

Introduction To Algorithms

Generated by Doxygen 1.8.1.2

Tue Jul 24 2012 11:36:15

Contents

1	Introduction to Algorithms	1
2	Data Structure Index	3
2.1	Data Structures	3
3	File Index	5
3.1	File List	5
4	Data Structure Documentation	7
4.1	Maxtag Struct Reference	7
4.1.1	Detailed Description	7
4.1.2	Field Documentation	7
4.1.2.1	maxlf	7
4.1.2.2	maxrt	7
4.1.2.3	maxsum	7
5	File Documentation	9
5.1	Chapter 2/Sort.h File Reference	9
5.1.1	Detailed Description	9
5.1.2	Function Documentation	9
5.1.2.1	insertion_sort	9
5.1.2.2	merge	10
5.1.2.3	merge_sort	10
5.1.2.4	reverse_insertion_sort	10
5.1.2.5	selection_sort	10
5.2	Chapter 4/matrix.h File Reference	10
5.2.1	Detailed Description	10
5.2.2	Function Documentation	11
5.2.2.1	square_matrix_multiply	11
5.3	Chapter 4/maxarray.h File Reference	11
5.3.1	Detailed Description	11
5.3.2	Typedef Documentation	12
5.3.2.1	Max	12

5.3.3	Function Documentation	12
5.3.3.1	max_crossing_sub	12
5.3.3.2	max_subarray	12
5.3.3.3	max_subarray_linear	12
5.3.3.4	max_subarray_quadratic	13
5.4	Chapter 5/random.h File Reference	13
5.4.1	Detailed Description	13
5.4.2	Function Documentation	13
5.4.2.1	randomized_hire_assistant	13
5.5	Chapter 6/heap.h File Reference	13
5.5.1	Detailed Description	14
5.5.2	Function Documentation	14
5.5.2.1	build_max_heap	14
5.5.2.2	heapsort	14
5.5.2.3	max_heapify	14
5.5.2.4	min_heapify	14
5.6	Chapter 7/quicksort.h File Reference	15
5.6.1	Detailed Description	15
5.6.2	Function Documentation	15
5.6.2.1	partition	15
5.6.2.2	quicksort	15
5.6.2.3	quicksort_algo	16
5.6.2.4	randomized_partition	16
5.6.2.5	randomized_quicksort	16
5.6.2.6	randomized_quicksort_algo	16
5.7	Chapter 8/linear.h File Reference	17
5.7.1	Detailed Description	17
5.7.2	Function Documentation	17
5.7.2.1	countingSort	17
5.7.2.2	findMaxForCountingSort	17
5.8	Chapter 9/medians.h File Reference	17
5.8.1	Detailed Description	18
5.8.2	Function Documentation	18
5.8.2.1	randomized_select	18
5.8.2.2	randomized_select_algo	18
5.8.2.3	select	19
5.8.2.4	select_algo	19

Chapter 1

Introduction to Algorithms

The Algorithms are all written in C.

Compilation with these flags is error/warning free:

```
-Wall -Wextra -std=c99 -pedantic
```

Algorithms not yet implemented

- Chapter 4
 - Strassen multiplication
 - Non square matrix multiplication
- Chapter 6:
 - Priority queue
- Chapter 7:
 - Hoare Partition
- Chapter 8:
 - Radix sort
 - Bucket sort

Side Notes

Doxygen documentation is being written.

