Median Task

Generated by Doxygen 1.8.16

Sun Oct 27 2019 13:01:55

Chapter 1

Median

Demonstrates several implementations of median of sorted elements calculation:

MedianVector - uses std::sort
MedianNthElement - uses std::nth_element
MedianVectorLBound - uses std::lower_bound
MedianMultisetAdvance - uses std::multiset and std::advance
MedianMultisetIterator - uses std::multiset with median iterator
MedianRBTree - uses Red-Black tree
MedianMinMaxHeap - uses min and max binary heaps (fastest method)

Compiled with VS2017 or later (with loading CMakeLists.txt from File->Open->CMake..).

This will build a console application that runs Unit Tests for several containers in Debug and several Performance tests in Release.

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

Chapter 3

Class Index

3.1 Class List

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

Chapter 4

File Index

4.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 5

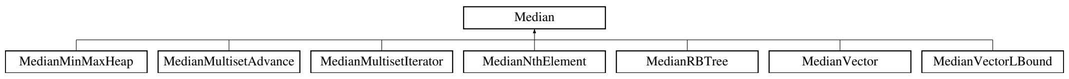
Class Documentation

5.1 Median Class Reference

Median class.

```
#include <Median.h>
```

Inheritance diagram for Median:



Public Member Functions

- virtual bool add (int a)=0

    add new integer element
- virtual int middle () const =0

    return value of the middle element
- virtual size_t size () const =0

    return number of all elements
- virtual void clear ()=0

    empty container
- virtual const char $*$ name () const =0

    return string with container name

5.1.1 Detailed Description

Median class.

Median calculation class interface

All Median Calculations classes needs to support Median Units Tests platform and should inherit **Median** interface and implement its virtual methods.
Purpose of Median and the derived classes is to find the **middle** element. By **middle** it is understood the sorted in ascending order element exactly at position N/2, where N is the number of all added elements, if N is odd. If N is even, the **middle** element is at positoon N/2-1. This is shown below:

Odd - 1,2,3 N=3, **middle** is 2, at position 1
Even - 1,2,3,4 N=4, **middle** is 2 also at position 1
Odd - 1,2,3,4,5 N=5, **middle** is 3 at position 2

Median classes implementations were tested with units test (see UnitTests.h).
Performance tests comparing the algorithms are run in Release (see PerformanceTest.h).

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

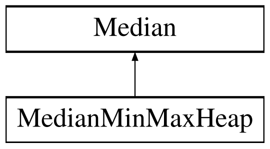
- Median.h

5.2 MedianMinMaxHeap Class Reference

MedianMinMaxHeap finds median using two **std::priority_queue**

```
#include <Median.h>
```

Inheritance diagram for MedianMinMaxHeap:



Public Member Functions

- virtual bool add (int a)

  add new integer element
- virtual int middle () const

  return value of the middle element
- virtual size_t size () const

  return number of all elements
- virtual void clear ()

  empty container
- virtual const char ∗ name () const

  return string with container name

Static Public Member Functions

- static const char ∗ **Name** ()

Protected Attributes

- std::priority_queue< int, std::vector< int >, std::less< int > > **m_s**
- std::priority_queue< int, std::vector< int >, std::greater< int > > **m_g**
- int **m_middle**

5.2.1 Detailed Description

MedianMinMaxHeap finds median using two **std::priority_queue**

MedianMinMaxHeap uses for storage container two binary heaps, one min and one max.
Allows inserting duplicating values.
Obtain the middle value as checks which of both heaps has more elements.
See: `https://www.geeksforgeeks.org/median-of-stream-of-running-integers-using-stl/`

add() Average Complexity O(log n)
middle() Average Complexity O(1)

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

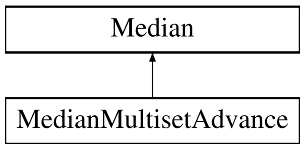
- Median.h
- Median.cpp

5.3 MedianMultisetAdvance Class Reference

MedianMultisetAdvance finds median using **std::multiset** and **std::advance**

```
#include <Median.h>
```

