## Matrix Test with Strasson Algorithm

Generated by Doxygen 1.7.6.1

Wed Sep 16 2015 14:03:07

## **Contents**

1	Data	Structi	ure Index		1
	1.1	Data S	tructures		1
2	File	Index			3
	2.1	File Lis	st		3
3	Data	Structi	ure Docur	mentation	5
	3.1	matrix	Struct Ref	ference	5
		3.1.1	Field Doo	cumentation	5
			3.1.1.1	a	5
	3.2	test_t s	Struct Refe	erence	5
		3.2.1	Field Doo	cumentation	6
			3.2.1.1	a	6
			3.2.1.2	b	6
			3.2.1.3	c	6
			3.2.1.4	d	6
			3.2.1.5	$n\ldots\ldots\ldots\ldots\ldots$	6
			3.2.1.6	rn	6
4	File	Docum	entation		7
	4.1	src/arr.	c File Refe	erence	7
		4.1.1	Function	Documentation	8
			4.1.1.1	arr_add	8
			4.1.1.2	arr_mul	8
			4.1.1.3	arr_sub	8
			4.1.1.4	comp arr	9

ii CONTENTS

		4.1.1.5	copy_arr
		4.1.1.6	del_arr
		4.1.1.7	dump_arr
		4.1.1.8	new_arr
		4.1.1.9	rand_arr
		4.1.1.10	round_up_power_of_two
		4.1.1.11	set_ones_arr
		4.1.1.12	set_seqs_arr
4.2	src/arr.	h File Refe	erence
	4.2.1	Define De	ocumentation
		4.2.1.1	MAX_DATA_VALUE
		4.2.1.2	MAX_DIM
		4.2.1.3	MIN_DATA_VALUE
	4.2.2	Function	Documentation
		4.2.2.1	arr_add
		4.2.2.2	arr_mul
		4.2.2.3	arr_sub
		4.2.2.4	comp_arr
		4.2.2.5	copy_arr
		4.2.2.6	del_arr
		4.2.2.7	del_arr_p
		4.2.2.8	dump_arr
		4.2.2.9	new_arr
		4.2.2.10	new_arr_p
		4.2.2.11	rand_arr
		4.2.2.12	round_up_power_of_two
		4.2.2.13	set_ones_arr
		4.2.2.14	set_seqs_arr
		4.2.2.15	sub_arr
4.3	src/foo	.c File Refe	erence
	4.3.1	Function	Documentation
		4.3.1.1	main
4.4	src/ma	trix.c File F	Reference
	4.4.1	Function	Documentation

CONTENTS iii

		4.4.1.1	del_matrix
		4.4.1.2	new_matrix
4.5	src/ma	ıtrix.h File F	Reference
	4.5.1	Function	Documentation
		4.5.1.1	del_matrix
		4.5.1.2	new_matrix
4.6	src/stra	assen.c File	e Reference
	4.6.1	Function	Documentation
		4.6.1.1	arr_qsplit
		4.6.1.2	arr_qsplit_recall 20
		4.6.1.3	s_add
		4.6.1.4	s_mul
		4.6.1.5	s_sub
	4.6.2	Variable I	Documentation
		4.6.2.1	g_break
4.7	src/stra	assen.h Fil	e Reference
	4.7.1	Define Do	ocumentation
		4.7.1.1	a11
		4.7.1.2	a12
		4.7.1.3	a21
		4.7.1.4	a22
		4.7.1.5	b11
		4.7.1.6	b12
		4.7.1.7	b21
		4.7.1.8	b22
		4.7.1.9	c11
		4.7.1.10	c12
		4.7.1.11	c21
		4.7.1.12	c22
		4.7.1.13	d11
		4.7.1.14	d12
		4.7.1.15	d21
		4.7.1.16	d22
	4.7.2	Function	Documentation

iv CONTENTS

		4.7.2.1	arr_qsplit	24
		4.7.2.2	arr_qsplit_recall	24
		4.7.2.3	s_add	24
		4.7.2.4	s_mul	25
		4.7.2.5	s_sub	25
	4.7.3	Variable I	Documentation	25
		4.7.3.1	g_break	25
4.8	src/tes	t.c File Ref	ference	25
	4.8.1	Function	Documentation	27
		4.8.1.1	del_test	27
		4.8.1.2	dump_result	27
		4.8.1.3	init_test_data	27
		4.8.1.4	new_test	27
		4.8.1.5	round_down_power_of_two	28
		4.8.1.6	test	28
		4.8.1.7	test_case	28
		4.8.1.8	test_check_result	29
		4.8.1.9	test_normal_multiply	29
		4.8.1.10	test_strassen_multiply	29
		4.8.1.11	test_tweak_breaks	29
		4.8.1.12	test_valid_data	30
	4.8.2	Variable I	Documentation	30
		4.8.2.1	g_pattern	30
4.9	src/tes	t.h File Re	ference	30
	4.9.1	Enumera	tion Type Documentation	32
		4.9.1.1	OPS	32
		4.9.1.2	PATTERN_MODE	32
	4.9.2	Function	Documentation	32
		4.9.2.1	del_test	32
		4.9.2.2	dump_result	33
		4.9.2.3	init_test_data	33
		4.9.2.4	new_test	33
		4.9.2.5	round_down_power_of_two	33
		4.9.2.6	test	34

