|
| *Ptr* | wskaźnik na przeszukiwany graf |

### 5.7 Dokumentacja pliku search.hh

Plik zawierający definicje funkcji przeszukujących grafy.

```
#include "graph.hh"
```

**Funkcje**

- void DFS (Graph< int > *Ptr)

  *Przeszukiwanie grafu w głąb.*
- void BFS (Graph< int > *Ptr)

  *Przeszukiwanie grafu wszerz.*

#### 5.7.1 Opis szczegółowy

Plik zawierający definicje funkcji przeszukujących grafy.

#### 5.7.2 Dokumentacja funkcji

##### 5.7.2.1 void BFS ( Graph< int > * *Ptr* )

Przeszukiwanie grafu wszerz.

**Parametry**

| | |
|---|---|
| *Ptr* | wskaźnik na przeszukiwany graf |

#### 5.7.2.2 void DFS ( Graph< int > * *Ptr* )

Przeszukiwanie grafu w głąb.

**Parametry**

| | |
|---|---|
| *Ptr* | wskaźnik na przeszukiwany graf |

## Skorowidz