Reference Manual

Generated by Doxygen 1.7.1

Sat Sep 3 2011 19:40:02

Contents

1 Class Index **1**

1.1 Class List 1

2 File Index **3**

2.1 File List 3

3 Class Documentation **5**

3.1 Bench Class Reference 5

3.1.1 Member Function Documentation 5

3.1.1.1 Cost_If_Swap 5

3.1.1.2 Cost_On_Variable 6

3.1.1.3 Display_Solution 6

3.1.1.4 Executed_Swap 6

3.1.1.5 Next_I 6

3.1.1.6 Next_J 6

3.1.1.7 Reset 7

3.2 MagicSquare Class Reference 7

3.2.1 Member Function Documentation 8

3.2.1.1 Check_Solution 8

3.2.1.2 Cost_If_Swap 8

3.2.1.3 Cost_Of_Solution 9

3.2.1.4 Cost_On_Variable 9

3.2.1.5 Executed_Swap 9

3.2.1.6 Init_Parameters 9

3.2.1.7 Solve 9

3.3 MagicSquare::XRef Struct Reference 10

4 File Documentation **11**

4.1 bench.cpp File Reference 11

4.1.1 Detailed Description 11
4.2 bench.h File Reference 11
4.2.1 Detailed Description 11
4.3 magic_square.cpp File Reference 12
4.3.1 Detailed Description 12
4.4 magic_square.h File Reference 12
4.4.1 Detailed Description 13

Chapter 1

Class Index

1.1 Class List

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

Bench 5
MagicSquare 7
MagicSquare::XRef 10

Chapter 2

File Index

2.1 File List

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

bench.cpp (Benchmark mother class) 11
bench.h (Benchmark mother class) 11
magic_square.cpp (Magic Square benchmark) 12
magic_square.h (Magic Square benchmark) 12

Chapter 3

Class Documentation

3.1 Bench Class Reference

Public Member Functions

- int Cost_If_Swap (int current_cost, int i, int j)
  Wrapper when user function Cost_If_Swap is not defined.
- int Cost_On_Variable (int k)
  Wrapper when user function Cost_On_Variable is not defined.
- void Display_Solution (AdData *p_ad)
  Wrapper when user function Display_Solution is not defined.
- void Executed_Swap (int k1, int k2)
  Wrapper when user function Executed_Swap is not defined.
- int Next_I (int i)
  Wrapper when user function Next_I is not defined.
- int Next_J (int i, int j)
  Wrapper when user function Next_J is not defined.
- int Reset (int n, AdData *p_ad)
  Wrapper when user function Reset is not defined.

3.1.1 Member Function Documentation

3.1.1.1 int Bench::Cost_If_Swap (int *current_cost,* int *i,* int *j*)

Wrapper when user function Cost_If_Swap is not defined.

Parameters

current_cost,: the current cost when this function is called. i and j, the variables with which we simulate a swap to compute the resulting cost.

Returns

The cost if we swap variables i and j.

3.1.1.2 int Bench::Cost_On_Variable (int *k*)

Wrapper when user function Cost_On_Variable is not defined.

Parameters

k,: the variable on which we project the cost.

Returns

The cost projected on variable k.

3.1.1.3 void Bench::Display_Solution (AdData * *p_ad*)

Wrapper when user function Display_Solution is not defined.

Parameters

p_ad,: Pointer toward the current configuration (or solution).

3.1.1.4 void Bench::Executed_Swap (int *k1*, int *k2*)

Wrapper when user function Executed_Swap is not defined.

Parameters

k1 and k2: variables to swap.

3.1.1.5 int Bench::Next_I (int *i*)

Wrapper when user function Next_I is not defined.

Parameters

i,: a variable.

Returns

The next variable (i+1)

3.1.1.6 int Bench::Next_J (int *i*, int *j*)

Wrapper when user function Next_J is not defined.

Parameters

i and j: two variables.

