Algorithms project- Task 6

Generated by Doxygen 1.9.8

Chapter 1

Class Index

1.1 Class List

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

Chapter 2

File Index

2.1 File List

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

Chapter 3

Class Documentation

3.1 ChessBoard Class Reference

Represents a chessboard.

```
#include <ChessBoard.h>
```

Public Member Functions

- ChessBoard ()

    Constructs a chessboard with squares and knights.
- void **printBoard** ()

    Prints the current state of the chessboard.
- Square ∗ getSquare (int row, int col)

    Gets the square at the specified row and column.
- Square ∗ getSquare (int posNo)

    Gets the square at the specified position number.

3.1.1 Detailed Description

Represents a chessboard.

This class provides functionalities for creating and manipulating a chessboard, as well as printing its current state.

3.1.2 Constructor & Destructor Documentation

3.1.2.1 ChessBoard()

```
ChessBoard::ChessBoard ( )
```

Constructs a chessboard with squares and knights.

Initializes a 4x3 grid representing the chessboard and populates it with squares. Initializes the knights vector with black and white knight values. Places white knights (W1, W2, W3) on the first row and black knights (B1, B2, B3) on the last row of the chessboard.

The layout of the chessboard is as follows: number(knight name), Em == empty 0 (W1) 1 (W2) 2 (W3) 3 (Em) 4 (Em) 5 (Em) 6 (Em) 7 (Em) 8 (Em) 9 (B1) 10 (B2) 11 (B3)

Note: each square represented by: number (piece name), E.g., W1 is placed on position 0 at beginning. Note: 'Em' represents an empty square.

3.1.3 Member Function Documentation

3.1.3.1 getSquare() [1/2]

```
Square * ChessBoard::getSquare (
              int posNo )
```

Gets the square at the specified position number.

Parameters

posNo	The position number of the square.

Returns

Pointer to the square at the specified position.

3.1.3.2 getSquare() [2/2]

```
Square * ChessBoard::getSquare (
              int row,
              int col )
```

Gets the square at the specified row and column.

Parameters

row	The row index of the square.
col	The column index of the square.

Returns

Pointer to the square at the specified position.

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

- ChessBoard.h
- ChessBoard.cpp

3.2 Cost Class Reference

Represents the cost of moving a knight from one square to another.

```
#include <Graph.h>
```

Friends

- class **Graph**

3.2.1 Detailed Description

Represents the cost of moving a knight from one square to another.

The cost used to control the movement of knights to the correct squares.

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

- Graph.h

3.3 Graph Class Reference

Represents a graph for solving the knight's puzzle on a chessboard.

```
#include <Graph.h>
```

Public Member Functions

- Graph (ChessBoard &board)

  Constructs a Graph object with the given ChessBoard.
- void **printGraph** ()

  Prints the graph (for debugging purposes).
- vector< Move * > * solvePuzzle ()

  Solves the knight's puzzle and returns a vector of Move objects representing the solution.

3.3.1 Detailed Description

Represents a graph for solving the knight's puzzle on a chessboard.

The Graph class encapsulates the logic for solving the knight's tour puzzle using graph traversal algorithms.

3.3.2 Constructor & Destructor Documentation

3.3.2.1 Graph()

```
Graph::Graph (
            ChessBoard & board )
```

Constructs a Graph object with the given ChessBoard.

The constructor initializes a Graph object with the provided ChessBoard by creating nodes and establishing connections between them based on the specified rules. Each square is connected to other squares based on specific knight moves allowed in chess.

TODO: add the graph with cost image.

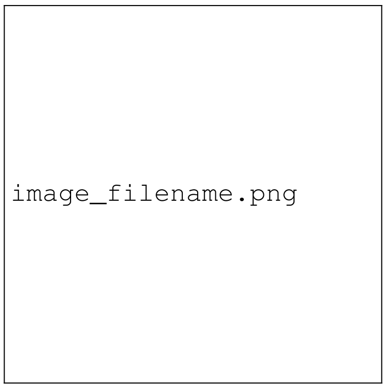

Figure 3.1 Caption text

Parameters

board	The ChessBoard object representing the chessboard.

3.3.3 Member Function Documentation

3.3.3.1 solvePuzzle()

```
vector< Move * > * Graph::solvePuzzle ( )
```

Solves the knight's puzzle and returns a vector of Move objects representing the solution.

This function implements a breadth-first search algorithm to solve the knight's puzzle.

- It starts from the given starting positions (9 and 11) and explores all possible moves using a modified queue-based approach.
- The algorithm iteratively explores all reachable squares considering the cost of each move and ensuring that the knight does not visit any square twice.
- If a square is empty, its neighboring squares are added to the queue for exploration.
- If the knight occupies a square, the algorithm attempts to move it to an empty neighboring square based on the cost associated with each move.
- After each move, the algorithm checks if the knight has reached its final destination, indicated by a square with only one edge.
- The algorithm continues until it completes all necessary swaps to solve the puzzle, i.e., until the knight has visited all squares exactly once.
- It maintains a record of all moves made during the puzzle solution, which are returned as a vector of Move objects.

