Laboratorium 7 - Projektowanie algorytmów i metod sztucznej inteligencji.

2.0

Generated by Doxygen 1.8.9.1

Wed May 13 2015 14:14:37

Contents

1 modyfkacja sortowania - benchmark - obserwator

Author

Wojcich Makuch

Date

12.05.2015

Version

2.0 program wzbogacono o menu, pozwalajace spradzic poprawnosc algorytmow. wszystkie klasy, metody, zmienne itp. zaimplementowano w jezyku angielskim.

2 Hierarchical Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

3 Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

4 File Index

4.1 File List

Here is a list of all files with brief descriptions:

5 Class Documentation

5.1 CBenchmark Class Reference

`#include <benchmark.hh>`

Inherited by CSort.

Public Member Functions

- 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.

Private Member Functions

- LARGE_INTEGER startTimer ()
- LARGE_INTEGER endTimer ()

Private Attributes

- LARGE_INTEGER performanceCountStart
- LARGE_INTEGER performanceCountEnd

5.1.1 Detailed Description

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

5.1.2 Member Function Documentation

5.1.2.1 LARGE_INTEGER CBenchmark::endTimer ( ) [private]

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

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.

Parameters

size_of_list	- rozmiar listy

Returns

czas przeprowadzenia operacji
-1 w przypadu bledu otwarcia pliku

5.1.2.3 void CBenchmark::start_timer ( ) [virtual]

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

5.1.2.4 LARGE_INTEGER CBenchmark::startTimer ( ) [private]

5.1.2.5 void CBenchmark::stop_timer ( ) [virtual]

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

5.1.3 Member Data Documentation

5.1.3.1 LARGE_INTEGER CBenchmark::performanceCountEnd [private]

5.1.3.2 LARGE_INTEGER CBenchmark::performanceCountStart [private]

The documentation for this class was generated from the following files:

- benchmark.hh
- benchmark.cpp

5.2 CHeapSort Class Reference

definicja klasy CHeapSort definiuje sortowanie przez kopcowanie przyklad klasy obserwowanej

`#include <heap_sort.hh>`

Inherits CSort.

Public Member Functions

- void sorting (CList *list, int useless, int last)

  definicja metody sorting sortuje wykorzystujac algorytm heapsort tworzy kopiec zamienia najwiekszy element z najwiekszym(wysyla najwiekszy na koniec listy przywraca wlasciwosc kopca korzystajac z metody up_heap
- void sort (CList *list, int useless, int last)

  definicja metody sort implemetacja metody abstakcyjnej wykorzystuje heapsort z wykorzystaniem timerow
- void build_heap (CList *list, int last)

  definicja metody build_heap tworzy kopiec z listy wykorzystuje metode down_heap
- void down_heap (CList *list, int parent, int last)

  definicja metody max_heapify przywraca wlasciwosc kopca zakladajac korzen - najwiekszy element
- void up_heap (CList *list, int last)

  definicja metody up_heap przywraca wlasciwosc kopca zakladajac korzen - najwiekszy element
- void benchmarking (CList *list)

  feinicja metody benchmarking sortuje listy w zakresie 1- 10 000 wykonujac metode sort, zapisuje dane z licznikw czasu do pliku

5.2.1 Detailed Description

definicja klasy CHeapSort definiuje sortowanie przez kopcowanie przyklad klasy obserwowanej

5.2.2 Member Function Documentation

5.2.2.1 void CHeapSort::benchmarking (CList * *list*) [virtual]

feinicja metody benchmarking sortuje listy w zakresie 1- 10 000 wykonujac metode sort, zapisuje dane z licznikw czasu do pliku

Parameters

list	- benchmarkowana lista

Implements CSort.

5.2.2.2 void CHeapSort::build_heap (CList * *list*, int *last*)

definicja metody build_heap tworzy kopiec z listy wykorzystuje metode down_heap

Parameters

list	- modyfkowana lista
last	- maksymalny rozmiar kopca

5.2.2.3 void CHeapSort::down_heap (CList * *list*, int *parent*, int *last*)

definicja metody max_heapify przywraca wlasciwosc kopca zakladajac korzen - najwiekszy element

Parameters

list	- modyfikowana lista
parent	- 'korzen poddrzewa'
last	- maksymalny rozmiar drzewa algorytm przyraca wlasciwosc kopca zaczynajac od korzenia

5.2.2.4 void CHeapSort::sort (CList * *list*, int *useless*, int *last*) [virtual]

definicja metody sort implemetacja metody abstakcyjnej wykorzystuje heapsort z wykorzystaniem timerow

Parameters

list	- sortowana lista
useless	- useless
last	- maksymany romziar listy

Implements CSort.

