

Nine puzzle

Report (ITRW 317)

By:

Anneke Lourens

24087122

Due:

18 March 2016

Contents

1	Introduction	3
1.1	Background	3
2	Literature Study	3
2.1	History	3
3	User Guide	3
3.1	How to make a .csv file	3
3.2	How to begin the Nine-puzzle	4
3.3	Functionality	6
4	Code	7

1 Introduction

1.1 Background

The nine puzzle is a 3×3 board with eight tiles also called cells, and the ninth one is blank space. The blank space is always in the lower right corner of the puzzle. The objective of the nine puzzle is to slide the tiles/cells into the space to make a picture complete or arrange the tiles in numerical order. The nine puzzle is the only puzzle that can be completely solved. If we look at the nine puzzle, there are $9!$ permutations that the board can be solved (Reinefeld, 1993). There are $9!/2 = 181440$ possible ways to solve the nine puzzle. The problem requires an average of 22 moves to complete the puzzle (Reinefeld, 1993). The nine puzzle cannot be solved if the numbers one to seven is arranged in order from left to right, top to bottom with the seventh and eight number that has been switched. We want to optimize the moves so that the puzzle can be solved in the fewest moves. The n puzzle is considered an NP-Hard problem when the minimum amount of moves need to be answered.

2 Literature Study

2.1 History

Sam Loyd, the impish puzzle maker, introduced the 15-puzzle to the United States, Britain, and Europe in the 1870's. (Archer, 1999). The original fifteen puzzle had numbers from 1 - 15 on the tiles. If one would have bought the 15 puzzle back then the numbers from 1 to 13 was arranged in the right order from left to right, top to bottom with the blank space in the lower right corner, and the 14 and 15 have been reversed. Loyd drove the world crazy with the puzzle, and he offered \$1 000 to the first person that could solve the problem. The 15 puzzle was impossible to solve if the numbers 14 and 15 has been switched. The 15 puzzle cannot be solved with the blank space in the lower right corner. It is only possible to address the problem if there are an even number of moves, so the resulting permutation is even (Archer, 1999).

The puzzle then becomes in many sizes. Like the nine puzzle that is the smallest and there exists a 24 puzzle as well.

3 User Guide

3.1 How to make a .csv file

Open the file named "waarde_puzzle.csv" in your favorite text editor e.g. Notepad++, Notepad, Vim, Nano. If you want to change the order of the numbers, just remember that there must only be a "," the numbers in between the numbers. The numbers that are used is 1 to 8 and a lowercase "b" that will represent the blank space Figure 1.

Figure 2 is an example of what the .csv file looks like in Notepad++ or Notepad.

The first row in the file is the initial puzzle that the player must solve according to the solved puzzle. The second row in the file is the solved puzzle. If the player has saved the puzzle to continue at a later stage, then there will be a third row in the .csv file that shows the moves that the player has made already.




 execute.bat	3/15/2016 9:02 PM	Windows Batch File	1 KB
 NinePuzzle.java	3/16/2016 11:22 AM	JAVA File	8 KB
 waardes_puzzle.csv	3/16/2016 2:58 PM	Microsoft Excel C...	1 KB

Figure 1: This is the files that one get which will run the nine puzzle

1	b, 1, 2, 3, 4, 5, 6, 7, 8
2	1, b, 2, 3, 4, 5, 6, 7, 8
3	

Figure 2: How the data looks in the .csv file. This data will be used in the nine puzzle.

3.2 How to begin the Nine-puzzle

Step 1:

If you have your own .csv file - right click on the execute.bat file and open it in Notepad++ or Notepad Figure 1. Change the file name of the .csv file to your own .csv file's name. Save the execute.bat file and exit it. Double-click on the execute.bat file to run the program Figure 3a.

Step 2:

If the program has run correctly then you will get a message telling you how to play the nine puzzle. Figure 3b shows the message dialog to inform the player on the rules of the nine-puzzle game.

Step 3:

Enter a "y" for you are a new player or "n" for you are not a new player. Figure 4 shows how the program asks if you are a new player or not.

Step 4:

After Step 3 has been completed the begin puzzle will be given to the player. Figure 5 shows Step 4.