Inheritance diagram for MedianMultisetAdvance:



Public Member Functions

- virtual bool add (int a)

  add new integer element
- virtual int middle () const

  return value of the middle element
- virtual size_t size () const

  return number of all elements
- virtual void clear ()

  empty container
- virtual const char ∗ name () const

  return string with container name

Static Public Member Functions

- static const char ∗ **Name** ()

Protected Attributes

- std::multiset< int > m_set

  multiset used for storage of the sorted elements

5.3.1 Detailed Description

MedianMultisetAdvance finds median using **std::multiset** and **std::advance**

MedianMultisetAdvance uses for storage container **std::multiset** all values are sorted by default in ascending order. Insert is done with the standard insert() routine of the multiset, obtaining then the middle element is done with std↩
::advance.
add() Average Complexity O(log n)
middle() Average Complexity O(n)

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

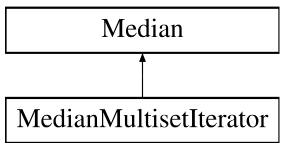
- Median.h
- Median.cpp

5.4 MedianMultisetIterator Class Reference

MedianMultisetIterator finds median using **std::multiset** and saved iterator to the middle element.

```
#include <Median.h>
```

Inheritance diagram for MedianMultisetIterator:



Public Member Functions

- virtual bool add (int a)

  add new integer element
- virtual int middle () const

  return value of the middle element
- virtual size_t size () const

  return number of all elements
- virtual void clear ()

  empty container
- virtual const char ∗ name () const

  return string with container name

Static Public Member Functions

- static const char ∗ **Name** ()

Protected Attributes

- std::multiset< int > m_set

  vector used for storage of the sorted elements
- std::multiset< int >::iterator m_it

  iterator to the middle element inside the set
- std::set< int >::size_type m_pos

  position of the middle element inside the set

5.4.1 Detailed Description

MedianMultisetIterator finds median using **std::multiset** and saved iterator to the middle element.

MedianMultisetIterator uses for storage container std::multiset, all values are sorted by default in ascending order. After insert **m_it** and **m_pos** variables are updated to point to the middle element.
add() Average Complexity O(log n)
middle() Average Complexity O(1)

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

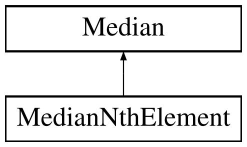
- Median.h
- Median.cpp

5.5 MedianNthElement Class Reference

MedianVector finds median using **std::vector>** and **std::nth_element**

`#include <Median.h>`

Inheritance diagram for MedianNthElement:



Public Member Functions

- virtual bool add (int a)

  add new integer element
- virtual int middle () const

  return value of the middle element
- virtual size_t size () const

  return number of all elements
- virtual void clear ()

  empty container
- virtual const char ∗ name () const

  return string with container name

Static Public Member Functions

- static const char ∗ **Name** ()

Protected Attributes

- std::vector< int > m_arr

  vector used for storage of the sorted elements

5.5.1 Detailed Description

MedianVector finds median using **std::vector>** and **std::nth_element**

MedianNthElement class implements Median with using for storage std::vector and std::nth_element to keep sorted the middle element only.
add() Average Complexity O(n)
middle() Average Complexity O(1)

There is also additional cost for memory copies.

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

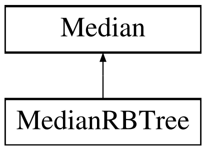
- Median.h
- Median.cpp

5.6 MedianRBTree Class Reference

MedianRBTree finds median using **RBTree**

```
#include <Median.h>
```

Inheritance diagram for MedianRBTree:



Public Member Functions

- virtual bool add (int a)

  add new integer element
- virtual int middle () const

  return value of the middle element
- virtual size_t size () const

  return number of all elements
- virtual void clear ()

  empty container
- virtual const char ∗ name () const

  return string with container name

Static Public Member Functions

- static const char ∗ **Name** ()