CONTENTS	V

	4.9.2.7	test_case	34
	4.9.2.8	test_check_result	34
	4.9.2.9	test_normal_multiply	35
	4.9.2.10	test_strassen_multiply	35
	4.9.2.11	test_tweak_breaks	35
	4.9.2.12	test_valid_data	36
4.9.3	Variable [	Documentation	36
	4.9.3.1	g pattern	36

# **Chapter 1**

## **Data Structure Index**

1	1	Data	<b>Stri</b>	ıctı	Irac

Here are the data structures with brief descriptions:	
matrix	
test t	

# **Chapter 2**

## File Index

## 2.1 File List

Here is a list of all files with brief descriptions:

src/arr.c																			7
src/arr.h																			11
src/foo.c																			15
src/matrix.c																			17
src/matrix.h																			18
src/strassen.	С																		19
src/strassen.	h																		22
src/test.c .																			25
src/test.h .																			30

4 File Index

## **Chapter 3**

## **Data Structure Documentation**

## 3.1 matrix Struct Reference

#include <matrix.h>

## **Data Fields**

• double \*\* a

## 3.1.1 Field Documentation

3.1.1.1 double \*\* matrix::a

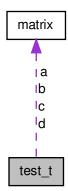
The documentation for this struct was generated from the following file:

• src/matrix.h

## 3.2 test\_t Struct Reference

#include <test.h>

Collaboration diagram for test\_t:



## **Data Fields**

- int n
- int rn
- matrix \* a
- matrix \* b
- matrix \* c
- matrix \* d
- 3.2.1 Field Documentation
- 3.2.1.1 matrix\* test\_t::a
- 3.2.1.2 matrix\* test\_t::b
- 3.2.1.3 matrix\* test\_t::c
- 3.2.1.4 matrix\* test\_t::d
- 3.2.1.5 int test\_t::n
- 3.2.1.6 int test\_t::rn

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

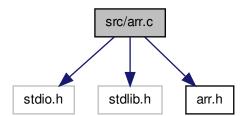
• src/test.h

## **Chapter 4**

## **File Documentation**

## 4.1 src/arr.c File Reference

 $\# include < \!\! stdio.h \!\! > \# include < \!\! stdlib.h \!\! > \# include \mbox{"arr.h"} \times \mbox{Include dependency graph for arr.c:}$ 



## **Functions**

- int round\_up\_power\_of\_two (int n)
   round up a number to power of two.
- void arr\_mul (double \*\*a, double \*\*b, double \*\*c, int n) perform matrices multiply, C = A mul B.
- void arr\_add (double \*\*a, double \*\*b, double \*\*c, int n)
   perform matrices addition, C = A + B.
- void arr\_sub (double \*\*a, double \*\*b, double \*\*c, int n)
   perform matrices subtraction, C = A + B.

File Documentation

double \*\* new\_arr (int m, int n)
 create a new array with dimension m by n.

void del\_arr (double \*\*r, int m)

release the resource of an array.

void dump\_arr (double \*\*r, int m, int n)
 dump the content of an array

void copy\_arr (double \*\*dest, double \*\*src, int m, int n)
 copy the content of an array to another one.

void rand\_arr (double \*\*r, int m, int n)
 assign random data to an array.

• void set\_ones\_arr (double \*\*r, int m, int n)

assign all 1's data to an array.void set\_seqs\_arr (double \*\*r, int m, int n)

assign sequential data to an array, i.e.

• int comp\_arr (double \*\*a, double \*\*b, int m, int n) compare the content of two arrays.

#### 4.1.1 Function Documentation

4.1.1.1 void arr\_add ( double 
$$** a$$
, double  $** b$ , double  $** c$ , int  $n$  )

perform matrices addition, C = A + B.

#### **Parameters**

8

а	the input matrix A
b	the input matrix B
С	the result of matrices operation.
n	the dimension of input/output matrices

4.1.1.2 void arr\_mul ( double 
$$** a$$
, double  $** b$ , double  $** c$ , int  $n$ )

perform matrices multiply, C = A mul B.

### **Parameters**

а	the input matrix A
b	the input matrix B
С	the result of matrices multiply operation.
n	the dimension of input/output matrices

4.1.1.3 void arr\_sub ( double 
$$** a$$
, double  $** b$ , double  $** c$ , int  $n$ )

perform matrices subtraction, C = A + B.

а	the input matrix A
b	the input matrix B
С	the result of matrices operation.
n	the dimension of input/output matrices

4.1.1.4 int comp\_arr ( double \*\* a, double \*\* b, int m, int n)

compare the content of two arrays.

