Knight problem to find longest path in 32x32 grid

Generated by Doxygen 1.7.6.1

Mon Sep 8 2014 22:59:28

Contents

Chapter 1

Knight Path Problem: This is problem to find certain

paths that a knight can follow.

Author

Suresh Golconda

Description: The system finds the longest path from given

start and end node on a 32x32 chess board.

Chapter 2

Namespace Index

2.1 Namespace List

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

Chapter 3

Class Index

3.1 Class List

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

Chapter 4

Namespace Documentation

4.1 knight_path_t Namespace Reference

Classes

- class k_32_search_t
- class k_getpath_t
- class k_path_adapter_t
- class k_searcher_t
- class k_sub_board_t
- struct sub_board_grid_t
- class solution_t
- class k_tour_t

  This is tour_t def.
- struct parameters_t
- struct tour_path_t
- struct node_t

  A node on board.
- struct node_pair_t

  pair of node, used mostly for representing start-end
- struct stitch_t

Typedefs

- typedef std::vector< parameters_t > parameter_list_t
- typedef std::vector< tour_path_t > tour_path_list_t

  To declare a vector of tour_path_t's.
- typedef std::vector< stitch_t > stitch_list_t

Enumerations

- enum color_t { **BLACK**, **WHITE** }

Functions

- bool same_color (int x1, int y1, int x2, int y2)
- color_t get_cell_color (int x, int y)
- std::string get_color_name (color_t color)
- bool scan_dim_src_dest (std::string indata_, int &sx, int &sy, int &ex, int &ey)
- void print_path (std::vector< node_t > const &path)

4.1.1 Detailed Description

.H file to define k_getpath_t class. Loads precomputed paths from database and provides an interface for query.

Author

Suresh Golconda

.H file to define k_path_adapter_t class. Adapter function for getting path. Can either load from pre-computed results or perform fresh search using k_tour_t.

Author

Suresh Golconda

.H file to define k_searcher_t class, which implements functions required for generating database of paths for smaller grids such as 6x6, 6x8, 8x6, 8x8.

Author

Suresh Golconda

.H file to define k_sub_board_t class, which implements operations at sub_board level

Author

Suresh Golconda

.H file with code for finding knight's touring path. Here by "TOUR"ing means, hamilton path covering all the cells exactly once. With an extension of hamilton path with one-less path.

This class searches grid for such paths. EXTENSIVELY tested on grids of size: 6x6, 6x8, 8x6, and 8x8.

Defines two class: solution_t and k_tour_t

Author

Suresh Golconda

.H file to define utility functions.

Author

Suresh Golconda

4.1.2 Typedef Documentation

4.1.2.1 typedef std::vector<parameters_t> knight_path_t::parameter_list_t

Typedef for declaring vector of parameters

4.1.2.2 typedef std::vector<stitch_t> knight_path_t::stitch_list_t

A vector of stitchs'

4.1.3 Enumeration Type Documentation

4.1.3.1 enum knight_path_t::color_t

Enum to reprsent color of a cell

4.1.4 Function Documentation

4.1.4.1 color_t knight_path_t::get_cell_color (int *x,* int *y*)

Returns the color of the given (x, y) node. It is assumed that node (0,0) white (typically in cheess). It holds for all sub-boards in our case, as all sub-boards have have even size dimensions

4.1.4.2 string knight_path_t::get_color_name (color_t *color*)

Returns the name of the given color as a string. Used for printing

4.1.4.3 void knight_path_t::print_path (std::vector< node_t > const & *path*)

Prints the provided path onto screen

4.1.4.4 bool knight_path_t::same_color (int *x1,* int *y1,* int *x2,* int *y2*)

Returns true if given pair of nodes both have similar color

4.1.4.5 bool knight_path_t::scan_dim_src_dest (std::string *indata_*, int & *sx*, int & *sy*, int & *ex*, int & *ey*)

Utility function to scan, src, and destination node's coordinates. Expected format: comma seperated four numbers.