Protected Attributes

- RBTree m_bt

  RBTree container used for storage of the sorted elements.
- const char ∗ **N** = "n"

5.6.1 Detailed Description

MedianRBTree finds median using **RBTree**

MedianRBTree uses for storage container Red-Black tree, all values are sorted in ascending order. Allows obtaining of the middle value as start iterating from the root. Since Red-Black is a self balancing tree, the middle value would be very near to the root so the search has constant average complexity
add() Average Complexity O(log n)
middle() Average Complexity O(1)

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

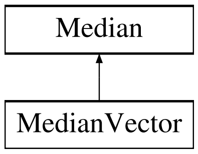
- Median.h
- Median.cpp

5.7 MedianVector Class Reference

MedianVector finds median using **std::vector** and **std::sort**

```
#include <Median.h>
```

Inheritance diagram for MedianVector:



Public Member Functions

- virtual bool add (int a)
  add new integer element
- virtual int middle () const
  return value of the middle element
- virtual size_t size () const
  return number of all elements
- virtual void clear ()
  empty container
- virtual const char ∗ name () const
  return string with container name

Static Public Member Functions

- static const char ∗ **Name** ()

Protected Attributes

- std::vector< int > m_arr
  vector used for storage of the sorted elements

5.7.1 Detailed Description

MedianVector finds median using **std::vector** and **std::sort**

MedianVector class implements Median with using for storage **std::vector** and **std::sort** to keep elements sorted.
add() Average Complexity O(n log n)
middle() Average Complexity O(1)

There is also additional cost for memory copies.

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

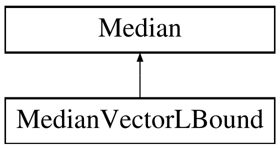
- Median.h
- Median.cpp

5.8 MedianVectorLBound Class Reference

MedianVectorLBound calculate median using **std::vector** and **std::lower_bound**

```
#include <Median.h>
```

Inheritance diagram for MedianVectorLBound:



Public Member Functions

- virtual bool add (int a)

  add new integer element
- virtual int middle () const

  return value of the middle element
- virtual size_t size () const

  return number of all elements
- virtual void clear ()

  empty container
- virtual const char ∗ name () const

  return string with container name

Static Public Member Functions

- static const char ∗ **Name** ()

5.8.1 Detailed Description

MedianVectorLBound calculate median using **std::vector** and **std::lower_bound**

MedianVectorLBound class implements Median with using for storage std::vector and std::lower_bound to sort only the last inserted element only. First the position of the inserted element is found with **std::lower_bound**, and then the new element is inserted there.
add() Average Complexity O(log n)
middle() Average Complexity O(1)

There is also additional cost for memory copies.

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

- Median.h
- Median.cpp

5.9 RBTree::Node Class Reference

Public Member Functions

- **Node** (const int &key, Node ∗parent=0)
- void **init** ()
- bool **isEmpty** () const
- int **numChildsLeft** () const
- int **numChildsRight** () const

Public Attributes

- Node ∗ **left**
- Node ∗ **right**
- Node ∗ **parent**
- int **key**
- int **type**
- int **nChilds**

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

- RBTree.h

5.10 RBTree Class Reference

Classes

- class Node

Public Types

- enum { **BLACK** = 0, **RED** = 1 }

Public Member Functions

- bool insert (const int &key)

  insert new element
- int size () const

  return number of all inserted elements
- bool clear ()

  empty tree
- int middleValue () const

  get the value of the middle element

Protected Member Functions

- void **rotateRight** (Node ∗node)
- void **rotateLeft** (Node ∗node)
- void **insertFixup** (Node ∗x)
- virtual Node ∗ **allocNode** (const int &key, Node ∗parent=0)
- virtual void **deleteNode** (Node ∗node)
- Node ∗ **findParent** (Node ∗node, const int &key) const
- Node ∗ **insertNode** (Node ∗node, const int &key)
- void **emptySubtree** (Node ∗node)