#### **Parameters**

а	the pointer to array A.
b	the pointer to array B.
m	the row size of the arrays.
n	the column size of the arrays.

#### Returns

0 indicates A and B are the same. 1 inticates A and B are different.

4.1.1.5 void copy\_arr ( double \*\* dest, double \*\* src, int m, int n)

copy the content of an array to another one.

## Parameters

dest	the pointer to the destination array.
src	the pointer to the source array.
m	the row size of the arrays.
п	the column size of the arrays.

4.1.1.6 void del\_arr ( double \*\* r, int m )

release the resource of an array.

### Parameters

r	the pointer to an array.
m	the row size of an array.

4.1.1.7 void dump\_arr ( double \*\* r, int m, int n)

dump the content of an array

r	the pointer to the array
m	the row size of the array
n	the column size of the array

## 4.1.1.8 double\*\* new\_arr ( int *m*, int *n* )

create a new array with dimension m by n.

#### **Parameters**

т	the row size of an array.
n	the column size of an array.

#### Returns

an array with dimension m by n.

#### 4.1.1.9 void rand\_arr ( double \*\* r, int m, int n)

assign random data to an array.

## Parameters

r	the pointer to an array.
т	the row size of the array.
n	the column size of the array.

## 4.1.1.10 int round\_up\_power\_of\_two ( int n )

round up a number to power of two.

it's used to expand the array with its dimension to be power-of-two.

#### **Parameters**

n	the number to be rounded-up

## Returns

the rounded-up number

## 4.1.1.11 void set\_ones\_arr ( double \*\* r, int m, int n )

assign all 1's data to an array.

	r	the pointer to an array.
ĺ	т	the row size of the array.
ĺ	n	the column size of the array.

4.1.1.12 void set\_seqs\_arr ( double \*\* r, int m, int n )

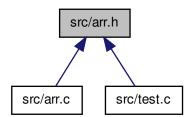
assign sequential data to an array, i.e. all rows contains data  $\{0, 1, 2, ..., n-1\}$ .

#### **Parameters**

r	the pointer to an array.
m	the row size of the array.
n	the column size of the array.

## 4.2 src/arr.h File Reference

This graph shows which files directly or indirectly include this file:



## **Defines**

- #define MAX\_DIM 10000
- #define MAX\_DATA\_VALUE 46340
- #define MIN\_DATA\_VALUE -46340

## **Functions**

• int round\_up\_power\_of\_two (int n)

12 File Documentation

```
round up a number to power of two.
    • void arr_mul (double **a, double **b, double **c, int n)
          perform matrices multiply, C = A mul B.

    void arr add (double **a, double **b, double **c, int n)

          perform matrices addition, C = A + B.

    void arr_sub (double **a, double **b, double **c, int n)

          perform matrices subtraction, C = A + B.
    • double ** new_arr (int m, int n)
          create a new array with dimension m by n.
    void del_arr (double **r, int m)
          release the resource of an array.
    double ** new_arr_p (int m)

    void del arr p (double **r)

    void dump_arr (double **r, int m, int n)

          dump the content of an array

    void copy_arr (double **dest, double **src, int m, int n)

          copy the content of an array to another one.
    • void rand_arr (double **r, int m, int n)
          assign random data to an array.
    • void set_ones_arr (double **r, int m, int n)
          assign all 1's data to an array.
    • void set_seqs_arr (double **r, int m, int n)
          assign sequential data to an array, i.e.

    void sub_arr (double **a, double **b, double **c, int m, int n)

    int comp_arr (double **a, double **b, int m, int n)

          compare the content of two arrays.
4.2.1 Define Documentation
4.2.1.1 #define MAX_DATA_VALUE 46340
4.2.1.2 #define MAX DIM 10000
4.2.1.3 #define MIN DATA VALUE -46340
4.2.2 Function Documentation
```

#### **Parameters**

perform matrices addition, C = A + B.

а	the input matrix A
b	the input matrix B
С	the result of matrices operation.
n	the dimension of input/output matrices Generated on Wed Sep 16 2015 14:03:07 for Matrix Test with Strasson Algorithm by Doxygen

4.2.2.1 void arr\_add ( double \*\* a, double \*\* b, double \*\* c, int n)

4.2.2.2 void arr\_mul ( double \*\* a, double \*\* b, double \*\* c, int n )

perform matrices multiply, C = A mul B.

#### **Parameters**

	а	the input matrix A
	b	the input matrix B
	С	the result of matrices multiply operation.
Γ	n	the dimension of input/output matrices

4.2.2.3 void arr\_sub ( double \*\* a, double \*\* b, double \*\* c, int n)

perform matrices subtraction, C = A + B.

#### **Parameters**

а	the input matrix A
b	the input matrix B
С	the result of matrices operation.
п	the dimension of input/output matrices

4.2.2.4 int comp\_arr ( double \*\* a, double \*\* b, int m, int n)

compare the content of two arrays.