Step 5:

Enter the number that you want to move to the empty space called "b". Figure 6 shows how the number has been moved to the empty space "b" and gives then the updated puzzle.

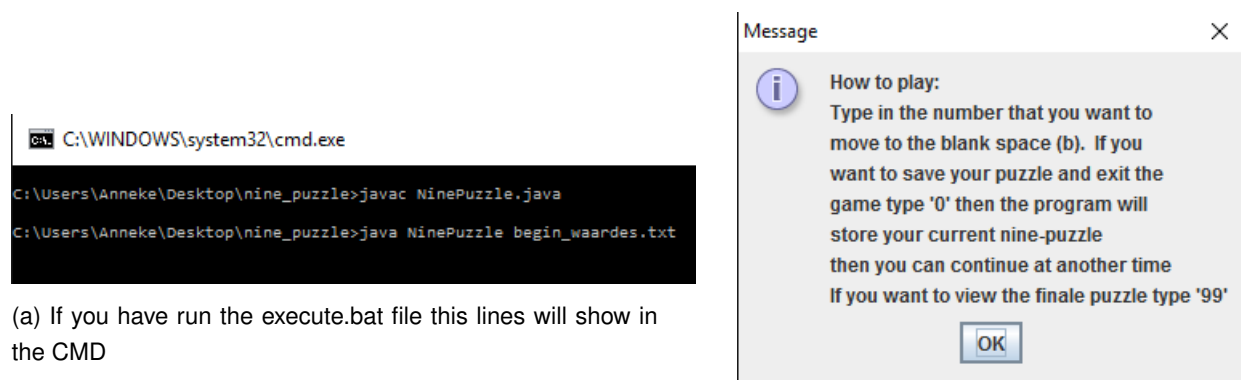


Figure 3: The beginning of the program

```
C:\WINDOWS\system32\cmd.exe

C:\Users\Anneke\Desktop\nine_puzzle>javac NinePuzzle.java

C:\Users\Anneke\Desktop\nine_puzzle>java NinePuzzle waardes_puzzle.csv
new player? y or n
```

Figure 4: New player or not

```
C:\WINDOWS\system32\cmd.exe

C:\Users\Anneke\Desktop\nine_puzzle>javac NinePuzzle.java

C:\Users\Anneke\Desktop\nine_puzzle>java NinePuzzle waardes_puzzle.csv
new player? y or n
y
Current Puzzle
b 1 2
3 4 5
6 7 8
What is your next move?
```

Figure 5: The begin puzzle is given to the player

```
C:\WINDOWS\system32\cmd.exe

C:\Users\Anneke\Desktop\nine_puzzle>javac NinePuzzle.java

C:\Users\Anneke\Desktop\nine_puzzle>java NinePuzzle waardes_puzzle.csv
new player? y or n
y
Current Puzzle
b 1 2
3 4 5
6 7 8
What is your next move?
1
Current Puzzle
1 b 2
3 4 5
6 7 8
What is your next move?
```

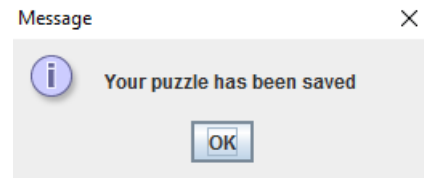
Figure 6: After a number has been moved to the empty space

```
C:\WINDOWS\system32\cmd.exe

C:\Users\Anneke\Desktop\nine_puzzle>javac NinePuzzle.java

C:\Users\Anneke\Desktop\nine_puzzle>java NinePuzzle waardes_puzzle.csv
new player? y or n
y
Current Puzzle
b 1 2
3 4 5
6 7 8
What is your next move?
1
Current Puzzle
1 b 2
3 4 5
6 7 8
What is your next move?
0
```

(a) If you type in 0



(b) Message dialog when the player saves

Figure 7

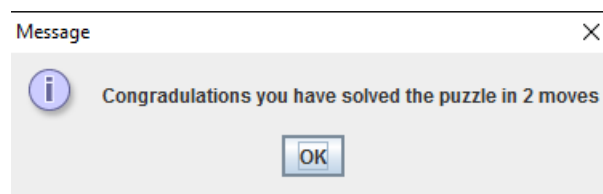


Figure 8: The congratulation message

3.3 Functionality

Stop, save and exit of the program:

Figure 7a shows that the player has type in "0" to save their puzzle to continue at a later stage. Figure 7b show the message that the program will give to the player is the puzzle have been saved to the .csv file. After the player has pressed "OK" the program will exit.