5.2.2.5 void CHeapSort::sorting (CList * *list*, int *useless*, int *last*)

definicja metody sorting sortuje wykorzystujac algorytm heapsort tworzy kopiec zamienia najwiekszy element z najwiekszym(wysyla najwiekszy na koniec listy przywraca wlasciwosc kopca korzystajac z metody up_heap

Parameters

list	- sortowana lista
useless	- useless
last	- maksymany romziar listy

5.2.2.6 void CHeapSort::up_heap (CList * *list*, int *last*)

definicja metody up_heap przywraca wlasciwosc kopca zakladajac korzen - najwiekszy element

Parameters

list	- modyfikowana lista
last	- maksymalny rozmiar kopca

The documentation for this class was generated from the following files:

- heap_sort.hh
- heap_sort.cpp

5.3 CList Class Reference

klasa lista - ADT modeluje prost liste jednokierunkowa zwiera metody niezbedne do implementacji sortowania

`#include <list.hh>`

Public Member Functions

- CList ()
- ~CList ()
- void print () const

  definicja funkcji wyswietlajacej wyswietla na strumieniu wyjsciowym ciag elementow zapisanych na liscie
- void push (int element)

  definicja metody push dodaje nowy element na liste
- int pop ()

  definicja metody pop usuwa element z lsity
- int get_value (int i) const

  definicja metody get_value odowluje sie do wybranego elemntu listy
- int & get_value (int i)

  definicja przeciazenia get_value
- void swap (int i, int j)

  definicja metody swap zamienia elementy listy
- bool is_empty ()

  definicja metody is_empty
- int get_size ()

  definicja metody get_size
- void pull (int i)

  definicja metody pull wypelnia liste liczbami pesudolosowymi

Private Attributes

- int value
- CList * next

5.3.1 Detailed Description

klasa lista - ADT modeluje prost liste jednokierunkowa zwiera metody niezbedne do implementacji sortowania

5.3.2 Constructor & Destructor Documentation

5.3.2.1 CList::CList ( )

5.3.2.2 CList::~CList ( )

5.3.3 Member Function Documentation

5.3.3.1 int CList::get_size ( )

definicja metody get_size

Returns

ilosc elementow na liscie

5.3.3.2 int CList::get_value (int *i*) const

definicja metody get_value odowluje sie do wybranego elemntu listy

Parameters

i	- indeks kom listy

Returns

element o indeksie i

5.3.3.3 int & CList::get_value (int *i*)

definicja przeciazenia get_value

Returns

referencje do elementu na lsicie

5.3.3.4 bool CList::is_empty ( )

definicja metody is_empty

Returns

true - gdy lista jest pusta
false - w przypadku przeciwnym

5.3.3.5 int CList::pop ( )

definicja metody pop usuwa element z lsity

Returns

usuwany element

5.3.3.6 void CList::print ( ) const

definicja funkcji wyswietlajacej wyswietla na strumieniu wyjsciowym ciag elementow zapisanych na liscie

5.3.3.7 void CList::pull (int *i*)

definicja metody pull wypelnia liste liczbami pesudolosowymi

Parameters

i	- ilosc elementow

5.3.3.8 void CList::push (int *element*)

definicja metody push dodaje nowy element na liste

Parameters

element	- dodawana komorka do listy

5.3.3.9 void CList::swap (int *i*, int *j*)

definicja metody swap zamienia elementy listy

Parameters

i	- inkeks do pierwszego elementu
j	- indeks do drugiego elementu

5.3.4 Member Data Documentation

5.3.4.1 CList* CList::next [private]

5.3.4.2 int CList::value [private]

The documentation for this class was generated from the following files:

- list.hh
- list.cpp

5.4 CMergeSort Class Reference

definicja klasy CMergeSort definiuje sortowanie przez scalanie jest przykadem klasy obserwowanej implementuje metode abstakcyjna sort

`#include <merge_sort.hh>`

Inherits CSort.