The returned vector of Move objects represents the solution and is used by the GUI to display the sequence of knight movements on the chessboard.

Returns

vector<Move∗>∗ A pointer to a vector of Move objects representing the solution.

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

- Graph.h
- Graph.cpp

3.4 Move Class Reference

Represents a move made on a chessboard.

```
#include <Move.h>
```

Public Member Functions

- Move (int oldPos, int newPos, string knight)

  Constructor for the Move class.
- string getknight ()

  Retrieves the name of the knight.
- int getOldPos ()

  Retrieves the old position of the piece.
- int getNewPos ()

  Retrieves the new position of the piece.
- void **printMove** ()

  Prints the details of the move on console.

3.4.1 Detailed Description

Represents a move made on a chessboard.

The Move class provides functionality to record and represent individual moves made by the knight, including the old and new positions of the knight.

Moves will be used by the GUI to display moves at each step of solving the puzzle.

3.4.2 Constructor & Destructor Documentation

3.4.2.1 Move()

```
Move::Move (
             int oldPos,
             int newPos,
             string knight )  [inline]
```

Constructor for the Move class.

Parameters

oldPos	The old position of the piece.
newPos	The new position of the piece.
knight	The name of the knight making the move.

3.4.3 Member Function Documentation

3.4.3.1 getknight()

```
string Move::getknight ( )  [inline]
```

Retrieves the name of the knight.

Returns

The name of the knight.

3.4.3.2 getNewPos()

```
int Move::getNewPos ( )  [inline]
```

Retrieves the new position of the piece.

Returns

The new position of the piece.

3.4.3.3 getOldPos()

```
int Move::getOldPos ( )  [inline]
```

Retrieves the old position of the piece.

Returns

The old position of the piece.

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

- Move.h

3.5 Square Class Reference

Represents a square on the chessboard.

```
#include <Square.h>
```

Public Member Functions

- **Square** ()

  Constructor for the Square class. Initialize the position number of the square as its declaration order.
- bool placeKnight (const string ∗knight)

  Places a knight on the square.
- const string ∗ getKnight () const

  Retrieves the knight on the square.
- const string ∗ removeKnight ()

  Removes the knight from the square.
- const string ∗ getPrevValue () const

  Removes the knight from the square.
- int getPosNo () const

  Retrieves the position number of the square.

3.5.1 Detailed Description

Represents a square on the chessboard.

3.5.2 Member Function Documentation

3.5.2.1 getKnight()

```
const string * Square::getKnight ( ) const
```

Retrieves the knight on the square.

Returns

Pointer to the knight on the square, or nullptr if no knight is present.

3.5.2.2 getPosNo()

```
int Square::getPosNo ( ) const
```

Retrieves the position number of the square.

Returns

The position number of the square.

3.5.2.3 getPrevValue()

```
const string * Square::getPrevValue ( ) const
```

Removes the knight from the square.

Returns

Pointer to the removed knight.

If a knight is present on the square, it is removed, and its value is stored as the previous value.

3.5.2.4 placeKnight()

```
bool Square::placeKnight (
            const string * knight )
```

Places a knight on the square.

Parameters

knight	Pointer to the knight to be placed.

Returns

True if the knight was successfully placed, false otherwise.

Returns false if the square already contains a knight. Check if the square already contains a knight no more than one piece in the square. no piece takes other piece.

3.5.2.5 removeKnight()

```
const string * Square::removeKnight ( )
```

Removes the knight from the square.

Returns

Pointer to the removed knight.

If a knight is present on the square, it is removed, and its value is stored as the previous value.

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

- Square.h
- Square.cpp

Chapter 4

File Documentation

4.1 ChessBoard.cpp File Reference

Implementation of the ChessBoard class methods.

```
#include "ChessBoard.h"
```

4.1.1 Detailed Description

Implementation of the ChessBoard class methods.

Author

eslam

Date

March 2024

4.2 ChessBoard.h File Reference

Declaration of the ChessBoard class.

```
#include "Square.h"
#include <vector>
#include <iostream>
```

Classes

- class ChessBoard

  Represents a chessboard.

4.2.1 Detailed Description

Declaration of the ChessBoard class.

The ChessBoard class provides functionality to manage the chessboard state, including placing and moving knights on the board.

Author

eslam

Date

March 2024

4.3 ChessBoard.h

Go to the documentation of this file.

```
/*******************************************************************/
#pragma once

#ifndef CHESSBOARD_H
#define CHESSBOARD_H

#include "Square.h"
#include <vector>
#include <iostream>
using namespace std;

class ChessBoard
{
public:
    ChessBoard();

    void printBoard();

    Square* getSquare(int row, int col);

    Square* getSquare(int posNo);

private:
    // 2D matrix to store the chessboard squares.
    vector<vector<Square» squars;

    // Values of knights.
    vector<string> knights;

};// class ChessBoard

#endif // !CHESSBOARD_H
```

4.4 Graph.cpp File Reference

Implementation of the Graph class methods.

```
#include "Graph.h"
```

4.4.1 Detailed Description

Implementation of the Graph class methods.

