University of Montpellier, Faculty of Sciences

Artificial Intelligence for Software Engineering

Master 2 Software Engineering

HAI916I

Lab 1 Solution - Sudoku

Realized by:
Mr. Ilyas AMIRAT

Supervised by:
Dr. Nadjib Lazaar

Academic Year : 2022/2023

Contents

List of Figures

Chapter 1

Lab Solution

Github Repo Link : https://github.com/amixyas/sudoku-ppc

```
Menu :
1 : Make comparison for one n*n solution between the Sudoku PPC model and the Sudoku BackTrack, in term of time.
2 : Make comparison for all n*n solutions between the Sudoku PPC model and the Sudoku BackTrack, in term of time.
3 : Solving Sudoku(9*9) with a provided pre-incomplete-instance.
4 : Solving Sudoku(16*16) with a provided pre-incomplete-instance.
5 : Solving 'Greater Than' Sudoku(16*16) with a provided pre-incomplete-instance.
Notes for 3 and 4 : - The provided pre-incomplete-instance, it's must be written in txt file please.
                    - To get a hint , take a look at resources folder.
                    - If you get an unwanted results, know that's due to the content of your file, please review it.

So, what do you choose :
```

Figure 1.1: Menu.

1.1 Question 1

Making comparison for one NxN solution between the Sudoku PPC model and the Sudoku BackTrack, in term of time.

```
So, what do you choose : 1
Please, provide to system the sudoku 'n' number, to get sudoku of n*n, for example 4 or 9 : 4

[1, 2, 3, 4]
[3, 4, 1, 2]
[2, 1, 4, 3]
[4, 3, 2, 1]

Start : 68071168089896 -- End : 68083860728236 -- So, Duration is: 12692 milliseconds
┌───┬───┬───┬───┐
│ 3 │ 4 │ 1 │ 2 │
├───┼───┼───┼───┤
│ 1 │ 2 │ 3 │ 4 │
├───┼───┼───┼───┤
│ 2 │ 3 │ 4 │ 1 │
├───┼───┼───┼───┤
│ 4 │ 1 │ 2 │ 3 │
└───┴───┴───┴───┘
Start : 68083890338658 -- End : 68084022110256 -- So, Duration is: 131 milliseconds

 >>> As we see PPC is mush more faster then BackTrack
```

Figure 1.2: Question1.

1.2 Question 2

Making comparison for all NxN solutions between the Sudoku PPC model and the Sudoku BackTrack, in term of time. Figure 1.3 and Figure 1.4 shows captures (start and end) of all solutions using BackTrack, which takes to solve all of them 96325 milliseconds.

```
So, what do you choose : 2
Please, provide to system the sudoku 'n' number, to get sudoku of n*n, for example 4 or 9 : 7

Please, choose 4 or 9 for option 1 and just 4 for option 2
We don't want you to burn down your computer processor :)
ReInput : 4

[1, 2, 3, 4]
[3, 4, 1, 2]
[2, 1, 4, 3]
[4, 3, 2, 1]


[1, 2, 3, 4]
[3, 4, 1, 2]
[2, 3, 4, 1]
[4, 1, 2, 3]


[1, 2, 3, 4]
[3, 4, 1, 2]
```

Figure 1.3: Question2-1.

```
[1, 4, 3, 2]



[4, 3, 2, 1]
[2, 1, 4, 3]
[3, 4, 1, 2]
[1, 2, 3, 4]

Start : 72014412865597 -- End : 72105046080523 -- So, Duration is: 90633 milliseconds
```

Figure 1.4: Question2-2.