### **Parameters**

а	the pointer to array A.
b	the pointer to array B.
т	the row size of the arrays.
n	the column size of the arrays.

#### Returns

0 indicates A and B are the same. 1 inticates A and B are different.

4.2.2.5 void copy\_arr ( double \*\* dest, double \*\* src, int m, int n)

copy the content of an array to another one.

### **Parameters**

dest	the pointer to the destination array.
src	the pointer to the source array.
m	the row size of the arrays.
n	the column size of the arrays.

Generated on Wed Sep 16 2015 14:03:07 for Matrix Test with Strasson Algorithm by Doxygen

4.2.2.6 void del\_arr ( double \*\* r, int m )

release the resource of an array.

#### **Parameters**

r	the pointer to an array.
m	the row size of an array.

```
4.2.2.7 void del_arr_p ( double ** r )
```

4.2.2.8 void dump\_arr ( double \*\* r, int m, int n)

dump the content of an array

#### **Parameters**

r	the pointer to the array
m	the row size of the array
n	the column size of the array

4.2.2.9 double\*\* new\_arr ( int *m*, int *n* )

create a new array with dimension m by n.

#### **Parameters**

т	the row size of an array.
n	the column size of an array.

#### Returns

an array with dimension m by n.

```
4.2.2.10 double** new_arr_p ( int m )
```

4.2.2.11 void rand\_arr ( double \*\* r, int m, int n)

assign random data to an array.

r	the pointer to an array.
m	the row size of the array.
n	the column size of the array.

4.2.2.12 int round\_up\_power\_of\_two (int n)

round up a number to power of two.

it's used to expand the array with its dimension to be power-of-two.

#### **Parameters**

n	the number to be rounded-up

#### **Returns**

the rounded-up number

4.2.2.13 void set\_ones\_arr ( double \*\* r, int m, int n)

assign all 1's data to an array.

#### **Parameters**

r	the pointer to an array.
m	the row size of the array.
n	the column size of the array.

4.2.2.14 void set\_seqs\_arr ( double \*\* r, int m, int n )

assign sequential data to an array, i.e.

all rows contains data  $\{0,\,1,\,2,\,..\,\,,\,n$  -  $1\}.$ 

### **Parameters**

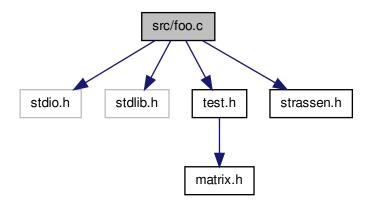
r	the pointer to an array.
m	the row size of the array.
n	the column size of the array.

4.2.2.15 void sub\_arr ( double \*\* a, double \*\* b, double \*\* c, int m, int n)

## 4.3 src/foo.c File Reference

 $\label{eq:continuity} \verb§#include < stdlib.h> \verb§#include "test.h" × \\$ 

#include "strassen.h" Include dependency graph for foo.c:



## **Functions**

• int main (int argc, const char \*argv[])

main the entry of this project.

## 4.3.1 Function Documentation

4.3.1.1 int main ( int argc, const char \* argv[] )

main the entry of this project.

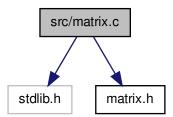
argc	the number of string parameters.
argv[]	the content of input string parameters. argv[0] the command name
	argv[1] the dimension of the test square matrices. argv[2] the opera-
	tions of the test. argv[3] the number of g_break used in strassen algo-
	rithm. argv[4] the patterns generated for test others reserved.

#### Returns

int 0 as program finished well.

## 4.4 src/matrix.c File Reference

#include <stdlib.h> #include "matrix.h" Include dependency graph for matrix.c:



## **Functions**

- matrix \* new\_matrix (double \*\*a) create a new matrix structure.
- void del\_matrix (matrix \*a)
   release the resource of a matrix

## 4.4.1 Function Documentation

4.4.1.1 void del\_matrix ( matrix \* a )

release the resource of a matrix

#### **Parameters**

a the pointer to the matrix.

4.4.1.2 matrix\* new\_matrix ( double \*\* a )

create a new matrix structure.

a the pointer to a matrix with type 'double' elements.

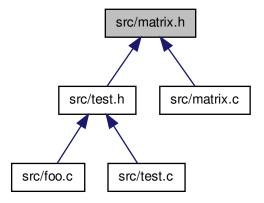
#### Returns

#### Note

the matrix structure can be expanded as necessary.

## 4.5 src/matrix.h File Reference

This graph shows which files directly or indirectly include this file:



#### **Data Structures**

• struct matrix

## **Functions**

- matrix \* new\_matrix (double \*\*a) create a new matrix structure.
- void del\_matrix (matrix \*a)
   release the resource of a matrix

#### 4.5.1 Function Documentation

#### 4.5.1.1 void del\_matrix ( matrix \* a )

release the resource of a matrix

#### **Parameters**

а	the	pointer	to th	ne matrix.	
---	-----	---------	-------	------------	--

## 4.5.1.2 matrix\* new\_matrix ( double \*\* a )

create a new matrix structure.