Author

eslam

Date

March 2024

4.5 Graph.h File Reference

Declaration of the Graph class and the Cost class.

```
#include "ChessBoard.h"
#include "Move.h"
#include <utility>
#include <deque>
#include <iostream>
```

Classes

- class Cost

  Represents the cost of moving a knight from one square to another.
- class Graph

  Represents a graph for solving the knight's puzzle on a chessboard.

4.5.1 Detailed Description

Declaration of the Graph class and the Cost class.

The Graph class encapsulates the logic for solving the knight's tour puzzle using graph traversal algorithms.

Author

eslam

Date

March 2024

4.6 Graph.h

Go to the documentation of this file.

```
/*********************************************************************/
#pragma once
#ifndef GRAPH_H
#define GRAPH_H

#include "ChessBoard.h"
#include "Move.h"
#include <utility>
#include <deque>
#include <iostream>


using namespace std;

class Cost {
    friend class Graph;
private:
    Cost(int whiteCost, int blackCost) : whiteCost(whiteCost), blackCost(blackCost) {}

    int getCost(int color) {
         return color ? blackCost: whiteCost;
    }

    // The cost of moving a white knight.
    int whiteCost;

    // The cost of moving a black knight.
    int blackCost;
}; // class Cost


class Graph
{
public:
    Graph(ChessBoard& board);

    void printGraph();

    vector<Move*>* solvePuzzle();
private:
    // Represents the nodes (squares) of the graph, each containing a cost and a pointer to a Square object.
    vector<
        vector
        <pair<Cost, Square*>
        >
    > nodes;

    // ChessBoard to solve.
    ChessBoard* board;

    // Helper methods

    void addNeighborSquaresToQueue(vector<pair<Cost, Square*»* neighborSquares, deque<int>& positionsToProcess);

    int getKnightColor(Square* square);

    bool tryMovingKnight(Square* currentSquare, vector<pair<Cost, Square*»* neighborSquares, int knightColor,
        int& completedSwaps, int& totalMoves, vector<Move*>* allMoves, deque<int>& positionsToProcess);

    bool canMoveToSquare(pair<Cost, Square*>* neighborPair, Square* currentSquare, int knightColor);

    void performMove(pair<Cost, Square*>* neighborPair, Square* currentSquare);

    void checkForFinalDestination(pair<Cost, Square*>* neighborPair, bool& isFinalDestination, int& completedSwaps, int knightColor);

    void storeMoveData(vector<Move*>* allMoves, Square* currentSquare, pair<Cost, Square*>* neighborPair);

    void addNeighborsToQueue(vector<pair<Cost, Square*»* neighborSquares, deque<int>& positionsToProcess, int destinationIndex);

    void printMoveDetails(int& totalMoves, Square* currentSquare, vector<Move*>* allMoves);

    void printTotalMoves(int totalMoves);
};// class Graph

#endif // !GRAPH_H
```

4.7 Move.h File Reference

Declaration of the Move class.

```
#include <string>
#include <iostream>
```

Classes

- class Move

  Represents a move made on a chessboard.

4.7.1 Detailed Description

Declaration of the Move class.

Author

eslam

Date

March 2024

4.8 Move.h

Go to the documentation of this file.

```
/*******************************************************************/
#pragma once

#ifndef MOVE_H
#define MOVE_H

#include <string>
#include <iostream>
using namespace std;

class Move
{
public:
    Move(int oldPos, int newPos, string knight):
        oldPos(oldPos), newPos(newPos), knight(knight) {}

    string getknight() {
        return this->knight;
    }

    int getOldPos() {
        return this->oldPos;
    }

    int getNewPos() {
        return this->newPos;
    }

    void printMove() {
        cout « "Move: " « knight « " from " « oldPos « " to " « newPos « endl;
    }
private:
    // The old position of the piece.
    int oldPos;
    // The new position of the piece.
    int newPos;
    // The name of the knight making the move.
    string knight;
};// class Move



#endif // !MOVE_H


```

4.9 Square.cpp File Reference

Implementation of the Square class.

```
#include "Square.h"
```

4.9.1 Detailed Description

Implementation of the Square class.

Author

eslam

Date

March 2024

4.10 Square.h File Reference

Declaration of the Square class.

```
#include <string>
```

Classes

- class Square
  Represents a square on the chessboard.

4.10.1 Detailed Description

Declaration of the Square class.

Author

eslam

Date

March 2024

4.11 Square.h

Go to the documentation of this file.

```
/*******************************************************************/
#pragma once
#ifndef SQUARE_H
#define SQUARE_H

#include <string>

using namespace std;

class Square
{
public:
    Square() : posNo(counter++), knight(nullptr), prevValue(nullptr) {}

    bool placeKnight(const string* knight);

    const string* getKnight() const;

    const string* removeKnight();

    const string* getPrevValue() const;

    int getPosNo() const;

private:
    //Static counter for numbering the squares.
    static int counter;
    //The position number of the square.
    int posNo;
    const string* knight;

    // The name of the last piece entered the square, used for certain conditions.
    const string* prevValue;

};// class Square
#endif // !SQUARE_H
```

Index