Protected Attributes

- Node ∗ root

  pointer to the root of the tree
- int count

  integer to keep the number of the inserted elements

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

- RBTree.h
- RBTree.cpp

Chapter 6

File Documentation

6.1 Median.h File Reference

Main declaration header for Median Inteface and several Median implemntations.

```
#include <vector>
#include <set>
#include <queue>
#include "RBTree.h"
```

Classes

- class Median

  Median class.
- class MedianVector

  *MedianVector finds median using **std::vector** and **std::sort***
- class MedianNthElement

  *MedianVector finds median using **std::vector**> and **std::nth_element***
- class MedianVectorLBound

  *MedianVectorLBound calculate median using **std::vector** and **std::lower_bound***
- class MedianMultisetAdvance

  *MedianMultisetAdvance finds median using **std::multiset** and **std::advance***
- class MedianMultisetIterator

  *MedianMultisetIterator finds median using **std::multiset** and saved iterator to the middle element.*
- class MedianRBTree

  *MedianRBTree finds median using **RBTree***
- class MedianMinMaxHeap

  *MedianMinMaxHeap finds median using two **std::priority_queue***

6.1.1 Detailed Description

Main declaration header for Median Inteface and several Median implemntations.

Following Median implementations are provided: MedianVector MedianNthElement MedianVectorLBound MedianMultisetAdvance MedianMultisetIterator MedianRBTree MedianMinMaxHeap

6.2 MedianFactory.h File Reference

Factory routines for Median.

```
#include "Median.h"
#include <memory>
#include <string>
```

Functions

- std::unique_ptr< Median > create_Median (const std::string &name)

  create a Median object

6.2.1 Detailed Description

Factory routines for Median.

6.2.2 Function Documentation

6.2.2.1 create_Median()

```
std::unique_ptr<Median> create_Median (
            const std::string & name )
```

create a Median object

create a Median object

Parameters

name	class name to create object of

Returns

null pointer if name not found, otherwise valid unique pointer

6.3 PerformanceTest.h File Reference

Performance tests for the Median Class.

```
#include "Median.h"
```

Functions

- bool performanceTestAllRandom (int N)

  Test with randomly generated values.
- bool performanceTestAllWorst (int N)

  Test with descending values.
- bool performanceTestAllRepeating (int N)

  Test with repeating values.

6.3.1 Detailed Description

Performance tests for the Median Class.

Tests can be run on Windows x86/x64 platforms.

6.3.2 Function Documentation

6.3.2.1 performanceTestAllRandom()

```
bool performanceTestAllRandom (
            int N )
```

Test with randomly generated values.

Performance test with randomly generated values, also checks if the returned values are correct

Parameters

N	number of the test values

Returns

True if sucessful

6.3.2.2 performanceTestAllRepeating()

```
bool performanceTestAllRepeating (
            int N )
```

Test with repeating values.

Performance test with repeating (0-9) values

Parameters

N	number of the test values

Returns

True if sucessful

6.3.2.3 performanceTestAllWorst()

```
bool performanceTestAllWorst (
            int N )
```

Test with descending values.

Performance test with linearly descending values, also checks if the returned values are correct

Parameters

N	number of the test values

Returns

True if sucessful

6.4 RBTree.h File Reference

multiset implementation of a Red-Black tree

Classes

- class RBTree
- class RBTree::Node

6.4.1 Detailed Description

multiset implementation of a Red-Black tree

Author

Anton Milev

Version

1.0

Date

October 2019

Similarily to std::multiset, this RB tree allows inserting of duplicate values. It is specially designed to find the median element in efficient way.

6.5 UnitTests.h File Reference

Provides unit tests for Median Class.

```
#include "Median.h"
```

Functions

- void **printc** (int color, const char ∗output,...)
- void **printResult** (const char ∗fname, bool bRes)
- int **test_median_basic** (Median &m)
- int **test_median_iter** (Median &m)
- int **test_median_random** (Median &m)
- int **unitTest** (const char ∗name)
- void **unitTestAll** ()

6.5.1 Detailed Description

Provides unit tests for Median Class.

Tests can be run on Windows x86/x64 platforms.

Index