Algorithms – Programming Assignment 4

Slider Puzzle

B06602035 李晴妍

E-Mail : b06602035@ntu.edu.tw

Phone : 0987883537

System information

Operating system: Windows

Compiler: IntelliJ IDEA

Text editor / IDE: IntelliJ IDEA

Explain briefly how you represented the Board data type.

Data type:

private int n;

n-by-n board

private int[] board;

n-by-n board

private int ham;

number of tiles out of place

private int man;

sum of Manhattan distances between tiles and goal

private int zero;

the index of 0 in board

Function:

public Board(int[][] tiles)

create a board from an n-by-n array of tiles, where tiles[row][col] = tile at (row, col)

public Board(int[] tiles)

create a [n*n] board from an n-by-n array of tiles

public String toString()

string representation of this board

public int tileAt(int row, int col)

tile at (row, col) or 0 if blank

public int size()

board size n

public int hamming()

number of tiles out of place

public int manhattan()

sum of Manhattan distances between tiles and goal

public boolean isGoal()

check if this board is the goal board?

public boolean equals(Object y)

check if this board is equal y

public Iterable<Board> neighbors()

put all the neighbors beside 0 into the queue

private Board exchange(Board a, int i, int j)

exchange the value of index i and j in board a

public boolean isSolvable()

check if this board is solvable

When n is odd, an n-by-n board is solvable if and only if its number of inversions is even.

When n is even, an n-by-n board is solvable if and only if the number of inversions plus the row of the blank square is odd.

Explain briefly how you represented a search node (board + number of moves + previous search node).

NODE CONTAINING PREVIOUS NODE, BOARD AND THE TIMES OF MOVING:

private class SearchNode implements Comparable<SearchNode>

private SearchNode pre;

parent node / previous node

private Board board;

the board after moving

private int moves;

the times of moving

public SearchNode(Board initial, SearchNode previous)

(Default constructor)

this.board = initial;

this.pre = previous;

check if pre == null, then moves = 0, otherwise moves = pre.moves + 1

@Override

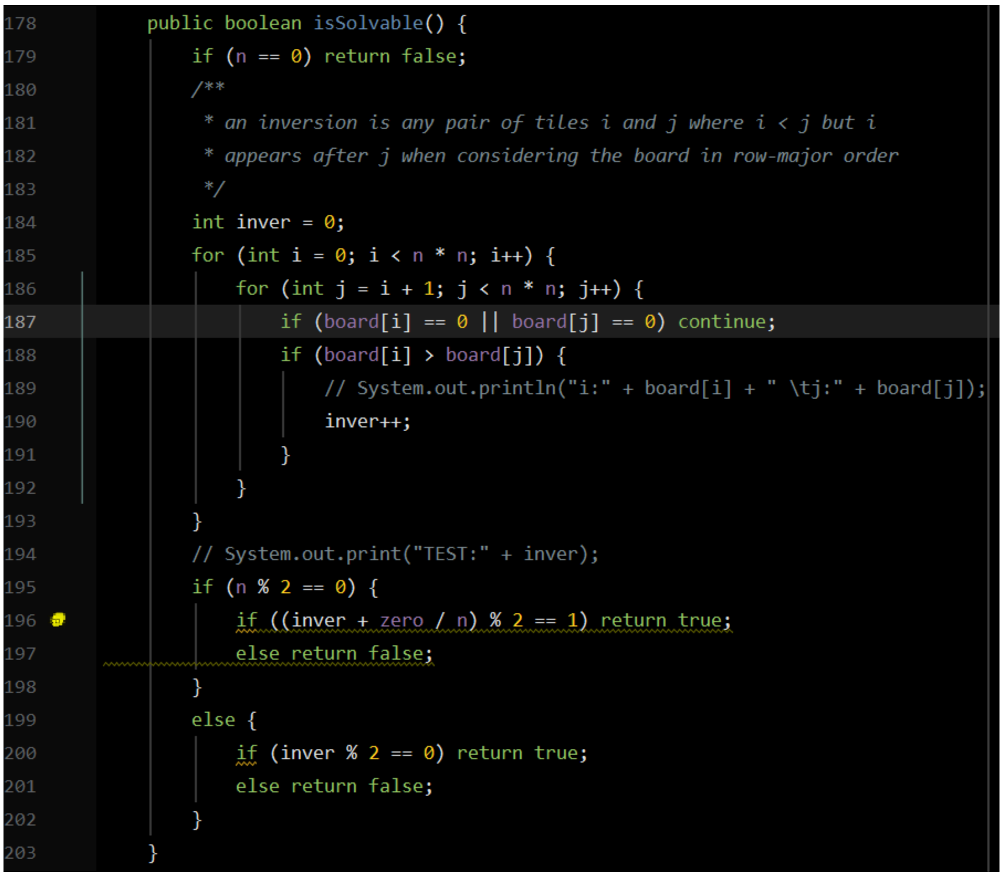
public int compareTo(SearchNode a)

Override function to compare 2 SearchNode by the sum of manhattan and moves

Explain briefly how you detected unsolvable puzzles. What is the order of growth of the running time of your isSolvable() method in the worst case as function of the board size n? Use big Theta notation to simplify your answer, e.g., Theta(n log n) or Theta(n^3).