#### **Parameters**

a the pointer to a matrix with type 'double' elements.

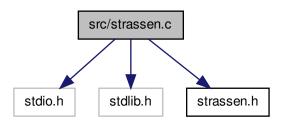
#### **Returns**

#### Note

the matrix structure can be expanded as necessary.

## 4.6 src/strassen.c File Reference

#include <stdio.h> #include <stdlib.h> #include "strassen.-h" Include dependency graph for strassen.c:



#### **Functions**

- double \*\*\* arr\_qsplit (double \*\*a, int n)
- split a given array into 4 equal sub-arrays.void arr qsplit recall (double \*\*\*a)

release the resources allocated for array split.

- void s\_mul (int n, double \*\*a, double \*\*b, double \*\*c, double \*\*d) perform the strassen multiply for matrices.
- void s\_add (int n, double \*\*a, double \*\*b, double \*\*c)
   perform the addition operation of matrices with matrices dividing.
- void s\_sub (int n, double \*\*a, double \*\*b, double \*\*c)
   perform the substraction operation of matrices with matrices dividing.

#### **Variables**

• int g break = 16

#### 4.6.1 Function Documentation

4.6.1.1 double \*\*\* arr\_qsplit ( double \*\* 
$$a$$
, int  $n$  )

split a given array into 4 equal sub-arrays.

#### **Parameters**

а	an input {2n x 2n} array
n	the dimension of new arrays, i.e. {n x n}.

#### Returns

a pointer to 4 arrays.

```
 | AAAABBBB \mid | AAAABBBB \mid | AAAABBBB \mid | AAAA \mid | BBBB \mid | CCCC \mid | DDDD \mid | AAAABBBB \mid -> | AAAA \mid , | BBBB \mid , | CCCC \mid , | DDDD \mid | CCCCDDDD \mid | AAAA \mid | BBBB \mid | CCCC \mid | DDDD \mid | CCCCDDDD \mid | AAAA \mid | BBBB \mid | CCCC \mid | DDDD \mid | CCCCDDDD \mid | AAAA \mid | BBBB \mid | CCCC \mid | DDDD \mid | CCCCDDDD | C
```

4.6.1.2 void arr\_qsplit\_recall ( double \*\*\* a )

release the resources allocated for array split.

	а	a pointer to 4 arrays.
--	---	------------------------

4.6.1.3 void  $s_add$  ( int n, double \*\* a, double \*\* b, double \*\* c )

perform the addition operation of matrices with matrices dividing.

#### **Parameters**

n	the dimension of the input and output matrices.
а	the first input matrix
b	the second input matrix
С	the result of matrix a adds matrix b.

if the dimension of input matrices are less than g\_break, perform a normal matrix substraction operation. else, divide the matrices and calculate the result of each sub matrices.

4.6.1.4 void s\_mul ( int n, double \*\* a, double \*\* b, double \*\* c, double \*\* d)

perform the strassen multiply for matrices.

#### **Parameters**

n	the dimension of the input and output matrices.
а	the first input matrix
b	the second input matrix
С	the result of matrix a multiplies matrix b.
d	the scratchpad for calculation.

## Note

the algorithm is refer to wiki, https://en.wikipedia.org/wiki/Strassen\_algorithm

if the dimension of input matrices are less than g\_break, perform a normal  $O(N^3)$  matrix multiply operation. else, divide the matrices and perform strassen algorithm.

4.6.1.5 void s\_sub ( int n, double \*\* a, double \*\* b, double \*\* c )

perform the substraction operation of matrices with matrices dividing.

#### **Parameters**

n	the dimension of the input and output matrices.
а	the first input matrix
b	the second input matrix
С	the result of matrix a substracts matrix b.

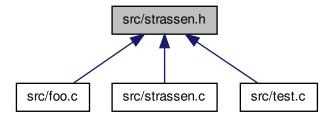
if the dimension of input matrices are less than g\_break, perform a normal matrix addition operation. else, divide the matrices and calculate the result of each sub matrices.

#### 4.6.2 Variable Documentation

4.6.2.1 int g\_break = 16

## 4.7 src/strassen.h File Reference

This graph shows which files directly or indirectly include this file:



#### **Defines**

- #define a11 p[0]
- #define a12 p[1]
- #define a21 p[2]
- #define a22 p[3]
- #define b11 q[0]
- #define b12 q[1]
- #define b21 q[2]
- #define b22 q[3]
- #define c11 r[0]
- #define c12 r[1]
- #define c21 r[2]
- #define c22 r[3]
- #define d11 s[0]
- #define d12 s[1]
- #define d21 s[2]
- #define d22 s[3]

## **Functions**

• double \*\*\* arr\_qsplit (double \*\*a, int n)

```
split a given array into 4 equal sub-arrays.
```

- void arr\_qsplit\_recall (double \*\*\*a)
  - release the resources allocated for array split.
- void s\_mul (int n, double \*\*a, double \*\*b, double \*\*c, double \*\*d)
   perform the strassen multiply for matrices.
- void s\_add (int n, double \*\*a, double \*\*b, double \*\*c)
   perform the addition operation of matrices with matrices dividing.
- void s\_sub (int n, double \*\*a, double \*\*b, double \*\*c)
   perform the substraction operation of matrices with matrices dividing.