If the game has been won:

Figure 8 will show if the player has won the game by solving the puzzle according to the final puzzle specifying in the .csv file.

How to see the goal puzzle that one must obtain from the initial puzzle.

Figure 9 shows that the player has entered "99" to see what the goal puzzle must look like. The program will write "Goal Puzzle" and directly after the goal puzzle have been given the program will give the puzzle that the player must use to obtain the goal puzzle.

```

C:\WINDOWS\system32\cmd.exe

C:\Users\Anneke\Desktop\nine_puzzle>javac NinePuzzle.java

C:\Users\Anneke\Desktop\nine_puzzle>java NinePuzzle waardes_puzzle.csv
new player? y or n
y
Current Puzzle
b 1 2
3 4 5
6 7 8
what is your next move?
99
Goal Puzzle
1 b 2
3 4 5
6 7 8

Current Puzzle
b 1 2
3 4 5
6 7 8
what is your next move?

```

Figure 9: Goal puzzle is given in the program

4 Code

```

import java.util.Scanner;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;
import java.io.PrintWriter;
import javax.swing.JOptionPane;

public class NinePuzzle {
    public static void main(String[] args) throws FileNotFoundException {
        // scanner to read out of txt file
        Scanner input = new Scanner(args[0]);
        // gebruiker se skuiwe
        Scanner gebruiker = new Scanner(System.in);
        boolean solved = false;
        boolean save = false;
        int count = 0;
        // create arrays
        int[] current = new int[9];
        int[] finaal = new int[9];
        // array for the string read in csv file
        String[] userInput = new String[1];
        String filename = input.nextLine();
        // call method that reads the csv file
        setUpArray(current, finaal, userInput, filename);
        // display how the game is played
        String message = "How to play:\n" +
            "Type in the number that you want to\n" +
            "move to the blank space (b). If you\n" +
            "want to save your puzzle and exit the\n" +
            "game type '0' then the program will\n" +
            "store your current nine-puzzle\n" +
            "then you can continue at another time\n" +
            "If you want to view the final puzzle type '99'";
        JOptionPane.showMessageDialog(null, message);
        System.out.println("new player? y or n");
        String newUser = gebruiker.nextLine();
        String yn = newUser.substring(0, 1);
        if (yn.equals("y") || yn.equals("Y")) {
            userInput[0] = "";
            writeArrayToFile(current, finaal, userInput[0], filename);
        }
        count = userInput.length;
    }
}

```

```

while ( !solved && !save) {
    System.out.println("Current_Puzzle");
    puzzle(current); // print current puzzle
    int lees = 0;
    // ask user what his next move is going to be
    System.out.println("What_is_your_next_move?");
    lees = gebruiker.nextInt();
    // test if the input from user is 0 or any other number between 1 – 8
    if ( lees != 0 ) {
        if(lees == 99) {
            System.out.println("Goal_Puzzle");
            puzzle(finaal);
            System.out.println("");
        } else {
            // move the index of the tiles
            if (move_index(lees, current)) {
                count++;
                solved = compare_solution(finaal, current);
                userInputs[0] += Integer.toString(lees) + ",";
                writeArrayToFile(current, finaal, userInputs[0], filename);
            }
        }
    } else { //if the user enters 0
        //write current puzzle to csv file
        writeArrayToFile(current, finaal, userInputs[0], filename);
        //stoor na txt file as begin waardes
        save = true;
    } //end if
    //show the final puzzle to the user
} // end while
// show message that say he has solved the puzzle
if ( solved ) {
    count = count-1;
    String message1 = "Congradulations_you_have_solved_the_puzzle_in" + " " + count + " " + "moves";
    JOptionPane.showMessageDialog(null, message1);
}

    if(save){
        String message2 = "Your_puzzle_has_been_saved";
        JOptionPane.showMessageDialog(null, message2);
    }
} //end main

public static void writeArrayToFile(int [] current,
                                    int [] finaal,
                                    String userinputs,
                                    String filename)
    throws FileNotFoundException {
    try {
        PrintWriter outputStream = new PrintWriter(filename);
        for ( int k = 0; k < 9; k++ ) {
            if ( k != 8 ) {
                // write the current puzzle to file
                if(current[k] == 0) {
                    outputStream.print("b,");
                } else {
                    outputStream.print(current[k] + ",");
                }
            } else {
                outputStream.print(current[k] + "\r\n");
            }
        }
    }
    for(int h = 0; h < 9; h++ ) {
        if (h != 8) {
            if (finaal[h] == 0) {
                outputStream.print("b,");
            } else {
                outputStream.print(finaal[h] + ",");
            }
        } else {
            outputStream.print(finaal[h] + "\r\n");
        }
    }
}

