Projektowanie algorytmów i metod sztucznej inteligencji. Laboratorium 9 - Sprawozdanie

1.0

Wygenerowano przez Doxygen 1.8.9.1

Śr, 27 maj 2015 13:06:26

Spis treści

1 Projektowanie algorytmetod sztucznej inteligencji. Laboratorium 9 - Sprawozdanie

Autor

Wojciech Makuch

Data

27.05.2015

Wersja

1.0

program testujacy oraz benchmarkujacy zaimplementowane grafy posiada menu uzytkownika do wyboru

1.1 graf

1.2 graf

4 Indeks plików

4.1 Lista plików

Tutaj znajduje się lista wszystkich plików z ich krótkimi opisami:

5 Dokumentacja klas

5.1 Dokumentacja klasy CBenchmark

`#include <benchmark.hh>`

Dziedziczona przez CGraph.

Metody publiczne

- virtual void start_timer ()

  definicja metody start_timer rozpoczyna pomiar czasu zapisuje dane do zmiennej performanceCountStart korzysta z metdoy StartTimer()

- virtual void stop_timer ()

  definicja metody stop_timer konczy pomiar czasu zapsiuje dane do zmiennej performanceCountEnd korzysta z metody endTimer

- virtual int put_time_to_file (int size_of_list)

  definicja metody put_time_to_file otiwra plik o nazwie 'timing.txt' zapisuje do niego ilosc elementow listy oraz czas przeprowadzenia operacji przez klasy obserwowane zamyka plik.

Metody prywatne

- LARGE_INTEGER startTimer ()
- LARGE_INTEGER endTimer ()

Atrybuty prywatne

- LARGE_INTEGER performanceCountStart
- LARGE_INTEGER performanceCountEnd

5.1.1 Opis szczegółowy

definicja klasy CBenchmark definijue stoper zliczajacy czas wykoania operacji przez inne klasy jest przykladem wzorca obserwatora obserwuje klase CSort i zlicza czas sortowania listy

Definicja w linii 16 pliku benchmark.hh.

5.1.2 Dokumentacja funkcji składowych

5.1.2.1 LARGE_INTEGER CBenchmark::endTimer ( ) [private]

Definicja w linii 19 pliku benchmark.cpp.

5.1.2.2 int CBenchmark::put_time_to_file (int *size_of_list*) [virtual]

Parametry

size_of_list	- rozmiar listy

Zwraca

czas przeprowadzenia operacji
-1 w przypadu bledu otwarcia pliku

Definicja w linii 38 pliku benchmark.cpp.

5.1.2.3 void CBenchmark::start_timer ( ) [virtual]

Definicja w linii 28 pliku benchmark.cpp.

5.1.2.4 LARGE_INTEGER CBenchmark::startTimer ( ) [private]

Definicja w linii 10 pliku benchmark.cpp.

5.1.2.5 void CBenchmark::stop_timer ( ) [virtual]

Definicja w linii 33 pliku benchmark.cpp.

5.1.3 Dokumentacja atrybutów składowych

5.1.3.1 LARGE_INTEGER CBenchmark::performanceCountEnd [private]

Definicja w linii 18 pliku benchmark.hh.

5.1.3.2 LARGE_INTEGER CBenchmark::performanceCountStart [private]

Definicja w linii 17 pliku benchmark.hh.

Dokumentacja dla tej klasy została wygenerowana z plików:

- benchmark.hh
- benchmark.cpp

5.2 Dokumentacja klasy CEdge

definicja klasy CEdge definuje krawedz grafu nie zawiera wag

`#include <graph.hh>`

Atrybuty prywatne

- CNode * prev
- CNode * next

Przyjaciele

- class CGraph

5.2.1 Opis szczegółowy

Definicja w linii 28 pliku graph.hh.

5.2.2 Dokumentacja przyjaciół i funkcji związanych

5.2.2.1 friend class CGraph [friend]

Definicja w linii 29 pliku graph.hh.

5.2.3 Dokumentacja atrybutów składowych

5.2.3.1 CNode* CEdge::next [private]

wskaznik na poprzedni wezel

Definicja w linii 31 pliku graph.hh.

5.2.3.2 CNode* CEdge::prev [private]

Definicja w linii 30 pliku graph.hh.

Dokumentacja dla tej klasy została wygenerowana z pliku:

- graph.hh

5.3 Dokumentacja klasy CGraph

definicja klasy CGraph definije graf skierowany bez wagowaych krawedzi dziedzicy po klasie CBenchmark

`#include <graph.hh>`

Dziedziczy CBenchmark.