Returns

True if scan was successful, else false

Chapter 5

Class Documentation

5.1 knight_path_t::k_32_search_t Class Reference

Public Member Functions

- k_32_search_t (bool use_db)
- void search_longest_path_32x (int gsx, int gsy, int gex, int gey, vector< node_t > &complete_path)

5.1.1 Constructor & Destructor Documentation

5.1.1.1 knight_path_t::k_32_search_t::k_32_search_t (bool *use_db*) [inline]

Constructor with a flag that determines if to use pre-computed paths or search from scratch

5.1.2 Member Function Documentation

5.1.2.1 void k_32_search_t::search_longest_path_32x (int *gsx*, int *gsy*, int *gex*, int *gey*, vector< node_t > & *complete_path*)

Searches for a path from global (gsx, gsy) to global (gex, gey). Returns path as (complete_path)

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

- k_32_search.H
- k_32_search.C

5.2 knight_path_t::k_getpath_t Class Reference

Public Member Functions

- k_getpath_t ()
- void load_all_para ()
- bool get_path (int maxx, int maxy, int sx, int sy, int ex, int ey, std::vector< node_t > &node_list)
- bool get_path (int maxx, int maxy, int sx, int sy, int ex, int ey, tour_path_t &tp)
- bool **test_all_paths** ()
- bool **test_paths_for_board** (int maxx, int maxy, bool allow_same_color=false)
- void **print_tour_path** (tour_path_list_t const &pl)

5.2.1 Constructor & Destructor Documentation

5.2.1.1 knight_path_t::k_getpath_t::k_getpath_t ( ) [inline]

Set paths to pre-computed paths

5.2.2 Member Function Documentation

5.2.2.1 bool knight_path_t::k_getpath_t::get_path (int *maxx,* int *maxy,* int *sx,* int *sy,* int *ex,* int *ey,* std::vector< node_t > & *node_list*)

Searches and returns a path on board of size (maxx, maxy) from given start-end location. Provides the path as a vector of nodes.

Returns

false if failed, else returns true

5.2.2.2 bool k_getpath_t::get_path (int *maxx,* int *maxy,* int *sx,* int *sy,* int *ex,* int *ey,* tour_path_t & *tp*)

Searches and returns a path on board of size (maxx, maxy) from given start-end location. Provides the path as a tour_path_t.

Returns

false if failed, else returns true

Checks maxx, maxy to determine which database to look for path. Checks maxx, maxy to determine which database to look for path.

5.2.2.3 void knight_path_t::k_getpath_t::load_all_para ( ) [inline]

Load the pre-computed parameter files

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

- k_getpath.H
- k_getpath.C

5.3 knight_path_t::k_path_adapter_t Class Reference

Public Member Functions

- k_path_adapter_t (bool use_db)
- bool get_path (int maxx, int maxy, int sx, int sy, int ex, int ey, std::vector< node_t > &path)

5.3.1 Constructor & Destructor Documentation

5.3.1.1 knight_path_t::k_path_adapter_t::k_path_adapter_t (bool *use_db*) [inline]

Takes in one bool argument which specifies if to use pre-computed database (if true).

5.3.2 Member Function Documentation

5.3.2.1 bool k_path_adapter_t::get_path (int *maxx,* int *maxy,* int *sx,* int *sy,* int *ex,* int *ey,* std::vector< node_t > & *path*)

Finds the path for given board of size (maxx, maxy), from given start point to given end point.

Returns

true if successful. Else return false

Iterate the search with different limit values. Initial lower values provides a chance to search path from end to start, which showed quicker results sometimes.

Get search limit

If failed, try searching from end to start

if reverse search found path, reverse the resultant path.

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

- k_path_adapter.H
- k_path_adapter.C

5.4 knight_path_t::k_searcher_t Class Reference

Public Member Functions

- void search_custom_set (int maxx, int maxy, std::string filename, int search_limit)
- int solutions_from (int sx, int sy, k_tour_t &kt, int maxx, int maxy)

5.4.1 Member Function Documentation

5.4.1.1 void k_searcher_t::search_custom_set (int *maxx,* int *maxy,* std::string *filename,* int *search_limit*)

Reads iput file to read start-end node pairs, then compute path for those pairs

5.4.1.2 int k_searcher_t::solutions_from (int *sx,* int *sy,* k_tour_t & *kt,* int *maxx,* int *maxy*)

Compute longest path for all possible destination nodes, starting from given (sx, sy).