Chapter 2

Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

Maxtag	7
----------------------------------	---

Chapter 3

File Index

3.1 File List

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

Chapter 2/ Sort.h	
Sorting Algorithms	9
Chapter 4/ matrix.h	
Matrix Multiplication	10
Chapter 4/ maxarray.h	
Maximum subarray Algorithms	11
Chapter 5/ random.h	
Randomize Array	13
Chapter 6/ heap.h	
Heap/Heapsort related algorithms	13
Chapter 7/ quicksort.h	
Quicksort Algorithms	15
Chapter 8/ linear.h	
Sorting in linear time algorithms	17
Chapter 9/ medians.h	
Select element in array according to rank in linear time	17

Chapter 4

Data Structure Documentation

4.1 Maxtag Struct Reference

```
#include <maxarray.h>
```

Data Fields

- int [maxlf](#)
- int [maxrt](#)
- int [maxsum](#)

4.1.1 Detailed Description

<Struct containing 2 indexes and the sum of array[maxlf] to array[maxrt]

4.1.2 Field Documentation

4.1.2.1 int maxlf

Left index of the maximum subarray

4.1.2.2 int maxrt

Right index of the maximum subarray

4.1.2.3 int maxsum

Sum of the elements from array[maxlf] to [maxrt]

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

- Chapter 4/[maxarray.h](#)

Chapter 5

File Documentation

5.1 Chapter 2/Sort.h File Reference

Sorting Algorithms.

Functions

- void [insertion_sort](#) (int *array, int size)
Insertion Sort.
- void [reverse_insertion_sort](#) (int *array, int size)
Insertion Sort in decreasing order.
- void [selection_sort](#) (int *array, int size)
Selection Sort.
- void [merge](#) (int *array, int l, int m, int h)
Merges and sorts 2 subarrays.
- void [merge_sort](#) (int *array, int l, int h)
Merge Sort.

5.1.1 Detailed Description

Sorting Algorithms.

Author

Ruggero Dalo

5.1.2 Function Documentation

5.1.2.1 void [insertion_sort](#) (int * *array*, int *size*)

Insertion Sort.

Parameters

<i>array</i>	Array to be sorted
<i>size</i>	Elements in the array

5.1.2.2 void merge (int * array, int l, int m, int h)

Merges and sorts 2 subarrays.

Parameters

<i>array</i>	Array to be sorted
<i>l</i>	Low/Left index
<i>m</i>	Medium/Middle index
<i>h</i>	High/Right index

5.1.2.3 void merge_sort (int * array, int l, int h)

Merge Sort.

Parameters

<i>array</i>	Array to be sorted
<i>l</i>	Low/Left index
<i>h</i>	High/Right index

5.1.2.4 void reverse_insertion_sort (int * array, int size)

Insertion Sort in decreasing order.

Parameters

<i>array</i>	Array to be sorted
<i>size</i>	Elements in the array

5.1.2.5 void selection_sort (int * array, int size)

Selection Sort.

Parameters

<i>array</i>	Array to be sorted
<i>size</i>	Elements in the array

5.2 Chapter 4/matrix.h File Reference

Matrix Multiplication.

Functions

- void [square_matrix_multiply](#) (int *A, int *B, int *C, int size)
Square matrix multiplication.

5.2.1 Detailed Description

Matrix Multiplication.

Author

Ruggero Dalo

5.2.2 Function Documentation

5.2.2.1 void square_matrix_multiply (int * *A*, int * *B*, int * *C*, int *size*)

Square matrix multiplication.

Parameters

<i>A</i>	First Matrix
<i>B</i>	Second Matrix
<i>C</i>	Multiplication result matrix
<i>size</i>	Number of colomns/rows of matrix A and B

5.3 Chapter 4/maxarray.h File Reference

Maximum subarray Algorithms.

```
#include <stdio.h>
#include <stdlib.h>
```

Data Structures

- struct [Maxtag](#)

Typedefs

- typedef struct [Maxtag](#) [Max](#)

Functions

- [Max](#) * [max_crossing_sub](#) (int *array, int l, int h)
Finds maximum sum of elements crossing the (l+h)/2 index.
- [Max](#) * [max_subarray](#) (int *array, int l, int h)
Recursive funtion.
- void [max_subarray_quadratic](#) (int *array, int size, [Max](#) *result)
Brute force solution to the max sub array problem.
- void [max_subarray_linear](#) (int *array, int size, [Max](#) *result)
Kadane's algorithm.

5.3.1 Detailed Description

Maximum subarray Algorithms.

