

Przeszukiwania grafów

Wygenerowano przez Doxygen 1.8.4

Śr, 28 maj 2014 19:44:09

Spis treści

1	Graf nieskierowany z wagami	1
1.1	Opis programu	1
1.2	Autor	1
2	Indeks klas	1
2.1	Lista klas	1
3	Indeks plików	1
3.1	Lista plików	2
4	Dokumentacja klas	2
4.1	Dokumentacja klasy Benchmark	2
4.1.1	Opis szczegółowy	2
4.1.2	Dokumentacja funkcji składowych	3
4.1.3	Dokumentacja atrybutów składowych	3
4.2	Dokumentacja struktury Graph< Type >::Edge	3
4.2.1	Dokumentacja konstruktora i destruktora	4
4.2.2	Dokumentacja atrybutów składowych	4
4.3	Dokumentacja szablonu klasy Graph< Type >	4
4.3.1	Opis szczegółowy	4
4.3.2	Dokumentacja składowych definicji typu	5
4.3.3	Dokumentacja funkcji składowych	5
4.3.4	Dokumentacja atrybutów składowych	6
4.4	Dokumentacja struktury LessF	6
4.4.1	Opis szczegółowy	6
4.4.2	Dokumentacja funkcji składowych	6
4.5	Dokumentacja struktury VertStruct	6
4.5.1	Opis szczegółowy	7
4.5.2	Dokumentacja konstruktora i destruktora	7
4.5.3	Dokumentacja atrybutów składowych	7
5	Dokumentacja plików	7
5.1	Dokumentacja pliku benchmark.cpp	7
5.1.1	Opis szczegółowy	7
5.2	Dokumentacja pliku benchmark.hh	7
5.2.1	Opis szczegółowy	8
5.2.2	Dokumentacja typów wyliczanych	8
5.3	Dokumentacja pliku graph.hh	8
5.3.1	Opis szczegółowy	8
5.4	Dokumentacja pliku main.cpp	8

5.4.1	Opis szczegółowy	8
5.4.2	Dokumentacja funkcji	9
5.5	Dokumentacja pliku mainpage.dox	9
5.6	Dokumentacja pliku search.cpp	9
5.6.1	Opis szczegółowy	9
5.6.2	Dokumentacja funkcji	9
5.7	Dokumentacja pliku search.hh	10
5.7.1	Opis szczegółowy	10
5.7.2	Dokumentacja funkcji	10

Indeks

12

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 wykonała: 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:

Benchmark	
Definicja klasy Benchmark	2
Graph< Type >::Edge	3
Graph< Type >	
Class Graph Jest to klasa definiująca graf nieskierowany z wagą pozwalająca na wykonywaniu wybranych funkcji	4
LessF	
Struktura pomocnicza do użycia przez algorytm A*	6
VertStruct	
Struktura pomocnicza do użycia przez algorytm A*	6

3 Indeks plików

3.1 Lista plików

Tutaj znajduje się lista wszystkich plików z ich krótkimi opisami:

benchmark.cpp	
Plik zawierający funkcję mierzącą czas wykonywania algorytmu	7
benchmark.hh	
Plik zawiera definicje klasy Benchmark oraz typu Implementation	7
graph.hh	8
main.cpp	
Plik zawierający główną funkcję programu	8
search.cpp	
Plik zawierający funkcje przeszukujące graf wszerek oraz w głąb	9
search.hh	
Plik zawierający definicje funkcji przeszukujących grafy	10

4 Dokumentacja klas

4.1 Dokumentacja klasy [Benchmark](#)

Definicja klasy [Benchmark](#).

```
#include <benchmark.hh>
```

Metody publiczne

- double [benchmark](#) (int nolliteration, [Implementation](#) Type)
Funkcja mierząca 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](#)
- unsigned int [Vertices](#)
- unsigned int [Side](#)

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 *nolteration*, Implementation *Type*)

Funkcja mierzaca czas wykonywania algorytmu przeszukiwania grafu.

Parametry

<i>nolteration</i>	liczba powtórzeń algorytmu
<i>Type</i>	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

<i>VertCount</i>	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]

4.1.3.3 unsigned int Benchmark::Side [private]

4.1.3.4 unsigned int Benchmark::Vertices [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 znajdujaca 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

<i>vert1</i>	współrzędna pierwszego wierzchołka
<i>vert2</i>	współrzędna drugiego wierzchołka
<i>Weight</i>	waga krawędzi

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

Funkcja dodająca nowy wierzchołek.