Returns

The next j-variable (j+1), unless j < 0 (then returns i+1)

3.1.1.7 int Bench::Reset (int *n*, AdData * *p_ad*)

Wrapper when user function Reset is not defined.

Parameters

n,: number of reset loop to perform. p_ad: pointer toward the configuration.

Returns

The new cost, or -1 if unknown.

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

- bench.h
- bench.cpp

3.2 MagicSquare Class Reference

Classes

- struct XRef

Public Member Functions

- void Solve (AdData *p_ad)

  Initializations needed for the resolution.

- int Cost_Of_Solution (int should_be_recorded)

  Computes the total cost of the current solution. Also computes errors on constraints for subsequent calls to Cost_On_Variable, Cost_If_Swap and Executed_Swap.

- int Cost_On_Variable (int k)

  Evaluates the error on a variable.

- int Cost_If_Swap (int current_cost, int k1, int k2)

  Computes the cost if we swap k1 and k2. No swaps are recorded.

- void Executed_Swap (int k1, int k2)

  Records a swap between k1 and k2.

- void Init_Parameters (AdData *p_ad)

  Initializes parameters like freeze_swap, reset_percent, ...

- int Check_Solution (AdData *p_ad)

  Checks if the configuration is a solution.

Static Public Attributes

- static int **size**
- static int * **sol**
- static int **square_length**
- static int **square_length_m1**
- static int **square_length_p1**
- static int **avg**
- static int * **err_l**
- static int * **err_l_abs**
- static int * **err_c**
- static int * **err_c_abs**
- static int **err_d1**
- static int **err_d1_abs**
- static int **err_d2**
- static int **err_d2_abs**
- static XRef * **xref**

3.2.1 Member Function Documentation

3.2.1.1 int MagicSquare::Check_Solution (AdData * *p_ad*)

Checks if the configuration is a solution.

Parameters

p_ad,: Pointer toward the current configuration.

Returns

1 if the configuration is a solution, 0 otherwise.

3.2.1.2 int MagicSquare::Cost_If_Swap (int *current_cost,* int *i,* int *j*)

Computes the cost if we swap k1 and k2. No swaps are recorded.

Parameters

current_cost,: the current cost when this function is called.

i and j, the variables with which we simulate a swap to compute the resulting cost.

Returns

The cost if we swap variables i and j.

3.2.1.3 int MagicSquare::Cost_Of_Solution (int *should_be_recorded*)

Computes the total cost of the current solution. Also computes errors on constraints for subsequent calls to Cost_On_Variable, Cost_If_Swap and Executed_Swap.

Parameters

should_be_recorded,: dummy input.

Returns

The cost of the current configuration.

3.2.1.4 int MagicSquare::Cost_On_Variable (int *k*)

Evaluates the error on a variable.

Parameters

k,: the variable on which we project the cost.

Returns

The cost projected on variable k.

3.2.1.5 void MagicSquare::Executed_Swap (int *k1*, int *k2*)

Records a swap between k1 and k2.

Parameters

k1 and k2: variables to swap.

3.2.1.6 void MagicSquare::Init_Parameters (AdData * *p_ad*)

Initializes parameters like freeze_swap, reset_percent, ...

Parameters

p_ad,: Pointer toward the current configuration.

3.2.1.7 void MagicSquare::Solve (AdData * *p_ad*)

Initializations needed for the resolution.

Parameters

p_ad,: Pointer toward the current configuration.

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

- magic_square.h
- magic_square.cpp

3.3 MagicSquare::XRef Struct Reference

Public Attributes

- unsigned int **d1**:1
- unsigned int **d2**:1
- unsigned int **l**:15
- unsigned int **c**:15

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

- magic_square.h

Chapter 4

File Documentation

4.1 bench.cpp File Reference

benchmark mother class

```
#include "bench.h"
```

4.1.1 Detailed Description

benchmark mother class Adaptive search C++

Author

Florian Richoux

Date

2011-09-03

4.2 bench.h File Reference

benchmark mother class

Classes

- class Bench

4.2.1 Detailed Description

benchmark mother class Adaptive search C++

Author

Florian Richoux

Date

2011-09-03

4.3 magic_square.cpp File Reference

Magic Square benchmark.