Metody publiczne

- CGraph (int v, int e)

  definicja konstruktora parametrycznego alokuje pamiec dla tablic wezlow krawdzi oraz dla macierzy sasiedztwa
- ~CGraph ()

  definicja destruktora

- void set_graph (int edge1, int vertice1, int vertice2)

  definicja metody set_graph wiaze wezly wskaznikami krawdziom ustawia odpowiednie wezly wypelnia macierz sasiedztwa
- void print_matrix () const

  definicja metody print_maatrix wyswietla macierz sasiedztwa
- CEdge ∗ search (int key)

  definicja metody search metoda powinna zwracac liste krawedzi prowadzoca do danego elementu WYMAGA DO⟷ PRACOWANIA
- void DFSing (int v)

  definicja metody DFSing przeszukuje graf algorytmem DFS (przeszukiwanie w glab) zaimplementowana rekuencyjnie wykorzystana w metodzie DFS wyposarzonej w timery
- void DFS (int v)

  definicja metody DFS wywoluje metode DFSing posiada timery zapisujace zmierzony czas do pliku
- void BFS (int v)

  definicja meotdy BFS przeszukuje graf algorytmem BFS (przeszukiwanie w wszerz) posiada timery zapisujace zmierzony czas do pliku nie jest zaimplementowana rekurencyjnie, nie wymaga zewnetrznej metody
- CGraph ∗ make_random_graph (int sizeV, int sizeE)

  definicja metody make_random_graph tworzy graf o losowych wezlach oraz losowych krawedziach przypisuje this na wygenerowanych graf
- void benchmarking_DFS ()

  definicja metody benchmarking_DFS metoda posiada petle wywolujace metode DFS w zakresie 1-10 000 dzieki timerom do pliku zostaja zapisane dane benchmarku
- void benchmarking_BFS ()

  definicja metody benchmarking_BFS metoda posiada petle wywolujace metode BFS w zakresie 1-10 000 dzieki timerom do pliku zostaja zapisane dane benchmarku

Atrybuty prywatne

- int V
- int E
- CNode ∗ ListV
- CEdge ∗ ListE
- int ∗∗ matrix
- bool ∗ visited

5.3.1 Opis szczegółowy

Definicja w linii 41 pliku graph.hh.

5.3.2 Dokumentacja konstruktora i destruktora

5.3.2.1 CGraph::CGraph (int *v*, int *e*)

lista odwiedzonych elementow grafu

Parametry

v	ilosc wezlow
e	ilosc krawedzi

Definicja w linii 12 pliku graph.cpp.

5.3.2.2 CGraph::∼CGraph ( )

Definicja w linii 31 pliku graph.cpp.

5.3.3 Dokumentacja funkcji składowych

5.3.3.1 void CGraph::benchmarking_BFS ( )

Definicja w linii 182 pliku graph.cpp.

5.3.3.2 void CGraph::benchmarking_DFS ( )

Definicja w linii 157 pliku graph.cpp.

5.3.3.3 void CGraph::BFS (int *v*)

Parametry

v	element poczatkowy od ktorego algorytm DFS rozpoczyna przeszukiwanie

Definicja w linii 86 pliku graph.cpp.

5.3.3.4 void CGraph::DFS (int *v*)

Parametry

v	element poczatkowy od ktorego algorytm DFS rozpoczyna przeszukiwanie

Definicja w linii 78 pliku graph.cpp.

5.3.3.5 void CGraph::DFSing (int *v*)

Parametry

v	element poczatkowy od ktorego algorytm rozpoczyna przeszukiwanie

Definicja w linii 66 pliku graph.cpp.

5.3.3.6 CGraph * CGraph::make_random_graph (int *sizeV,* int *sizeE*)

Parametry

sizeV	ilosc wierzholkow
sizeE	ilosc krawedzi

Zwraca

losowy graf

Definicja w linii 134 pliku graph.cpp.

5.3.3.7 void CGraph::print_matrix ( ) const

Definicja w linii 56 pliku graph.cpp.

5.3.3.8 CEdge * CGraph::search (int *key*)

Parametry

key	szukany klucz/element

Zwraca

lista krawiedzi

Definicja w linii 115 pliku graph.cpp.