Public Member Functions

- void sorting (CList *list, int left, int right)

  definicja metody sorting sortuje liste poprzez algorytm sortowania przez scalanie
- void sort (CList *list, int left, int right)

  definicja metody sort implementacja metody czysto abstrakcyjnej korzysta z metody sorting korzysta z timerow
- void benchmarking (CList *list)

  feinicja metody benchmarking sortuje listy w zakresie 1- 10 000 wykonujac metode sort, zapisuje dane z licznikw czasu do pliku

5.4.1 Detailed Description

definicja klasy CMergeSort definiuje sortowanie przez scalanie jest przykadem klasy obserwowanej implementuje metode abstakcyjna sort

5.4.2 Member Function Documentation

5.4.2.1 void CMergeSort::benchmarking (CList * *list*) [virtual]

feinicja metody benchmarking sortuje listy w zakresie 1- 10 000 wykonujac metode sort, zapisuje dane z licznikw czasu do pliku

Parameters

list	- benchmarkowana lista

Implements CSort.

5.4.2.2 void CMergeSort::sort (CList * *list*, int *left*, int *right*) [virtual]

definicja metody sort implementacja metody czysto abstrakcyjnej korzysta z metody sorting korzysta z timerow

Parameters

list	- sortowana lista
left	- indeks na 1-szy element
right	- indeks na ostatni element sortowanej listy

Implements CSort.

5.4.2.3 void CMergeSort::sorting (CList * *list*, int *left*, int *right*)

definicja metody sorting sortuje liste poprzez algorytm sortowania przez scalanie

Parameters

list	- sortowana lista
left	- indeks na 1-szy element
right	- indeks na ostatni element sortowanej listy

The documentation for this class was generated from the following files:

- merge_sort.hh
- merge_sort.cpp

5.5 CQuickSort Class Reference

defnijca klasy CSort definiuje sortowanie szybkie jest przykaladem klasy obserwowanej

`#include <quick_sort.hh>`

Inherits CSort.

Public Member Functions

- void sorting (CList *list, int left, int right)

  defnicja metody sorting implementacja algortymu quicksort bez timerow
- void sort (CList *list, int left, int right)

  definicja metody sort implementacja algorytmu quicksort z wykorzystaniem timerow
- void benchmarking (CList *list)

  feinicja metody benchmarking sortuje listy w zakresie 1- 10 000 wykonujac metode sort, zapisuje dane z licznikw czasu do pliku

5.5.1 Detailed Description

defnijca klasy CSort definiuje sortowanie szybkie jest przykaladem klasy obserwowanej

5.5.2 Member Function Documentation

5.5.2.1 void CQuickSort::benchmarking (CList * *list*) [virtual]

feinicja metody benchmarking sortuje listy w zakresie 1- 10 000 wykonujac metode sort, zapisuje dane z licznikw czasu do pliku

Parameters

list	- benchmarkowana lista

Implements CSort.

5.5.2.2 void CQuickSort::sort (CList * *list*, int *left*, int *right*) [virtual]

definicja metody sort implementacja algorytmu quicksort z wykorzystaniem timerow

Parameters

list	- sortowana lista
left	- indeks na 1 element sortowanej listy
right	- indeks na prawy element sortowanej listy

Implements CSort.

5.5.2.3 void CQuickSort::sorting (CList * *list*, int *left*, int *right*)

defnicja metody sorting implementacja algortymu quicksort bez timerow

Parameters

list	- sortowana lista
left	- indeks na 1 element sortowanej listy
right	- indeks na prawy element sortowanej listy

The documentation for this class was generated from the following files:

- quick_sort.hh
- quick_sort.cpp

5.6 CSort Class Reference

definicja klasy abstrakcyjnej CSort jest klasa bazowa dla konkretnych typow sortowan. jest przykladem klasy obserwowanej.

`#include <csort.hh>`

Inherits CBenchmark.

Inherited by CHeapSort, CMergeSort, and CQuickSort.

Public Member Functions

- virtual void sort (CList *, int, int)=0

  definicja metody sort sortuje elementy na liscie.
- virtual void benchmarking (CList *list)=0

5.6.1 Detailed Description

definicja klasy abstrakcyjnej CSort jest klasa bazowa dla konkretnych typow sortowan. jest przykladem klasy obserwowanej.

5.6.2 Member Function Documentation

5.6.2.1 virtual void CSort::benchmarking (CList * *list*) [pure virtual]

Implemented in CHeapSort, CMergeSort, and CQuickSort.

5.6.2.2 virtual void CSort::sort (CList * , int , int) [pure virtual]

definicja metody sort sortuje elementy na liscie.