Defines

- #define **AdjustL**(r, diff, k) r = r - err_l_abs[k] + abs(err_l[k] + diff)
- #define **AdjustC**(r, diff, k) r = r - err_c_abs[k] + abs(err_c[k] + diff)
- #define **AdjustD1**(r, diff) r = r - err_d1_abs + abs(err_d1 + diff)
- #define **AdjustD2**(r, diff) r = r - err_d2_abs + abs(err_d2 + diff)

Variables

- int **param_needed** = 1

4.3.1 Detailed Description

Magic Square benchmark. Adaptive search C++

Author

Florian Richoux

Date

2011-09-03

4.4 magic_square.h File Reference

Magic Square benchmark.

Classes

- class MagicSquare
- struct MagicSquare::XRef

Defines

- #define **XSet**(xr, line, col, diag1, diag2) xr.l = line; xr.c = col; xr.d1 = diag1; xr.d2 = diag2
- #define **XGetL**(xr) xr.l
- #define **XGetC**(xr) xr.c
- #define **XIsOnD1**(xr) (xr.d1 != 0)
- #define **XIsOnD2**(xr) (xr.d2 != 0)

4.4.1 Detailed Description

Magic Square benchmark. Adaptive search C++

Author

Florian Richoux

Date

2011-09-03

sol[] = vector of values (by line), sol[0..square_length-1] contain the first line, sol[square_length-2∗square_length-1] contain the 2nd line, ... values are in 1..square_length∗square_length

The constraints are: for each line, column, diagonal 1 (\) and 2 (/) the sum must be equal to avg = square_-length ∗ (square_length∗square_length + 1) / 2;

err_l[i] = -avg + sum of line i err_c[j] = -avg + sum of column j err_d1 = -avg + sum of diagonal 1 err_d2 = -avg + sum of diagonal 2

square_length-1 square_length-1 The total cost = Sum |err_l[i]| + Sum |err_c[i]| + |err_d1| + |err_d2| i=0 j=0

The projection on a variable at i, j: // err_var[i][j] = | err_l[i] + err_c[j] + F1(i,j) + F2(i,j) | SLOW version err_var[i][j] = | err_l[i] + err_c[j] + F1(i,j) + F2(i,j) | with F1(i,j) = err_d1 if i,j is on diagonal 1 (i.e. i=j) else = 0 and F2(i,j) = err_d2 if i,j is on diagonal 2 (i.e. j=square_length-1-i) else = 0

Index

Bench, 5
    Cost_If_Swap, 5
    Cost_On_Variable, 6
    Display_Solution, 6
    Executed_Swap, 6
    Next_I, 6
    Next_J, 6
    Reset, 7
bench.cpp, 11
bench.h, 11

Check_Solution
    MagicSquare, 8
Cost_If_Swap
    Bench, 5
    MagicSquare, 8
Cost_Of_Solution
    MagicSquare, 8
Cost_On_Variable
    Bench, 6
    MagicSquare, 9

Display_Solution
    Bench, 6

Executed_Swap
    Bench, 6
    MagicSquare, 9

Init_Parameters
    MagicSquare, 9

magic_square.cpp, 12
magic_square.h, 12
MagicSquare, 7
    Check_Solution, 8
    Cost_If_Swap, 8
    Cost_Of_Solution, 8
    Cost_On_Variable, 9
    Executed_Swap, 9
    Init_Parameters, 9
    Solve, 9
MagicSquare::XRef, 10

Next_I
    Bench, 6

Next_J
    Bench, 6

Reset
    Bench, 7

Solve
    MagicSquare, 9