Author

Ruggero Dalo

5.3.2 Typedef Documentation

5.3.2.1 typedef struct Maxtag Max

<Struct containing 2 indexes and the sum of array[maxlf] to array[maxrt] Name of the struct:

5.3.3 Function Documentation

5.3.3.1 Max* max_crossing_sub (int * array, int l, int h)

Finds maximum sum of elements crossing the (l+h)/2 index.

Parameters

<i>array</i>	Integer Array
<i>l</i>	Low/Left index
<i>h</i>	High/Right index

Returns

Pointer to a Max struct containing index and sum of the maximum subarray found in array[l]-array[h]

Warning

The Returned memory address must be free'd to avoid memory leaks

5.3.3.2 Max* max_subarray (int * array, int l, int h)

Recursive funtion.

Parameters

<i>array</i>	Integer Array
<i>l</i>	Low/Left index
<i>h</i>	High/Right index

Returns

Pointer to a Max struct containing index and sum of the maximum subarray found in array[l]-array[h]

Warning

The Returned memory address must be free'd to avoid memory leaks

5.3.3.3 void max_subarray_linear (int * array, int size, Max * result)

Kadane's algorithm.

Parameters

<i>array</i>	Integer Array
<i>size</i>	Elements in the array
<i>*result</i>	Pointer to an already allocated structure where the result is stored

5.3.3.4 void max_subarray_quadratic (int * array, int size, Max * result)

Brute force solution to the max sub array problem.

Parameters

<i>array</i>	Integer Array
<i>size</i>	Elements in the array
<i>*result</i>	Pointer to an already allocated structure where the result is stored

5.4 Chapter 5/random.h File Reference

Randomize Array.

```
#include <stdio.h>
#include <time.h>
#include <stdlib.h>
```

Functions

- void [randomized_hire_assistant](#) (int *array, int size)

5.4.1 Detailed Description

Randomize Array.

Author

Ruggero Dalo

5.4.2 Function Documentation

5.4.2.1 void randomized_hire_assistant (int * array, int size)

A new array (B) will be initialized with random values. B[i] = key, array[i] = value. B is then sorted and array is modified accordingly.

Parameters

<i>array</i>	input array
<i>size</i>	Elements in the array

5.5 Chapter 6/heap.h File Reference

Heap/Heapsort related algorithms.

```
#include <stdio.h>
```

Functions

- void [max_heapify](#) (int *array, int heap_size, int index)

- void [min_heapify](#) (int *array, int heap_size, int index)
- void [build_max_heap](#) (int *array, int size)
- void [heapsort](#) (int *array, int size)

5.5.1 Detailed Description

Heap/Heapsort related algorithms.

Author

Ruggero Dalo

5.5.2 Function Documentation

5.5.2.1 void build_max_heap (int * array, int size)

Starting at the last parent node it heapifies the array

Parameters

<i>array</i>	Array to be Heapified
<i>size</i>	Elements in the array

5.5.2.2 void heapsort (int * array, int size)

Heapsort

Parameters

<i>array</i>	Array to be Heapified
<i>size</i>	Elements in the array

5.5.2.3 void max_heapify (int * array, int heap_size, int index)

Checks parent and children nodes and the biggest element is saved in the parent node. Recursively calls it self on the exchanged children.

Parameters

<i>array</i>	Array to be Heapified
<i>heap_size</i>	Heap size
<i>index</i>	Position in the array

5.5.2.4 void min_heapify (int * array, int heap_size, int index)

Checks parent and children nodes and the smallest element is saved in the parent node. Recursively calls it self on the exchanged children.