Figure 1.5 and Figure 1.6 shows half-of-captures (start and end) of all solutions (287) using PPC, which takes to solve all of them 129 milliseconds.

```
Solutions PPC >>
Solution: [0][0]=3, [0][1]=2, [0][2]=1, [0][3]=4, [1][0]=1, [1][1]=4, [1][2]=3, [1][3]
Solution: [0][0]=1, [0][1]=2, [0][2]=3, [0][3]=4, [1][0]=3, [1][1]=4, [1][2]=1, [1][3]
Solution: [0][0]=1, [0][1]=4, [0][2]=3, [0][3]=2, [1][0]=3, [1][1]=2, [1][2]=1, [1][3]
Solution: [0][0]=4, [0][1]=1, [0][2]=3, [0][3]=2, [1][0]=3, [1][1]=2, [1][2]=1, [1][3]
```

Figure 1.5: Question2-3.

```
Solution: [0][0]=4, [0][1]=3, [0][2]=1, [0][3]=2, [1][0]=1, [1][1]=2, [1][2]=4, [1][3]=3,
Solution: [0][0]=4, [0][1]=3, [0][2]=1, [0][3]=2, [1][0]=1, [1][1]=2, [1][2]=4, [1][3]=3,
Solution: [0][0]=4, [0][1]=3, [0][2]=1, [0][3]=2, [1][0]=2, [1][1]=1, [1][2]=4, [1][3]=3,
Start : 68651965812216 -- End : 68652095566130 -- So, Duration is: 129 milliseconds
```

Figure 1.6: Question2-4.

1.3 Question 3

Solving Sudoku(9*9) with a provided pre-incomplete-instance. Figure 1.7 shows the result, Figure 1.8 shows the content of the input file, Figure1.9 shows my Matrix Reader (file reader), Figure1.10 shows the code modifications

```
So, what do you choose : 3
Please, provide to system the pre-incomplete-instance file path :
/home/ilyas/Documents/#Others/Univ-montpellier/Tps-um/IAGL/tp1-sudoku/src/main/resources/SudokuPPCLevel2.txt
```

8	1	2	7	5	3	6	4	9
9	4	3	6	8	2	1	7	5
6	7	5	4	9	1	2	8	3
1	5	4	2	3	7	8	9	6
3	6	9	8	4	5	7	2	1
2	8	7	1	6	9	5	3	4
5	2	1	9	7	4	3	6	8
4	3	8	5	2	6	9	1	7
7	9	6	3	1	8	4	5	2

Figure 1.7: Question3-1.

```
ilyas@machine: ~/.../resources
ilyas@machine:~/.../resources$ cat SudokuPPCLevel2.txt
8 0 0 0 0 0 0 0 0
0 0 3 6 0 0 0 0 0
0 7 0 0 9 0 2 0 0
0 5 0 0 0 7 0 0 0
0 0 0 0 4 5 7 0 0
0 0 0 1 0 0 0 3 0
0 0 1 0 0 0 0 6 8
0 0 8 5 0 0 0 1 0
0 9 0 0 0 0 4 0 0
ilyas@machine:~/.../resources$ pwd
/home/ilyas/Documents/#Others/Univ-montpellier/Tps-um/IAGL/tp1-sudoku/src/main/resources
ilyas@machine:~/.../resources$
```

Figure 1.8: Question3-2.

```java
import java.io.File;
import java.util.Scanner;

public class MatrixReader {

    public static String[][] reader(String[][] matrix, String path, int n) {

        try {
            Scanner input = new Scanner(new File(path));

            while (input.hasNextLine()) {
                for (int i = 0; i < n; i++) {
                    for (int j = 0; j < n; j++) {
                        try {
                            String temp = input.next();
                            if (temp!=null) matrix[i][j] = temp;
                        } catch (java.util.NoSuchElementException e) {}
                    }
                }
            }
        } catch (Exception e) {}

        return matrix;
    }
}
```

Figure 1.9: Question3-3.

```java
        for (int i = 0; i < n; i++) {
            model.allDifferent(rows[i]).post();
            model.allDifferent(cols[i]).post();
            model.allDifferent(shapes[i]).post();
        }

        // Reading sudoku instance
        MatrixReader matrixReader = new MatrixReader();
        String[][] matrix = new String[n][n];

        if (sudokuLevel==2 || sudokuLevel==3) {
            matrix = matrixReader.reader(matrix, this.Path, n);
            for (int i = 0; i < n; i++) {
                for (int j = 0; j < n; j++) {
                    int input = 0;
                    try {
                        input = Integer.valueOf(matrix[i][j]);
                    } catch (NumberFormatException e) {
                        input = (int) matrix[i][j].charAt(0) - 55;
                    }
                    if(input != 0)rows[i][j].eq(input).post();
                }
            }
```

Figure 1.10: Question3-4.