#### **Variables**

• int g break

#### 4.7.1 Define Documentation

- 4.7.1.1 #define a11 p[0]
- 4.7.1.2 #define a12 p[1]
- 4.7.1.3 #define a21 p[2]
- 4.7.1.4 #define a22 p[3]
- 4.7.1.5 #define b11 q[0]
- 4.7.1.6 #define b12 q[1]
- 4.7.1.7 #define b21 q[2]
- 4.7.1.8 #define b22 q[3]
- 4.7.1.9 #define c11 r[0]
- 4.7.1.10 #define c12 r[1]
- 4.7.1.11 #define c21 r[2]
- 4.7.1.12 #define c22 r[3]
- 4.7.1.13 #define d11 s[0]
- 4.7.1.14 #define d12 s[1]
- 4.7.1.15 #define d21 s[2]

4.7.1.16 #define d22 s[3]

#### 4.7.2 Function Documentation

4.7.2.1 double \*\*\* arr\_qsplit ( double \*\* 
$$a$$
, int  $n$  )

split a given array into 4 equal sub-arrays.

#### **Parameters**

а	an input {2n x 2n} array
n	the dimension of new arrays, i.e. {n x n}.

#### Returns

a pointer to 4 arrays.

```
 | AAAABBBB \mid | AAAABBBB \mid | AAAABBBB \mid | AAAA \mid | BBBB \mid | CCCC \mid | DDDD \mid | AAAABBBB \mid -> | AAAA \mid , | BBBB \mid , | CCCC \mid , | DDDD \mid | CCCCDDDD \mid | AAAA \mid | BBBB \mid | CCCC \mid | DDDD \mid | CCCCDDDD \mid | AAAA \mid | BBBB \mid | CCCC \mid | DDDD \mid | CCCCDDDD | | CCCCDDDD | | CCCCDDDD | CCCCDDD
```

4.7.2.2 void arr\_qsplit\_recall ( double \*\*\* a )

release the resources allocated for array split.

## **Parameters**

а	a pointer to 4 arrays.

4.7.2.3 void s\_add ( int n, double \*\* a, double \*\* b, double \*\* c )

perform the addition operation of matrices with matrices dividing.

#### **Parameters**

n	the dimension of the input and output matrices.
а	the first input matrix
b	the second input matrix
С	the result of matrix a adds matrix b.

if the dimension of input matrices are less than g\_break, perform a normal matrix substraction operation. else, divide the matrices and calculate the result of each sub matrices.

```
4.7.2.4 void s_mul ( int n, double ** a, double ** b, double ** c, double ** d)
```

perform the strassen multiply for matrices.

#### **Parameters**

n	the dimension of the input and output matrices.
а	the first input matrix
b	the second input matrix
С	the result of matrix a multiplies matrix b.
d	the scratchpad for calculation.

#### Note

```
the algorithm is refer to wiki, https://en.wikipedia.org/wiki/-
Strassen_algorithm
```

if the dimension of input matrices are less than g\_break, perform a normal  $O(N^3)$  matrix multiply operation. else, divide the matrices and perform strassen algorithm.

```
4.7.2.5 void s_sub ( int n, double ** a, double ** b, double ** c )
```

perform the substraction operation of matrices with matrices dividing.

## **Parameters**

n	the dimension of the input and output matrices.
а	the first input matrix
b	the second input matrix
С	the result of matrix a substracts matrix b.

if the dimension of input matrices are less than g\_break, perform a normal matrix addition operation. else, divide the matrices and calculate the result of each sub matrices.

#### 4.7.3 Variable Documentation

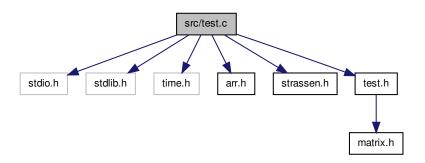
4.7.3.1 int g\_break

## 4.8 src/test.c File Reference

```
#include <stdio.h> #include <stdlib.h> #include <time.-
h> #include "arr.h" #include "strassen.h" #include "test.-
```

26 File Documentation

h" Include dependency graph for test.c:



#### **Functions**

```
test_t * new_test (int n)
```

create a new test object.

void del\_test (test\_t \*r)

delete a test object and release its resources.

void dump\_result (test\_t \*t)

dump test result

• void init\_test\_data (test\_t \*t, int pattern)

set up data data for input marices.

int test\_strassen\_multiply (test\_t \*t)

perform C = A + S(A, B), while strassen algorithm is applied in S().

int test\_normal\_multiply (test\_t \*t)

perform C = A + M(A, B), while a normal matrix multiply function is applied in M().