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

- k_searcher.H
- k_searcher.C

5.5 knight_path_t::k_sub_board_t Class Reference

`#include <k_sub_board.H>`

Public Member Functions

- **k_sub_board_t** (int maxx_, int maxy_)
- void set_max_xy (int maxx_, int maxy_)
- int get_maxx () const
- int get_maxy () const
- bool is_entry_point (int x, int y)
- bool is_exit_point (int x, int y)
- int to_global_x (int x_) const
- int to_global_y (int y_) const
- int to_local_x (int x_) const
- int to_local_y (int y_) const
- k_sub_board_t (k_sub_board_t const &)
- void **operator=** (k_sub_board_t const &)
- void free_mem ()

Public Attributes

- int **base_gx**
- int **base_gy**
- int **lsx**
- int **lsy**
- int **lex**
- int **ley**
- int ** **m_cells**

5.5.1 Detailed Description

For performing sub-board related operations

5.5.2 Constructor & Destructor Documentation

5.5.2.1 knight_path_t::k_sub_board_t::k_sub_board_t (k_sub_board_t const &)

deleting following functions provided by c++, to prevent shallow copy of this objects: As there is dynamic memory allocated

5.5.3 Member Function Documentation

5.5.3.1 void knight_path_t::k_sub_board_t::free_mem ( ) [inline]

deletes the two dimensiona array created for storing m_cells

5.5.3.2 int knight_path_t::k_sub_board_t::get_maxx ( ) const [inline]

Returns the y-dim size

5.5.3.3 int knight_path_t::k_sub_board_t::get_maxy ( ) const [inline]

Returns the x-dim size

5.5.3.4 bool knight_path_t::k_sub_board_t::is_entry_point (int *x*, int *y*) [inline]

returns true if given point is an entry point, else false

5.5.3.5 bool knight_path_t::k_sub_board_t::is_exit_point (int *x*, int *y*) [inline]

returns true if given point is an exit point, else false

5.5.3.6 void k_sub_board_t::set_max_xy (int *maxx_*, int *maxy_*)

To re-set the dimensions of the grid

5.5.3.7 int knight_path_t::k_sub_board_t::to_global_x (int *x_*) const [inline]

converts from local sub-grid coordinate to global coordinates

5.5.3.8 int knight_path_t::k_sub_board_t::to_global_y (int *y_*) const [inline]

converts from local sub-grid coordinate to global coordinates

5.5.3.9 int knight_path_t::k_sub_board_t::to_local_x (int *x_*) const [inline]

converts from global coordinate to local sub-grid coordinates

5.5.3.10 int knight_path_t::k_sub_board_t::to_local_y (int *y_*) const [inline]

converts from global coordinate to local sub-grid coordinates

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

- k_sub_board.H
- k_sub_board.C

5.6 knight_path_t::k_tour_t Class Reference

This is tour_t def.

`#include <k_tour.H>`

Classes

- struct **child_t**

Public Member Functions

- **k_tour_t** (int xim, int ydim, int search_limit, bool for_knight=true)
- void reset ()
- void set_search_limit (int limit)
- bool **solveKT** (int sx, int sy, int ex, int ey, solution_t &sol)
- bool **solveKT_new** (int sx, int sy, int ex, int ey, solution_t &sol, std::vector< node_t > const &start_seq)
- bool **solveKT_bi** (int sx, int sy, int ex, int ey, solution_t &sol, std::vector< node_t > const &start_seq)

5.6.1 Detailed Description

This is tour_t def.

k_tour_t class definition

5.6.2 Member Function Documentation

5.6.2.1 void k_tour_t::reset ( )

Resets the member variables for new search

5.6.2.2 void knight_path_t::k_tour_t::set_search_limit (int *limit*) [inline]

Reset the search depth limit

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

- k_tour.H
- k_tour.C

5.7 knight_path_t::node_pair_t Struct Reference

pair of node, used mostly for representing start-end

`#include <k_util.H>`

Public Attributes

- node_t **n1**
- node_t **n2**

5.7.1 Detailed Description

pair of node, used mostly for representing start-end

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

- k_util.H

5.8 knight_path_t::node_t Struct Reference

A node on board.