5.3.3.9 void CGraph::set_graph (int *edge1,* int *vertice1,* int *vertice2*)

Parametry

edge1	krawdz do ktorej przypisujemy poszczegolne wierzcholki
vertice1	przypisywany pierwszy wierzcholek
vertice2	przypisywany drugi wierzcholek

Definicja w linii 46 pliku graph.cpp.

5.3.4 Dokumentacja atrybutów składowych

5.3.4.1 int CGraph::E [private]

ilosc wezlow

Definicja w linii 43 pliku graph.hh.

5.3.4.2 CEdge* CGraph::ListE [private]

tablica wezlow

Definicja w linii 45 pliku graph.hh.

5.3.4.3 CNode* CGraph::ListV [private]

ilosc krawedzi

Definicja w linii 44 pliku graph.hh.

5.3.4.4 int** CGraph::matrix [private]

tablica krawedzi

Definicja w linii 46 pliku graph.hh.

5.3.4.5 int CGraph::V [private]

Definicja w linii 42 pliku graph.hh.

5.3.4.6 bool* CGraph::visited [private]

macierz sasiedztwa

Definicja w linii 47 pliku graph.hh.

Dokumentacja dla tej klasy została wygenerowana z plików:

- graph.hh
- graph.cpp

5.4 Dokumentacja klasy CNode

definicja klasy CNode definiuje pojedynczy wezel grafu

`#include <graph.hh>`

Atrybuty prywatne

- int value

Przyjaciele

- class CEdge

- class CGraph

5.4.1 Opis szczegółowy

Definicja w linii 17 pliku graph.hh.

5.4.2 Dokumentacja przyjaciół i funkcji związanych

5.4.2.1 friend class CEdge [friend]

Definicja w linii 18 pliku graph.hh.

5.4.2.2 friend class CGraph [friend]

Definicja w linii 19 pliku graph.hh.

5.4.3 Dokumentacja atrybutów składowych

5.4.3.1 int CNode::value [private]

Definicja w linii 20 pliku graph.hh.

Dokumentacja dla tej klasy została wygenerowana z pliku:

- graph.hh

5.5 Dokumentacja klasy queue

definicja klasy queue definicja kolejki ADT, kolejka typu FIFO zimplementowaa na liscie

`#include <queue.hh>`

Metody publiczne

- void push (int element)

  definicja metody push dodaje element na koniec kolejki
- void pop ()

  definicja metody pop usuwa element z poczatku kolejki
- ∼queue ()

  definicja destruktora
- queue ()

  desfinicja konstruktora bezparametrycznego ustawia wskazniki na NULL
- void print () const

  definicja metody print wyswietla zawartosc kolejki

Atrybuty publiczne

- queue_node ∗ first
- queue_node ∗ last

5.5.1 Opis szczegółowy

Definicja w linii 29 pliku queue.hh.

5.5.2 Dokumentacja konstruktora i destruktora

5.5.2.1 queue::∼queue ( )

Definicja w linii 49 pliku queue.cpp.

5.5.2.2 queue::queue ( ) [inline]

Definicja w linii 57 pliku queue.hh.

5.5.3 Dokumentacja funkcji składowych

5.5.3.1 void queue::pop ( )

Definicja w linii 23 pliku queue.cpp.

5.5.3.2 void queue::print ( ) const

Definicja w linii 35 pliku queue.cpp.

5.5.3.3 void queue::push (int *element*)

wskaznik na ostatni element

Parametry

element	dodawany element

Definicja w linii 9 pliku queue.cpp.

5.5.4 Dokumentacja atrybutów składowych

5.5.4.1 queue_node∗ queue::first

Definicja w linii 31 pliku queue.hh.

5.5.4.2 queue_node∗ queue::last

wskaznik na pierwszy element

Definicja w linii 32 pliku queue.hh.

Dokumentacja dla tej klasy została wygenerowana z plików:

- queue.hh
- queue.cpp

5.6 Dokumentacja klasy queue_node

definicja klasy queue_node definicja wezla dla kolejki definiuje pojedynczy element bedacy w kolejkce

`#include <queue.hh>`

Atrybuty publiczne

- queue_node ∗ next
- int data

5.6.1 Opis szczegółowy

Definicja w linii 17 pliku queue.hh.

5.6.2 Dokumentacja atrybutów składowych

5.6.2.1 int queue_node::data

wskaznik na nastepny

Definicja w linii 20 pliku queue.hh.

5.6.2.2 queue_node* queue_node::next

Definicja w linii 19 pliku queue.hh.