Parametry

<i>vert</i>	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

<i>vert1</i>	współrzędna pierwszego wierzchołka
<i>vert2</i>	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 znajdujący sąsiednie wierzchołki.

Parametry

<i>vert</i>	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

<i>vert1</i>	współrzędna pierwszego wierzchołka
<i>vert2</i>	współrzędna drugiego wierzchołka

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

Funkcja usuwająca wybrany wierzchołek.

Parametry

<i>vert</i>	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](#)

4.4 Dokumentacja struktury LessF

Struktura pomocnicza do użycia przez algorytm A*.

Metody publiczne

- `bool operator() (const VertStruct &Vert1, const VertStruct &Vert2)`

4.4.1 Opis szczegółowy

Struktura pomocnicza do użycia przez algorytm A*.

4.4.2 Dokumentacja funkcji składowych

4.4.2.1 `bool LessF::operator() (const VertStruct & Vert1, const VertStruct & Vert2) [inline]`

Dokumentacja dla tej struktury została wygenerowana z pliku:

- [search.cpp](#)

4.5 Dokumentacja struktury VertStruct

Struktura pomocnicza do użycia przez algorytm A*.

Metody publiczne

- `VertStruct (const int initValue, const int initG, const int initH, const int initF)`

Atrybuty publiczne

- `int Value`
- `int G`
- `int H`
- `int F`

4.5.1 Opis szczegółowy

Struktura pomocnicza do użycia przez algorytm A*.

4.5.2 Dokumentacja konstruktora i destruktora

4.5.2.1 `VertStruct::VertStruct (const int initValue, const int initG, const int initH, const int initF) [inline]`

4.5.3 Dokumentacja atrybutów składowych

4.5.3.1 `int VertStruct::F`

4.5.3.2 `int VertStruct::G`

4.5.3.3 `int VertStruct::H`

4.5.3.4 `int VertStruct::Value`

Dokumentacja dla tej struktury została wygenerowana z pliku:

- [search.cpp](#)

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`, `astar` }
Typ danych przechowujący 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 przechowujący rodzaj wyszukiwania.