1.4 Question 4

Solving Sudoku(16*16) with a provided pre-incomplete-instance. Figure 1.11 shows the result, Figure 1.12 shows the content of the input file, Figure1.9 shows my Matrix Reader (file reader), Figure1.10 shows the code modifications, Figure1.13 shows also the code modifications

```
So, what do you choose : 4
Please, provide to system the pre-incomplete-instance file path :
/home/ilyas/Documents/#Others/Univ-montpellier/Tps-um/IAGL/tp1-sudoku/src/main/resources/SudokuPPCLevel3.txt
```

A	G	E	1	F	8	9	6	4	B	D	5	7	2	3	C
6	C	8	F	1	B	4	E	2	7	3	A	D	G	5	9
B	9	5	D	2	3	G	7	F	E	C	1	6	4	8	A
2	7	4	3	A	C	D	5	9	8	G	6	1	B	F	E
7	4	6	5	8	F	2	3	A	1	B	E	9	D	C	G
8	F	G	A	9	7	C	4	D	6	5	3	B	1	E	2
C	1	3	E	D	A	6	B	7	G	9	2	8	F	4	5
9	D	B	2	E	G	5	1	8	4	F	C	3	7	A	6
G	B	A	8	4	2	1	C	3	9	7	F	E	5	6	D
5	6	F	9	3	D	A	G	E	2	1	B	C	8	7	4
D	2	1	4	6	E	7	9	5	C	8	G	A	3	B	F
3	E	7	C	B	5	F	8	6	D	A	4	G	9	2	1
E	5	9	6	G	1	3	F	B	A	4	7	2	C	D	8
1	A	2	G	7	4	B	D	C	5	6	8	F	E	9	3
4	3	D	B	C	6	8	2	G	F	E	9	5	A	1	7
F	8	C	7	5	9	E	A	1	3	2	D	4	6	G	B

Figure 1.11: Question4-1.