Parameters

<i>array</i>	Array to be Heapified
<i>heap_size</i>	Heap size
<i>index</i>	Position in the array

5.6 Chapter 7/quicksort.h File Reference

Quicksort Algorithms.

```
#include <stdio.h>
#include <time.h>
#include <stdlib.h>
```

Functions

- int [partition](#) (int *array, int l, int h)
Partions the array around the pivot.
- void [quicksort_algo](#) (int *array, int l, int h)
Quicksort.
- void [quicksort](#) (int *array, int size)
Quicksort wrapper funtion.
- int [randomized_partition](#) (int *array, int l, int h)
Partions the array around a random pivot.
- void [randomized_quicksort_algo](#) (int *array, int l, int h)
Randomized Quicksort.
- void [randomized_quicksort](#) (int *array, int size)
Randomized Quicksort wrapper funtion.

5.6.1 Detailed Description

Quicksort Algorithms.

Author

Ruggero Dalo

5.6.2 Function Documentation

5.6.2.1 int partition (int * array, int l, int h)

Partions the array around the pivot.

Parameters

<i>array</i>	Array to be sorted
<i>l</i>	Low/Left index
<i>h</i>	High/Right index

Returns

Returns index of the pivot

5.6.2.2 void quicksort (int * array, int size)

Quicksort wrapper funtion.

Parameters

<i>array</i>	Array to be sorted
<i>size</i>	Elements in the array

5.6.2.3 void quicksort_algo (int * array, int l, int h)

Quicksort.

Parameters

<i>array</i>	Array to be sorted
<i>l</i>	Low/Left index
<i>h</i>	High/Right index

Note

Using the wrapper funtion, quicksort, is recommended.

5.6.2.4 int randomized_partition (int * array, int l, int h)

Partions the array around a random pivot.

Parameters

<i>array</i>	Array to be sorted
<i>l</i>	Low/Left index
<i>h</i>	High/Right index

Returns

Returns index of the pivot

5.6.2.5 void randomized_quicksort (int * array, int size)

Randomized Quicksort wrapper funtion.

Parameters

<i>array</i>	Array to be sorted
<i>size</i>	Elements in the array

5.6.2.6 void randomized_quicksort_algo (int * array, int l, int h)

Randomized Quicksort.

Parameters

<i>array</i>	Array to be sorted
<i>l</i>	Low/Left index
<i>h</i>	High/Right index

Note

Using the wrapper function, `randomized_quicksort`, is recommended.

5.7 Chapter 8/linear.h File Reference

Sorting in linear time algorithms.

```
#include <stdio.h>
```

Functions

- `int findMaxForCountingSort (int *array, int size)`
- `void countingSort (int *array, int *sorted, int size, int max)`

5.7.1 Detailed Description

Sorting in linear time algorithms.

Author

Ruggero Dalo

5.7.2 Function Documentation

5.7.2.1 `void countingSort (int * array, int * sorted, int size, int max)`

Sorts the array with help from an auxiliary array of size equal to the maximum element in array

Parameters

<i>array</i>	Array to be sorted
<i>sorted</i>	Pointer to an already initialized array of size equal to the one to be sorted
<i>size</i>	Elements in the array
<i>max</i>	Maximum element in the array

5.7.2.2 `int findMaxForCountingSort (int * array, int size)`

Finds maximum integer in the array

Parameters

<i>array</i>	Array to be scanned
<i>size</i>	Elements in the array

Returns

Maximum element

5.8 Chapter 9/medians.h File Reference

Select element in array according to rank in linear time.

Functions

- int [select_algo](#) (int *array, int l, int h, int i)
- int [select](#) (int *array, int size, int rank)
- int [randomized_select_algo](#) (int *array, int l, int h, int i)
- int [randomized_select](#) (int *array, int size, int rank)

5.8.1 Detailed Description

Select element in array according to rank in linear time.

Author

Ruggero Dalo

5.8.2 Function Documentation

5.8.2.1 int randomized_select (int * array, int size, int rank)

Rank 1 = smallest element in the array

Rank Size = biggest element in the array

Rank < 1 or Rank > size is not defined.