Dokumentacja dla tej klasy została wygenerowana z pliku:

- queue.hh

6 Dokumentacja plików

6.1 Dokumentacja pliku benchmark.cpp

implmentuje zdefiniowana klase benchmarka

```
#include "benchmark.hh"
#include <windows.h>
#include <fstream>
#include <iostream>
```

6.2 Dokumentacja pliku benchmark.hh

definiuje klase CBenchmark

```
#include <windows.h>
```

Komponenty

- class CBenchmark

6.3 Dokumentacja pliku graph.cpp

implementuje zdefiniowana klase grafu

```
#include "graph.hh"
#include "queue.hh"
#include "benchmark.hh"
#include <iostream>
```

6.4 Dokumentacja pliku graph.hh

zwiera definicje klas CNode, CEdge, CGrpah CNode - wezel grafu CEdge - krawedz grafu CGraph - graf

```
#include "benchmark.hh"
```

Komponenty

- class CNode

  definicja klasy CNode definiuje pojedynczy wezel grafu
- class CEdge

  definicja klasy CEdge definuje krawedz grafu nie zawiera wag
- class CGraph

  definicja klasy CGraph definije graf skierowany bez wagowaych krawedzi dziedzicy po klasie CBenchmark

6.5 Dokumentacja pliku main.cpp

glowna funkcja programu

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

Funkcje

- int main ()

6.5.1 Dokumentacja funkcji

6.5.1.1 int main ( )

Definicja w linii 37 pliku main.cpp.

6.6 Dokumentacja pliku queue.cpp

implementuje zdefiniowana klase kolejki

```
#include "queue.hh"
#include <iostream>
```

6.7 Dokumentacja pliku queue.hh

zawiera definicje klas queue_node, queue queue_node - wezel kolejki queue - kolejka

```
#include <iostream>
```

Komponenty

- class queue_node

  definicja klasy queue_node definicja wezla dla kolejki definiuje pojedynczy element bedacy w kolejkce
- class queue

  definicja klasy queue definicja kolejki ADT, kolejka typu FIFO zimplementowaa na liscie

Projektowanie algorytmów i metod sztucznej inteligencji. Laboratorium 9 - Sprawozdanie

Wojciech Makuch

7 Zadanie

Implementacja grafu oraz zbadanie złożoności obliczeniowej algorytmów przeszukiwania w głąb(ang. *Depth-first search, DFS*) oraz przeszukiwania wszerz(ang. *breadth-first search, BFS*).

8 Wstep

Graf – abstrakcyjna struktura danych. Jest zbudowana z wierzchołków i krawędzi. Wierzchołki grafu mogą być numerowane i czasem stanowią reprezentację jakichś obiektów, natomiast krawędzie mogą wówczas obrazować relacje między takimi obiektami. Wierzchołki należące do krawędzi nazywane są jej końcami. Krawędzie mogą mieć wyznaczony kierunek, a graf zawierający takie krawędzie nazywany jest grafem skierowanym. Krawędź grafu może posiadać wagę, to znaczy przypisaną liczbę, która określa, na przykład, odległość między wierzchołkami.

Przeszukiwanie w głąb - polega na badaniu wszystkich krawędzi wychodzących z podanego wierzchołka. Teoretyczna złożoność czasowa przeszukiwania w głąb wynosi $O(|V| + |E|)$, gdzie V-ilość wierzchołków, E – ilość krawędzi. Teoretyczna złożoność pamięciowa wynosi $O(h)$, gdzie h – długość najdłuższej prostej ścieżki.