Parameters

list	- sortowana lista
left	- indeks pierwszego elementu listy sortowanej.
right	- indeks ostatniego elementu listy sortowanej.

Implemented in CHeapSort, CMergeSort, and CQuickSort.

The documentation for this class was generated from the following file:

- csort.hh

6 File Documentation

6.1 benchmark.cpp File Reference

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

6.2 benchmark.hh File Reference

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

Classes

- class CBenchmark

6.3 csort.hh File Reference

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

Classes

- class CSort

  definicja klasy abstrakcyjnej CSort jest klasa bazowa dla konkretnych typow sortowan. jest przykladem klasy obserwowanej.

6.4 heap_sort.cpp File Reference

```
#include "heap_sort.hh"
```

6.5 heap_sort.hh File Reference

```
#include "list.hh"
#include "csort.hh"
```

Classes

- class CHeapSort

  definicja klasy CHeapSort definiuje sortowanie przez kopcowanie przyklad klasy obserwowanej

6.6 list.cpp File Reference

```
#include "list.hh"
#include <iostream>
#include <cstdlib>
```

6.7 list.hh File Reference

Classes

- class CList

  klasa lista - ADT modeluje prost liste jednokierunkowa zwiera metody niezbedne do implementacji sortowania

6.8 main.cpp File Reference

```
#include <iostream>
#include <ctime>
#include "list.hh"
#include "benchmark.hh"
#include "csort.hh"
#include "heap_sort.hh"
#include "quick_sort.hh"
#include "merge_sort.hh"
```

Functions

- int main ()

6.8.1 Function Documentation

6.8.1.1 int main ( )

6.9 merge_sort.cpp File Reference

```
#include <iostream>
#include "list.hh"
#include "merge_sort.hh"
```

6.10 merge_sort.hh File Reference

```
#include "list.hh"
#include "csort.hh"
```

Classes

- class CMergeSort

  definicja klasy CMergeSort definiuje sortowanie przez scalanie jest przykadem klasy obserwowanej implementuje metode abstakcyjna sort

6.11 quick_sort.cpp File Reference

```
#include "quick_sort.hh"
```

6.12 quick_sort.hh File Reference

```
#include "list.hh"
#include "benchmark.hh"
#include "csort.hh"
```

Classes

- class CQuickSort

  defnijca klasy CSort definiuje sortowanie szybkie jest przykaladem klasy obserwowanej

Projektowanie algorytmów i metod sztucznej inteligencji
Laboratorium 7 - Sprawozdanie

Wojciech Makuch

7 Zadanie

Jako zadanie na zajęcia należało zmodyfikować implementację licznika czasu tak, aby korzystały one ze wzorca projektowania: obserwator. Ponadto należało zmodyfikować algorytmy sortowania, aby zachowane zostały zasady SOLID, a w szczególności zasada *Open/Closed principle* - program otwarty na rozszerzanie, ale zamknięty na modyfikację.

8 Realizacja

Większość klas, metod i funkcji zmodyfikowano zmieniając ich nazwy, nazwy zmiennych itp. aby były napisane w Języku angielskim. Strukturę danych lista napisano od początku, ponieważ stara zawierała błąd w implementacji przy próbie odniesienia się do pierwszego elementu, poza tym brakowało jej niezbędnych metod, takich jak np. *get_value()*. Jeśli chodzi o sortowania, zgodnie z zaleceniem prowadzącego utworzono klasę abstrakcyjną po której dziedziczyły klasy *CQuickSort*, *CHeapSort* oraz *CMergeSort*, każda z nich implementowała metodę sort. Zmodyfikowano implementację licznika czasu; teraz korzysta ze wzorca projektowego obserwator za pomocą mechanizmu dziedziczenia. Obserwuje klasę abstrakcyjną *CSort*, w której wykorzystano metody benchmarkujące *star_timer()* i *stop_timer()*. Po wykonaniu sortowania metoda klasy Benchmark zapisuję do pliku ilość elementów sortowanych oraz czas sortowania. Ponadto dodano do funkcji głównej programu menu dla użytkownika, które pozwala na sprawdzenie poprawności zaimplementowanych metod.

9 Dzialanie

Wszystkie algorytmy działają prawidłowo. Dane do pliku dopisywane są po każdym sortowaniu. Plik trzeba usuwać ręcznie. Dzięki metodzie benchmarking nie trzeba recznie testowac sortowania kazdej listy.

10 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.