```



```

        } //end else
    }
    if (userinputs.length() > 0) {
        String user_inputs = userinputs.substring(0, userinputs.length() - 1) + "\r\n";
        outputStream.println(user_inputs);
    }
    outputStream.close();
} catch (FileNotFoundException e) {
    e.printStackTrace();
} //end catch
}
//read csv file into an array
public static void setUpArray(int [] current, int [] finaal,
                               String [] userinputs, String filename)
                               throws FileNotFoundException {

    File File = new File(filename);
    try {
        Scanner inputStream = new Scanner(File);
        int lines_read = 0;
        while (inputStream.hasNext() ) {
            String data = inputStream.next(); //gets a whole line
            String [] values = data.split(",");
            if ( lines_read == 0 ) {
                for ( int v = 0; v < 9; v++ ) {
                    if (values[v].equals("b")) {
                        current[v] = 0;
                    } else {
                        current[v] = Integer.parseInt(values[v]);
                    }
                }
            } else if ( lines_read == 1 ) {
                for ( int w = 0; w < 9; w++ ) {
                    if (values[w].equals("b")) {
                        finaal[w] = 0;
                    } else {
                        finaal[w] = Integer.parseInt(values[w]);
                    }
                }
            } else if ( lines_read == 2 ) {
                if (!data.equals(null) || !data.equals("")) {
                    userinputs[0] = data + ",";
                } else {
                    userinputs[0] = "";
                }
            }
            lines_read++;
        }
        inputStream.close();
    } catch (FileNotFoundException e) {
        e.printStackTrace();
    }
} //end setuparray
//print puzzle
public static int puzzle(int [] current) {
    for(int k = 0; k < 9; k++) {
        if (current[k] == 0) {
            System.out.print("b_");
        } else {
            System.out.print(current[k] + "_");
        }
        if ( ((k + 1) % 3) == 0 ) {
            System.out.print("\n");
        }
    } //end for
    return 0;
} //end puzzle

//swap the index of the tile that has been moved
public static boolean move_index(int lees, int [] current) {

```

```

int [][] lookup = {{1, 3, -1, -1},
                  {0, 2, 4, -1},
                  {1, 5, -1, -1},
                  {0, 4, 6, -1},
                  {1, 3, 5, 7},
                  {2, 4, 8, -1},
                  {3, 7, -1, -1},
                  {4, 6, 8, -1},
                  {5, 7, -1, -1}};
int zero_position = get_index(0, current);
int value_position = get_index(lees, current);
for ( int i = 0; i < 4; i++) {
    if ( lookup[zero_position][i] == value_position ) {
        current[value_position] = 0;
        current[zero_position] = lees;
        return true;
    }
}
return false;
} //end move_index
//get the index of the blank space and the number that the user want to move
public static int get_index(int lees, int [] current) {
    for ( int i = 0; i < 9; i++ ) {
        if ( lees == current[i] ) {
            return i;
        }
    }
    return 0;
}
//compare whether the current puzzle has the same values of the final puzzle
public static boolean compare_solution(int [] finaal, int [] current) {
    for ( int i = 0; i < 9; i++ ) {
        if ( finaal[i] != current[i] ) {
            return false;
        }
    }
    return true;
} //end compare
} //end class

```

References

- Archer, A. F. (1999). A modern treatment of the 15 puzzle, *The American mathematical monthly* **106**(9): 793–799.
- Reinefeld, A. (1993). Complete solution of the eight-puzzle and the benefit of node-ordering in ida*, pp. 248–253.