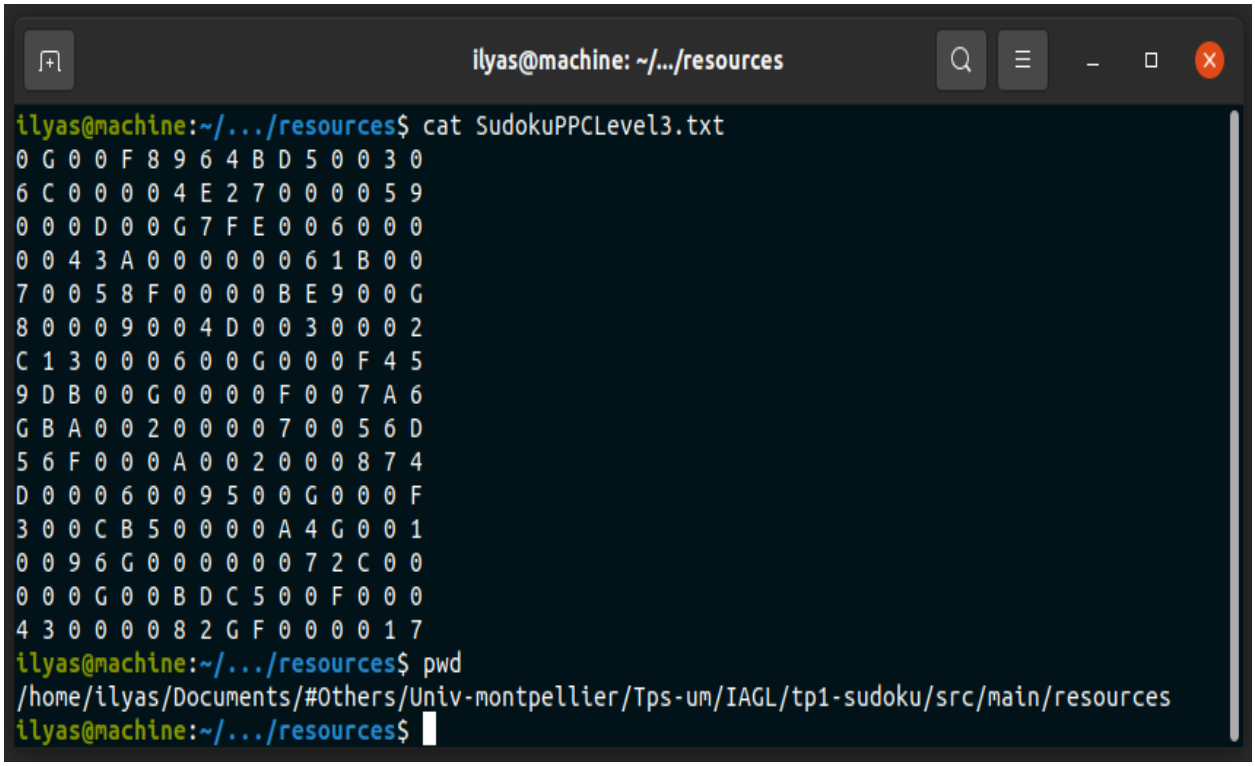

Figure 1.12: Question4-2.

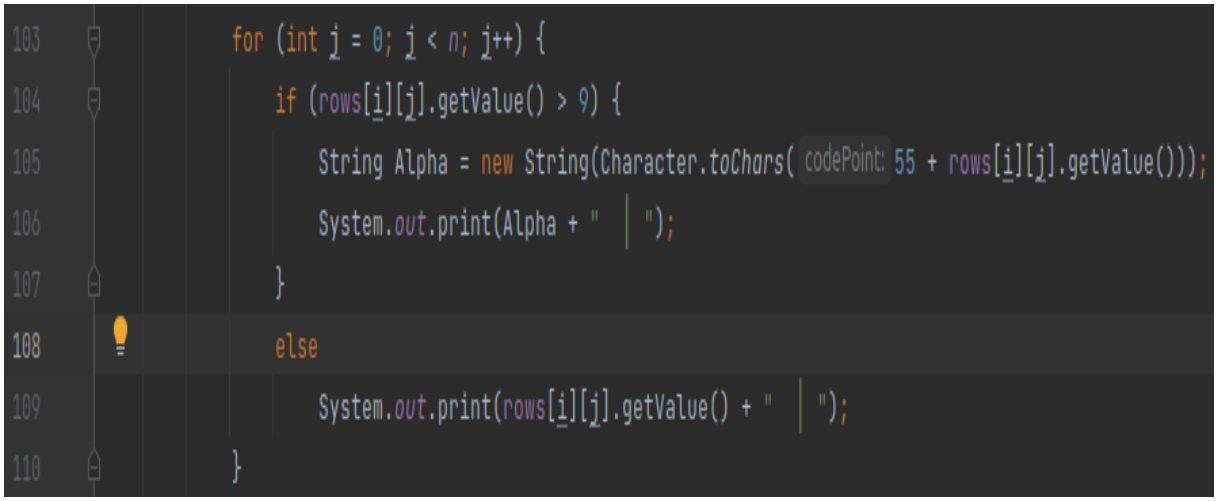

Figure 1.13: Question4-3.

1.5 Question 5

Solving 'Greater Than' Sudoku(16*16) with a provided constraints for both rows and cols. Figure 1.14 shows the result, Figure 1.15 and Figure 1.16 shows the content of the input file, Figure1.17 shows the code modifications

```
So, what do you choose : 5
Please, provide to system the pre-incomplete-instance file that contain row constraints :
/home/ilyas/Documents/#Others/Univ-montpellier/Tps-um/IAGL/tp1-sudoku/src/main/resources/SudokuPPCLevel4Row.txt
Please, provide to system the pre-incomplete-instance file that contain col constraints :
/home/ilyas/Documents/#Others/Univ-montpellier/Tps-um/IAGL/tp1-sudoku/src/main/resources/SudokuPPCLevel4Col.txt
```

2	6	7	1	3	9	8	5	4
1	9	3	5	4	8	7	6	2
8	5	4	6	7	2	3	9	1
9	3	8	2	1	5	6	4	7
6	2	1	7	9	4	5	3	8
4	7	5	3	8	6	2	1	9
5	1	9	8	2	3	4	7	6
7	8	6	4	5	1	9	2	3
3	4	2	9	6	7	1	8	5