Description:

I used the function public boolean isSolvable() in Board class to detected unsolvable puzzles in Solver default constructor. We know an inversion is any pair of tiles i and j where i<j but i appears after j when considering the board in row-major order. Briefly, when n is odd, an n-by-n board is solvable if and only if its number of inversions is even. And when n is even, an n-by-n board is solvable if and only if the number of inversions plus the row of the blank square is odd. So, we need to run n^4 time in the worst case to check any pair of tiles in n-by-n board to determine if this board is solvable or not.



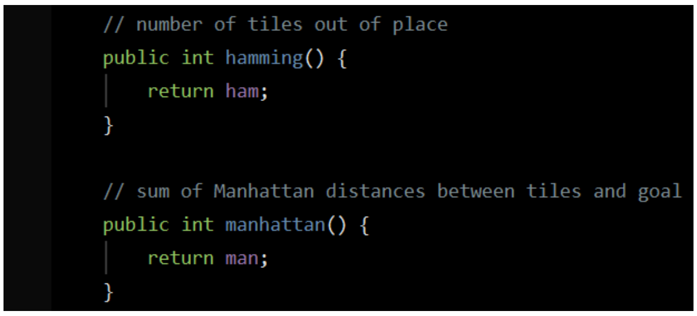
```java
    public boolean isSolvable() {
        if (n == 0) return false;
        /**
         * an inversion is any pair of tiles i and j where i < j but i
         * appears after j when considering the board in row-major order
         */
        int inver = 0;
        for (int i = 0; i < n * n; i++) {
            for (int j = i + 1; j < n * n; j++) {
                if (board[i] == 0 || board[j] == 0) continue;
                if (board[i] > board[j]) {
                    // System.out.println("i:" + board[i] + " \tj:" + board[j]);
                    inver++;
                }
            }
        }
        // System.out.print("TEST:" + inver);
        if (n % 2 == 0) {
            if ((inver + zero / n) % 2 == 1) return true;
            else return false;
        }
        else {
            if (inver % 2 == 0) return true;
            else return false;
        }
    }
```

Order of growth of running time: Theta(n^4)

For each of the following instances, give the minimum number of moves to solve the instance (as reported by your program). Also, give the amount of time your program takes with both the Hamming and Manhattan priority functions. If your program can't solve the instance in a reasonable amount of time (say, 5 minutes) or memory, indicate that instead. Note that your program may be able to solve puzzle[xx].txt even if it can't solve puzzle[yy].txt and xx > yy.

	MIN NUMBER	(SECONDS)	
INSTANCE	OF MOVES	HAMMING	MANHATTAN
------------	----------	----------	----------
PUZZLE28.TXT	12669	0.000	0.000
PUZZLE30.TXT	47412	0.000	0.000
PUZZLE32.TXT	155657	0.000	0.000
PUZZLE34.TXT	148474	0.000	0.000
PUZZLE36.TXT	958447	0.000	0.000
PUZZLE38.TXT	3213351	0.000	0.000
PUZZLE40.TXT	183181	0.000	0.000
PUZZLE42.TXT	1429580	0.000	0.000

(The order of Hamming and Manhattan function is constant time, so they always cost 0.00s.)



```java
// number of tiles out of place
public int hamming() {
    return ham;
}

// sum of Manhattan distances between tiles and goal
public int manhattan() {
    return man;
}
```

Describe whatever help (if any) that you received. Don't include readings, lectures, and precepts, but do include any help from people (including course staff, lab TAs, classmates, and friends) and attribute them by name.

https://github.com/ufarobot/8-puzzle/tree/master/src

https://github.com/Mamie/8-puzzle

Describe any serious problems you encountered.

* MinPQ 第一次用，不太熟悉
* 如果 input board 不是正確的數字或是有重複的數字沒辦法去辨別

List any other comments here. Feel free to provide any feedback on how much you learned from doing the assignment, and whether you enjoyed doing it.

* 這次需要自己建一個 SearchNode，蠻有趣的，需要先想好架構再開始寫。
* 作業有規定各個 function 的 order，所以不是有做出來就好，要想適合的方式去減少 order。