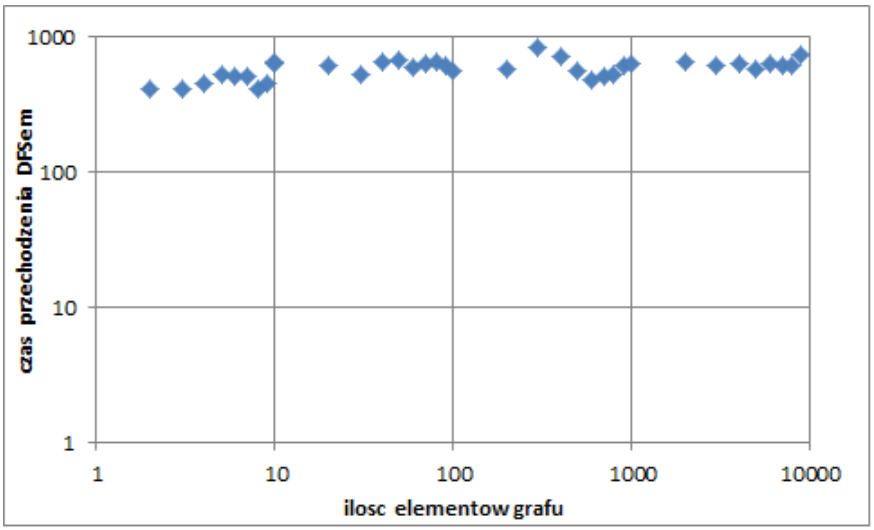
Przeszukiwanie wszerz - przechodzenie grafu rozpoczyna się od zadanego wierzchołka i polega na odwiedzeniu wszystkich osiągalnych z niego wierzchołków. Teoretyczna złożoność czasowa wynosi również $O(|V| + |E|)$. Złożoność pamięciowa w tym przypadku też wynosi $O(|V| + |E|)$.

9 Graf - implementacja

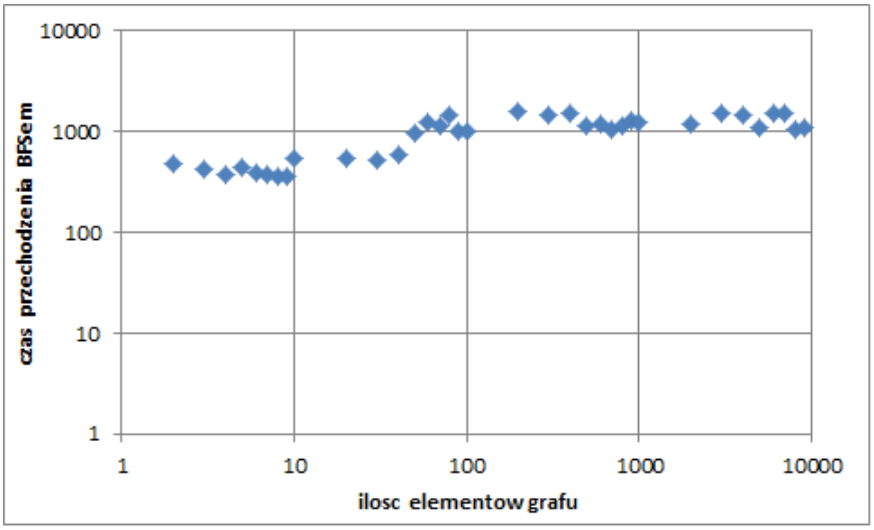
Utworzono klasy przechowujące elementy grafu takie krawędzie i wierzchołki oraz klasę nadrzędną, która łączy je w jeden graf. Informacja o połączeniu wierzchołków i krawędzi jest zapisywana w macierzy sąsiedztwa. Do zaalokowania pamięci niezbędne jest wcześniejsze podanie rozmiaru. Głowna funkcja programu zawiera menu użytkownika pozwalające testować poprawność programu oraz testować złożoność obliczeniową badanych algorytmów.

10 Zlozonosci obliczeniowe usyskane

Na rys 1. pokazano uzyskane wyniki badania złożoności obliczeniowej dla algorytmu DFS, natomiast to samo dla algorytmu BFS przedstawiono na rys 2. Z obydwu rysunków wynika, że złożoność czasowa jest w przybliżeniu stała i oscyluje w granicy wartości 1000. Z praktycznego punktu widzenia jest to złożoność zgodna z teoretyczną ponieważ zaimplementowana w programie metoda tworzenia losowych grafów tworzy również losowe krawędzie, przez co w większości wypadków nie wszystkie wierzchołki są połączone z grafem - tworzy się graf niepołączony/niespójny. W efekcie ilość krawędzi jest dużo mniejsza.



Rysunek 1: Dodawanie do drzewa binarnego.



Rysunek 2: Przszeszukiwanie drzewa binarnego.

11 Komentarz

Do utworzenia dokumentacji wykorzystano system Doxygen. Funkcja pomiaru czasu dla systemu Windows pobrana ze strony dr. J. Mierzwy. Program skompilowano w środowisku Code::Blocks. Do stworzenia wykresu posłużono się pakietem MS Excel, sprawozdanie napisano używając systemu LaTeX.

Literatura

[1] http://pl.wikipedia.org/wiki/Przeszukiwanie_w_glab

[2] http://pl.wikipedia.org/wiki/Przeszukiwanie_wszerz