`#include <k_util.H>`

Public Attributes

- int **posx**
- int **posy**

5.8.1 Detailed Description

A node on board.

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

- k_util.H

5.9 knight_path_t::parameters_t Struct Reference

`#include <k_util.H>`

Public Member Functions

- **parameters_t** (int sx, int sy, int ex, int ey)
- parameters_t ()

Public Attributes

- int **m_sx**
- int **m_sy**
- int **m_ex**
- int **m_ey**

5.9.1 Detailed Description

Parameter to store board config.

5.9.2 Constructor & Destructor Documentation

5.9.2.1 knight_path_t::parameters_t::parameters_t ( ) [inline]

Dummy constructor for allowing declaration of vector of objects

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

- k_util.H

5.10 knight_path_t::solution_t Class Reference

Public Member Functions

- solution_t (int dim_size_x, int dim_size_y)
- void display_board ()
- bool get_start_index (int &xi, int &yi) const
- bool get_end_index (int &xi, int &yi) const
- bool search_for (int val, int &xi, int &yi) const
- void reset (int value)
- **solution_t** (solution_t const &)
- void **operator=** (solution_t const &)

Public Attributes

- const int DIM_SIZE_X

  dimensions along either axis
- const int **DIM_SIZE_Y**
- int ** m_values

  Varaible to hold (cost) values of each cell.

5.10.1 Constructor & Destructor Documentation

5.10.1.1 knight_path_t::solution_t::solution_t (int *dim_size_x,* int *dim_size_y*) [inline]

Allocates memory for 2D array, which is delete in destructor

5.10.2 Member Function Documentation

5.10.2.1 void solution_t::display_board ( )

A utility function to print cost values of the grid's cells

5.10.2.2 bool solution_t::get_end_index (int & *xi,* int & *yi*) const

Returns x, y index of cell where solution ends

5.10.2.3 bool solution_t::get_start_index (int & *xi,* int & *yi*) const

Returns x, y index of cell where solution starts

5.10.2.4 void solution_t::reset (int *value*)

Reset all the nodes/cells to given value

5.10.2.5 bool solution_t::search_for (int *val,* int & *xi,* int & *yi*) const

Searches for given coordinates (x,y) of node with given (val). Function returns false if could not find the node. Else return true.

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

- k_tour.H
- k_tour.C

5.11 knight_path_t::stitch_t Struct Reference

`#include <k_util.H>`

Public Member Functions

- stitch_t (int dx1, int dy1, int dx2, int dy2)
  Stitch between subboards.
- void set (int dx1, int dy1, int dx2, int dy2)
  Set values of the stitch.

Public Attributes

- int m_delx1
- int **m_dely1**
- int **m_delx2**
- int **m_dely2**

5.11.1 Detailed Description

To define possible stitches interms of relative coordinate system. Stitch is applied between sub-boards.

5.11.2 Member Data Documentation

5.11.2.1 int knight_path_t::stitch_t::m_delx1

+ve add to lower end (0), -ve, sub at higher end (starts with -1, with -1 being last index

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

- k_util.H

5.12 knight_path_t::sub_board_grid_t Struct Reference

`#include <k_sub_board.H>`

Public Attributes

- k_sub_board_t **m_boards** [5][5]

5.12.1 Detailed Description

Grid of sub-boards

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

- k_sub_board.H

5.13 knight_path_t::tour_path_t Struct Reference

`#include <k_util.H>`

Public Attributes

- parameters_t **m_para**
- std::vector< int > **m_tour**

5.13.1 Detailed Description

To maintain a tour parameters (start, end) and vector of position of cell along the path.

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

- k_util.H