Introduction To Algorithms

Generated by Doxygen 1.8.1.2

Tue Jul 24 2012 11:36:15

Contents

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:

Chapter 3

File Index

3.1 File List

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

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 indexs 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

array	Array to be sorted
size	Elements in the array

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

Merges and sorts 2 subarrays.

Parameters

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

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

Merge Sort.

Parameters

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

5.1.2.4 void reverse_insertion_sort (int * *array*, int *size*)

Insertion Sort in decreasing order.

Parameters

array	Array to be sorted
size	Elements in the array

5.1.2.5 void selection_sort (int * *array*, int *size*)

Selection Sort.

Parameters

array	Array to be sorted
size	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

A	First Matrix
B	Second Matrix
C	Multiplication result matrix
size	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 indexs 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

array	Integer Array
l	Low/Left index
h	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

array	Integer Array
l	Low/Left index
h	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

array	Integer Array
size	Elements in the array
**result*	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

array	Integer Array
size	Elements in the array
*result	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

array	input array
size	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 heapifyes the array

Parameters

array	Array to be Heapifyed
size	Elements in the array

5.5.2.2 void heapsort (int * *array*, int *size*)

Heapsort

Parameters

array	Array to be Heapifyed
size	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

array	Array to be Heapifyed
heap_size	Heap size
index	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

array	Array to be Heapifyed
heap_size	Heap size
index	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

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

Returns

Returns index of the pivot

5.6.2.2 void quicksort (int * *array*, int *size*)

Quicksort wrapper funtion.

Parameters

array	Array to be sorted
size	Elements in the array

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

Quicksort.

Parameters

array	Array to be sorted
l	Low/Left index
h	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

array	Array to be sorted
l	Low/Left index
h	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

array	Array to be sorted
size	Elements in the array

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

Randomized Quicksort.

Parameters

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

Note

Using the wrapper funtion, 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 tha array with help from an auxiliary array of size equal to the maximum element in array

Parameters

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

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

Finds maximum integer in the array

Parameters

array	Array to be scanned
size	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

array	Array to be scanned
size	Elements in the array
rank	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_partion algorithm from quicksort

Parameters

array	Array to be scanned
l	Low/Left index
h	High/Right index
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

array	Array to be scanned
size	Elements in the array
rank	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

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

Returns

Integer of rank i

Note

Using the wrapper funtion, select, is recommended.

Index