Parameters

<i>array</i>	Array to be scanned
<i>size</i>	Elements in the array
<i>rank</i>	Desired rank

Returns

Integer with the specified rank

5.8.2.2 int randomized_select_algo (int * array, int l, int h, int i)

Uses randomized_partition algorithm from quicksort

Parameters

<i>array</i>	Array to be scanned
<i>l</i>	Low/Left index
<i>h</i>	High/Right index
<i>i</i>	Desired rank

Returns

Integer of rank i

Note

Using the wrapper funtion, randomized_select, is recommended.

5.8.2.3 `int select (int * array, int size, int rank)`

Rank 1 = smallest element in the array

Rank Size = biggest element in the array

Rank < 1 or Rank > size is not defined.

Parameters

<i>array</i>	Array to be scanned
<i>size</i>	Elements in the array
<i>rank</i>	Desired rank

Returns

Integer with the specified rank

5.8.2.4 `int select_algo (int * array, int l, int h, int i)`

Uses partion algorithm from quicksort

Parameters

<i>array</i>	Array to be scanned
<i>l</i>	Low/Left index
<i>h</i>	High/Right index
<i>i</i>	Desired rank

Returns

Integer of rank i

Note

Using the wrapper funtion, select, is recommended.

Index

- build_max_heap
 - heap.h, 14
- Chapter 2/Sort.h, 9
- Chapter 4/matrix.h, 10
- Chapter 4/maxarray.h, 11
- Chapter 5/random.h, 13
- Chapter 6/heap.h, 13
- Chapter 7/quicksort.h, 15
- Chapter 8/linear.h, 17
- Chapter 9/medians.h, 17
- countingSort
 - linear.h, 17
- findMaxForCountingSort
 - linear.h, 17
- heap.h
 - build_max_heap, 14
 - heapsort, 14
 - max_heapify, 14
 - min_heapify, 14
- heapsort
 - heap.h, 14
- insertion_sort
 - Sort.h, 9
- linear.h
 - countingSort, 17
 - findMaxForCountingSort, 17
- matrix.h
 - square_matrix_multiply, 11
- Max
- max_crossing_sub
 - maxarray.h, 12
- max_heapify
 - heap.h, 14
- max_subarray
 - maxarray.h, 12
- max_subarray_linear
 - maxarray.h, 12
- max_subarray_quadratic
 - maxarray.h, 12
- maxarray.h
 - Max, 12
 - max_crossing_sub, 12
 - max_subarray, 12
 - max_subarray_linear, 12
 - max_subarray_quadratic, 12
- maxxf
 - Maxtag, 7
- maxrt
 - Maxtag, 7
- maxsum
 - Maxtag, 7
- Maxtag, 7
 - maxxf, 7
 - maxrt, 7
 - maxsum, 7
- medians.h
 - randomized_select, 18
 - randomized_select_algo, 18
 - select, 18
 - select_algo, 19
- merge
 - Sort.h, 9
- merge_sort
 - Sort.h, 10
- min_heapify
 - heap.h, 14
- partition
 - quicksort.h, 15
- quicksort
 - quicksort.h, 15
- quicksort.h
 - partition, 15
 - quicksort, 15
 - quicksort_algo, 16
 - randomized_partition, 16
 - randomized_quicksort, 16
 - randomized_quicksort_algo, 16
- quicksort_algo
 - quicksort.h, 16
- random.h
 - randomized_hire_assistant, 13
- randomized_hire_assistant
 - random.h, 13
- randomized_partition
 - quicksort.h, 16
- randomized_quicksort
 - quicksort.h, 16
- randomized_quicksort_algo
 - quicksort.h, 16
- randomized_select
 - medians.h, 18

- randomized_select_algo
 - medians.h, [18](#)
- reverse_insertion_sort
 - Sort.h, [10](#)
- select
 - medians.h, [18](#)
- select_algo
 - medians.h, [19](#)
- selection_sort
 - Sort.h, [10](#)
- Sort.h
 - insertion_sort, [9](#)
 - merge, [9](#)
 - merge_sort, [10](#)
 - reverse_insertion_sort, [10](#)
 - selection_sort, [10](#)
- square_matrix_multiply
 - matrix.h, [11](#)