Wartości wyliczeń

bfs

dfs

astar

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>
#include <vector>
#include <algorithm>
```

Komponenty

- struct [VertStruct](#)
Struktura pomocnicza do użycia przez algorytm A.*
- struct [LessF](#)
Struktura pomocnicza do użycia przez algorytm A.*

Funkcje

- void [DFS](#) ([Graph](#)< int > *Ptr, int Start, int End)
Przeszukiwanie grafu w głąb.
- void [BFS](#) ([Graph](#)< int > *Ptr, int Start, int End)
Przeszukiwanie grafu wszerz.
- void [AStar](#) ([Graph](#)< int > *Ptr, int Start, int End, int Side)
Wyszukiwanie najkrótszej ścieżki - A.*

5.6.1 Opis szczegółowy

Plik zawierający funkcje przeszukujące graf wszerz oraz w głąb.

5.6.2 Dokumentacja funkcji

5.6.2.1 `void AStar (Graph< int > * Ptr, int Start, int End, int Side)`

Wyszukiwanie najkrótszej ścieżki - A*.

Parametry

<i>Ptr</i>	wskaźnik na przeszukiwany graf
------------	--------------------------------

<i>Side</i>	liczba wierzchołków przypadająca na bok planszy
<i>Start</i>	wierzchołek początkowy
<i>End</i>	wierzchołek końcowy

5.6.2.2 void BFS (Graph< int > * Ptr, int Start, int end)

Przeszukiwanie grafu wszerz.

Parametry

<i>Ptr</i>	wskaźnik na przeszukiwany graf
<i>Start</i>	wierzchołek początkowy
<i>End</i>	wierzchołek końcowy

5.6.2.3 void DFS (Graph< int > * Ptr, int Start, int End)

Przeszukiwanie grafu w głąb.

Parametry

<i>Ptr</i>	wskaźnik na przeszukiwany graf
<i>Start</i>	wierzchołek początkowy
<i>End</i>	wierzchołek końcowy

5.7 Dokumentacja pliku search.hh

Plik zawierający definicje funkcji przeszukujących grafy.

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

Funkcje

- void **DFS** (Graph< int > *Ptr, int Start, int End)
Przeszukiwanie grafu w głąb.
- void **BFS** (Graph< int > *Ptr, int Start, int end)
Przeszukiwanie grafu wszerz.
- void **AStar** (Graph< int > *Ptr, int Start, int End, int Side)
Wyszukiwanie najkrótszej ścieżki - A.*

5.7.1 Opis szczegółowy

Plik zawierający definicje funkcji przeszukujących grafy.

5.7.2 Dokumentacja funkcji

5.7.2.1 void AStar (Graph< int > * Ptr, int Start, int End, int Side)

Wyszukiwanie najkrótszej ścieżki - A*.

Parametry

<i>Ptr</i>	wskaźnik na przeszukiwany graf
<i>Side</i>	liczba wierzchołków przypadająca na bok planszy
<i>Start</i>	wierzchołek początkowy
<i>End</i>	wierzchołek końcowy

5.7.2.2 void BFS (Graph< int > * Ptr, int Start, int end)

Przeszukiwanie grafu wszerek.

Parametry

<i>Ptr</i>	wskaźnik na przeszukiwany graf
<i>Start</i>	wierzchołek początkowy
<i>End</i>	wierzchołek końcowy

5.7.2.3 void DFS (Graph< int > * Ptr, int Start, int End)

Przeszukiwanie grafu w głąb.

Parametry

<i>Ptr</i>	wskaźnik na przeszukiwany graf
<i>Start</i>	wierzchołek początkowy
<i>End</i>	wierzchołek końcowy

Skorowidz

- AStar
 - search.cpp, [9](#)
 - search.hh, [10](#)
- AddEdge
 - Graph, [5](#)
- AddVert
 - Graph, [5](#)
- astar
 - benchmark.hh, [8](#)
- BFS
 - search.cpp, [10](#)
 - search.hh, [11](#)
- benchGraph
 - Benchmark, [3](#)
- Benchmark, [2](#)
 - benchGraph, [3](#)
 - benchmark, [3](#)
 - calculate, [3](#)
 - Edges, [3](#)
 - GetEdges, [3](#)
 - SampleGraph, [3](#)
 - Side, [3](#)
 - Vertices, [3](#)
- benchmark
 - Benchmark, [3](#)
- benchmark.hh
 - astar, [8](#)
 - bfs, [8](#)
 - dfs, [8](#)
- benchmark.cpp, [7](#)
- benchmark.hh, [7](#)
 - Implementation, [8](#)
- bfs
 - benchmark.hh, [8](#)
- calculate
 - Benchmark, [3](#)
- DFS
 - search.cpp, [10](#)
 - search.hh, [11](#)
- dfs
 - benchmark.hh, [8](#)
- Edge
 - Graph::Edge, [4](#)
- EdgeS
 - Graph, [5](#)
- Edges
 - Benchmark, [3](#)
- F
 - VertStruct, [7](#)
- G
 - VertStruct, [7](#)
- GetEdges
 - Benchmark, [3](#)
- Graph
 - AddEdge, [5](#)
 - AddVert, [5](#)
 - EdgeS, [5](#)
 - graph, [6](#)
 - IfConnected, [5](#)
 - Neighbors, [5](#)
 - RemoveEdge, [5](#)
 - RemoveVert, [6](#)
- graph
 - Graph, [6](#)
- Graph< Type >, [4](#)
- Graph< Type >::Edge, [3](#)
- graph.hh, [8](#)
- Graph::Edge
 - Edge, [4](#)
 - SecEnd, [4](#)
 - Weight, [4](#)
- H
 - VertStruct, [7](#)
- IfConnected
 - Graph, [5](#)
- Implementation
 - benchmark.hh, [8](#)
- LessF, [6](#)
 - operator(), [6](#)
- main
 - main.cpp, [9](#)
- main.cpp, [8](#)
 - main, [9](#)
- mainpage.dox, [9](#)
- Neighbors
 - Graph, [5](#)
- operator()
 - LessF, [6](#)
- RemoveEdge
 - Graph, [5](#)
- RemoveVert
 - Graph, [6](#)
- SampleGraph
 - Benchmark, [3](#)
- search.cpp, [9](#)
 - AStar, [9](#)
 - BFS, [10](#)
 - DFS, [10](#)
- search.hh, [10](#)
 - AStar, [10](#)

BFS, 11
DFS, 11
SecEnd
 Graph::Edge, 4
Side
 Benchmark, 3

Value
 VertStruct, 7
VertStruct, 6
 F, 7
 G, 7
 H, 7
 Value, 7
 VertStruct, 7
 VertStruct, 7
Vertices
 Benchmark, 3

Weight
 Graph::Edge, 4