• int test\_check\_result (test\_t \*t)

check if the result of matrix c and d in test object are equal.

• int test valid data (test t \*t)

validate each element test object input matrix is valid.

• int round\_down\_power\_of\_two (int n)

round down a number to be power-of-two one.

• int test\_tweak\_breaks (int n)

validate the value of g\_break and tweak it.

• double test\_case (int n, int ops)

a test case for this matrix operation test.

• void test (int n, int ops)

entry of test.

#### **Variables**

```
• int g_pattern = PATTERN_ONES
```

#### 4.8.1 Function Documentation

```
4.8.1.1 void del_test ( test_t * r )
```

delete a test object and release its resources.

#### **Parameters**

```
r a test object.
```

```
4.8.1.2 void dump_result ( test_t * t )
```

dump test result

#### **Parameters**

```
t a test object
```

4.8.1.3 void init\_test\_data ( test\_t \* t, int pattern )

set up data data for input marices.

#### **Parameters**

t	a test object.
pattern	the pattern of given data.

PATTERN\_RANDOM apply random data to the test object input. PATTERN\_ONES apply all 1's to the test object input. PATTERN\_SEQS apply sequential data, {0, 1, 2, ..., n-1} for each row, to the object input.

```
4.8.1.4 test_t* new_test ( int n )
```

create a new test object.

n	the dimension of a matrix.

#### Returns

a pointer to new test object.

## 4.8.1.5 int round\_down\_power\_of\_two (int n)

round down a number to be power-of-two one.

#### **Parameters**

n	the number to be rounded-down

#### Returns

the rounded-down number

4.8.1.6 void test ( int n, int ops )

entry of test.

#### **Parameters**

n	the dimension of input matrices.
ops	the operation code of test

OP\_STRASSEN\_MULTIPLY perform C = A + S(A, B), where  $\{A, B, C\}$  are mapped to  $\{a, b, c\}$  in test object. OP\_NORMAL\_MULTIPLY perform D = A + M(A, B), where  $\{A, B, D\}$  are mapped to  $\{a, b, d\}$  in test object. OP\_VERIFY\_CORRECTNESS per form C = A + S(A, B) and D = A + M(A, B), where  $\{A, B, C, D\}$  are mapped to  $\{a, b, c, d\}$  in test object. And then, compare the matrix C and D to see if the results are the same.

4.8.1.7 double test\_case ( int n, int ops )

a test case for this matrix operation test.

## **Parameters**

n	the dimension of input metrices.
ops	the operations for this test.

#### Returns

the elapsed time in mini-second.

Note

the users can modify the time elapse functions in their specific platform to calculate the performance of the matrix operation test.

```
4.8.1.8 int test_check_result ( test_t * t )
```

check if the result of matrix c and d in test object are equal.

#### **Parameters**

```
t the test object.
```

#### Returns

0 as equal;, and -1 as inequal.

```
4.8.1.9 int test_normal_multiply ( test_t * t )
```

perform C = A + M(A, B), while a normal matrix multiply function is applied in M().

#### **Parameters**

```
t a test object.
```

## Returns

0 as passed.

```
4.8.1.10 int test_strassen_multiply ( test_t * t )
```

perform C = A + S(A, B), while strassen algorithm is applied in S().

#### **Parameters**

```
t a test object.
```

## Returns

0 as passed.

## 4.8.1.11 int test\_tweak\_breaks ( int n )

validate the value of g\_break and tweak it.

n the number to be validated and tweaked.

#### Returns

the number has been tweaked.

#### Note

MAX\_DIM is a predefined number for max dimension of the matrices.

4.8.1.12 int test\_valid\_data ( test\_t 
$$* t$$
 )

validate each element test object input matrix is valid.

#### **Parameters**

t the test object

#### Returns

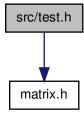
0 as all data are valid. -1 as input matrix a is invalid. -2 as input matrix b is invalid.

## 4.8.2 Variable Documentation

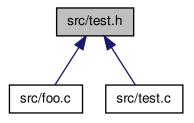
## 4.8.2.1 int g\_pattern = PATTERN\_ONES

## 4.9 src/test.h File Reference

#include "matrix.h" Include dependency graph for test.h:



This graph shows which files directly or indirectly include this file:



#### **Data Structures**

• struct test\_t

#### **Enumerations**

- enum PATTERN\_MODE { PATTERN\_RANDOM = 0, PATTERN\_ONES, PATTERN\_SEQS, NUMBER\_OF\_PATTERNS }
- enum OPS { OP\_STRASSEN\_MULTIPLY = 0, OP\_NORMAL\_MULTIPLY, OP-VERIFY\_CORRECTNESS, OP\_DUMP\_RESULT, NUMBER\_OF\_OPS }

## **Functions**

```
test_t * new_test (int n)
```

create a new test object.

void del\_test (test\_t \*r)

delete a test object and release its resources.

void dump\_result (test\_t \*t)

dump test result

void init\_test\_data (test\_t \*t, int pattern)

set up data data for input marices.