Figure 1.14: Question5-1.

```
ilyas@machine: ~/.../resources
ilyas@machine:~/.../resources$ cat SudokuPPCLevel4Row.txt
< > | < < | > < |
< < | < < | < < |
> < | > > | > < |
> > | < > | < < |
< > | > > | > > |
< < | < > | < > |
> < | > > | < < |
> > | > > | > > |
> > | > < | > > |
ilyas@machine:~/.../resources$ pwd
/home/ilyas/Documents/#Others/Univ-montpellier/Tps-um/IAGL/tp1-sudoku/src/main/resources
ilyas@machine:~/.../resources$
```

Figure 1.15: Question5-2.

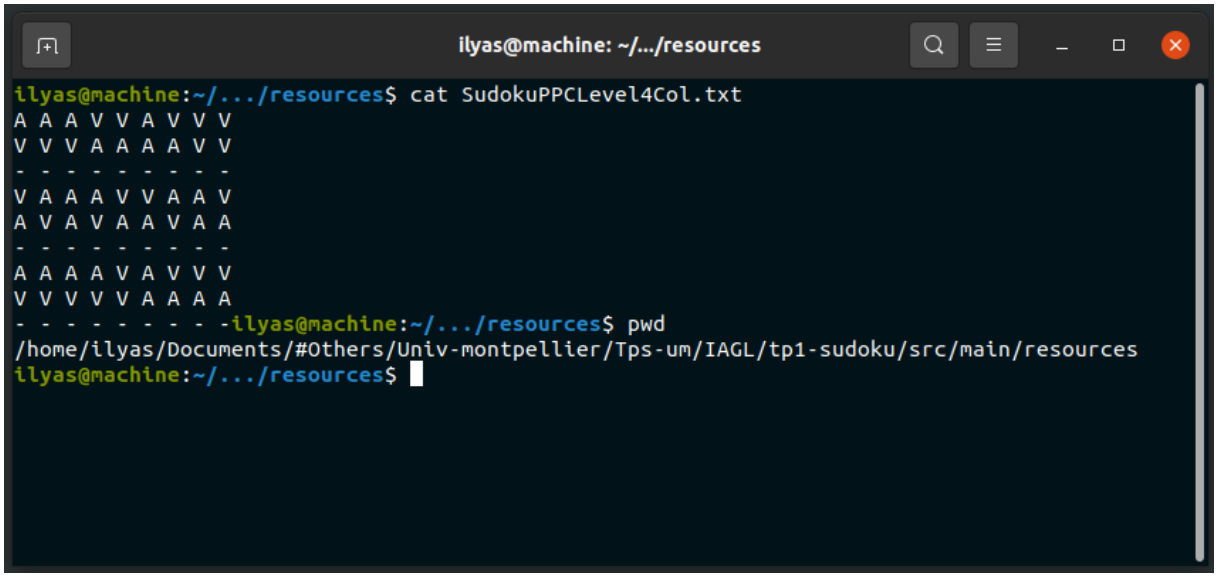

Figure 1.16: Question5-3.

```
// Reading sudoku instance
MatrixReader matrixReader = new MatrixReader();
String[][] matrix = new String[n][n];

matrix = matrixReader.reader(matrix, this.rowPath, n);

for (int i = 0; i < n; i++) {
    for (int j = 0; j < n - 1; j++) {
        if (matrix[i][j].equals("|")) continue;
        else if (matrix[i][j].equals("<")) rows[i][j].lt(rows[i][j + 1]).post();
        else rows[i][j].gt(rows[i][j + 1]).post();
    }
}

matrix = new String[n][n];
matrix = matrixReader.reader(matrix, this.colPath, n);

for (int i = 0; i < n-1; i++) {
    for (int j = 0; j < n; j++) {
        if (matrix[i][j].equals("V"))
            rows[i][j].gt(rows[i+1][j]).post();

        else if (matrix[i][j].equals("A"))
            rows[i][j].lt(rows[i+1][j]).post();}

}
```

Figure 1.17: Question5-4.