• int test\_strassen\_multiply (test\_t \*t)

perform C = A + S(A, B), while strassen algorithm is applied in S().

int test\_normal\_multiply (test\_t \*t)

perform C = A + M(A, B), while a normal matrix multiply function is applied in M().

int test\_check\_result (test\_t \*t)

check if the result of matrix c and d in test object are equal.

32 File Documentation

```
    int test_valid_data (test_t *t)
        validate each element test object input matrix is valid.
    int round_down_power_of_two (int n)
        round down a number to be power-of-two one.
    int test_tweak_breaks (int n)
        validate the value of g_break and tweak it.
    double test_case (int n, int ops)
```

a test case for this matrix operation test.

## Variables

• int g\_pattern

void test (int n, int ops)
 entry of test.

## 4.9.1 Enumeration Type Documentation

4.9.1.1 enum OPS

#### **Enumerator:**

OP\_STRASSEN\_MULTIPLY
OP\_NORMAL\_MULTIPLY
OP\_VERIFY\_CORRECTNESS
OP\_DUMP\_RESULT
NUMBER\_OF\_OPS

## 4.9.1.2 enum PATTERN\_MODE

#### **Enumerator:**

PATTERN\_RANDOM
PATTERN\_ONES
PATTERN\_SEQS
NUMBER\_OF\_PATTERNS

#### 4.9.2 Function Documentation

4.9.2.1 void del\_test ( test\_t \* r )

delete a test object and release its resources.

r a test object.

4.9.2.2 void dump\_result ( test\_t \* t )

dump test result

#### **Parameters**

t a test object

4.9.2.3 void init\_test\_data ( test\_t \* t, int pattern )

set up data data for input marices.

#### **Parameters**

t	a test object.
pattern	the pattern of given data.

PATTERN\_RANDOM apply random data to the test object input. PATTERN\_ONES apply all 1's to the test object input. PATTERN\_SEQS apply sequential data, {0, 1, 2, ..., n-1} for each row, to the object input.

4.9.2.4 test\_t\* new\_test ( int n )

create a new test object.

#### **Parameters**

n the dimension of a matrix.

## Returns

a pointer to new test object.

4.9.2.5 int round\_down\_power\_of\_two (int n)

round down a number to be power-of-two one.

n	the number to be rounded-down

#### Returns

the rounded-down number

4.9.2.6 void test ( int n, int ops )

entry of test.

#### **Parameters**

n	the dimension of input matrices.
ops	the operation code of test

OP\_STRASSEN\_MULTIPLY perform C = A + S(A, B), where  $\{A, B, C\}$  are mapped to  $\{a, b, c\}$  in test object. OP\_NORMAL\_MULTIPLY perform D = A + M(A, B), where  $\{A, B, D\}$  are mapped to  $\{a, b, d\}$  in test object. OP\_VERIFY\_CORRECTNESS per form C = A + S(A, B) and D = A + M(A, B), where  $\{A, B, C, D\}$  are mapped to  $\{a, b, c, d\}$  in test object. And then, compare the matrix C and D to see if the results are the same.

4.9.2.7 double test\_case ( int n, int ops )

a test case for this matrix operation test.

### **Parameters**

n	the dimension of input metrices.
ops	the operations for this test.

#### Returns

the elapsed time in mini-second.

#### Note

the users can modify the time elapse functions in their specific platform to calculate the performance of the matrix operation test.

4.9.2.8 int test\_check\_result ( test\_t \*t )

check if the result of matrix c and d in test object are equal.

	t	the test object.
--	---	------------------

#### **Returns**

0 as equal;, and -1 as inequal.

```
4.9.2.9 int test_normal_multiply ( test_t * t )
```

perform C = A + M(A, B), while a normal matrix multiply function is applied in M().

#### **Parameters**

t a test object.

#### Returns

0 as passed.

```
4.9.2.10 int test_strassen_multiply ( test_t * t )
```

perform C = A + S(A, B), while strassen algorithm is applied in S().

#### **Parameters**

t a test object.

#### **Returns**

0 as passed.

4.9.2.11 int test\_tweak\_breaks ( int n )

validate the value of g\_break and tweak it.

### **Parameters**

n the number to be validated and tweaked.

#### **Returns**

the number has been tweaked.

#### Note

MAX\_DIM is a predefined number for max dimension of the matrices.

4.9.2.12 int test\_valid\_data ( test\_t \*t )

validate each element test object input matrix is valid.

#### **Parameters**

t the test object

#### Returns

0 as all data are valid. -1 as input matrix a is invalid. -2 as input matrix b is invalid.

## 4.9.3 Variable Documentation

## 4.9.3.1 int g\_pattern