NAME: ARSHAN SHAFIEI

EMAIL:as12413@imperial.ac.uk

CID=00826335

Sudoku report

For this assignment, I used 4 different algorithms. This report will explain how the algorithms work, what advantages and disadvantages they have and evaluate and compare how quick they behave in different circumstances.

1) Standard backtracking algorithm:

This algorithm solves the Sudoku recursively. Basically, what it does is that it finds the first cell in the Sudoku, which is empty (assigned as 0). It then starts checking if the values from 1 to 9 can be written in that cell or not. As soon as a value can be written in that cell, we write the value there and the function calls it self again by moving to the next cell. In the case when no values can be written in an empty cell (this means that the previous assigned cell is wrong), we move back to the previous assigned cell and replace it with the next possible number. The function terminates and prints

the Sudoku as soon as we come out of the last position (position 81 as described in the comments in the code). If the function never comes out of the last position then the Sudoku cannot be solved (this happens when all values have been tested on the first empty cell and non solves the Sudoku).

Advantages:

1) This algorithm can solve all Sudoku's (if they can be solved)

Disadvantages:

1) It is not efficient since it uses no intuition

2) Logic algorithm:

This algorithm solves the Sudoku non-recursively and more like humans solves the Sudoku.
It finds the empty cells and checks if numbers 1 to 9 can be written in that cell or not. In contrast to backtracking that writes the value in the empty cell as soon as it can be written there, this algorithm writes the value in an empty cell only if that value is the only value that can be written there. Therefore the function checks all the numbers from 1 to 9 continuously in an empty cell. (More

details of implementation in the comments in the code). After writing the value in empty cells and reaching the end of the Sudoku, the program goes back to beginning of the Sudoku and does this process again and again until all the empty cells have been filled.

Advantages:

1) Faster than standard backtracking when there are a lot of empty cells

Disadvantages:

1) Not all Sudoku's can be solved (details bellow)

While checking this algorithm with different sudokus, I realized that this algorithm would not solve difficult sodukus. This is because in some cases a Sudoku will be solved by guessing a possible value in an empty cell.

Further improvements

3) Logic+ backtrack:

A good way of fixing the problem of logic algorithm not working when a value should be guessed in sodukus is combining it with backtracking algorithm. This means that we will fill the empty cells as much as we can using the logic algorithm and then use the backtracking algorithm to solve the rest of the sudoku. This algorithm is called logic backtrack in my source files.

4) Heuristic backtracking

This algorithm starts the backtracking from a different position than normal backtracking. As said before, backtracking starts checking all the numbers from 1 to 9 in positions and move to the next position if the value assigned is correct. This algorithm starts backtracking from the position, which has the least number of empty cells in its column, row and subgrid. The reason for that is because when there are few empty cells in the row, column and sub grid of a position, less possible numbers can be written in that position and therefore there is a more chance that the value assigned to that cell is the correct value (this means that the function does not need to come back and as a result efficiency increases).

Table comparing different algorithms

An example easy, hard, very hard, empty and full Sudoku are shown bellow.

Easy

```
0 4 0 8 0 1 0 6 7
6 0 0 0 9 0 0 0 8
0 0 0 0 0 4 1 2 0
1 9 0 0 4 0 0 7 5
0 0 8 0 0 0 2 0 0
2 5 0 0 8 0 0 3 1
0 1 6 3 0 0 0 0 0
3 0 0 0 5 0 0 0 6
4 8 0 2 0 6 0 5 0
```

hard

```
0 3 0 0 0 0 0 0 0
9 0 4 0 0 7 0 8 0
0 0 6 0 0 9 7 0 0
0 2 0 0 5 0 0 0 7
8 9 0 0 0 0 0 4 6
3 0 0 0 8 0 0 2 0
0 0 9 7 0 0 6 0 0
0 5 0 1 0 0 2 0 9
0 0 0 0 0 0 0 3 0
```

very hard

```
0 0 0 0 4 2 9 0 0
0 0 0 0 0 0 2 7 5
0 0 1 0 6 0 0 0 0
0 5 0 0 0 0 0 0 4
0 0 6 3 0 7 8 0 0
7 0 0 0 0 0 0 9 0
0 0 0 0 9 0 6 0 0
6 3 2 0 0 0 0 0 0
0 0 9 5 8 0 0 0 0
```

empty

```
0 2 0 0 0 0 0 0 0
0 0 0 6 0 0 0 0 3
0 7 4 0 8 0 0 0 0
0 0 0 0 0 3 0 0 2
0 8 0 0 4 0 0 1 0
6 0 0 5 0 0 0 0 0
0 0 0 0 1 0 7 8 0
5 0 0 0 0 9 0 0 0
0 0 0 0 0 0 0 4 0
```

full

```
0 2 0 8 1 0 9 7
7 0 1 6 9 0 2 0 8
9 8 4 5 2 7 6 3 1
4 2 5 1 3 8 9 7 6
3 7 6 9 5 2 1 8 4
8 1 9 7 4 6 3 0 0
0 0 3 8 7 0 0 0 0
2 0 8 0 1 9 7 6 0
0 0 7 2 6 0 8 0 0
```

Table bellow compares the behaviour of the 4 algorithms for these sudokus in terms of time.

	Easy	Hard	Very hard	Empty	Full

Backtrack	0.009s	0.028s	0.048s	1.444s	0.008s
Logic	0.007s	Cant be solved	Cant be solved	Cant be solved	Cant be solved
Logic backtrack	0.070s	0.016s	0.047s	1.430s	0.008s
Heuristic	0.014s	0.050s	0.428s	39.850s	0.006s

Evaluation

The table above clearly shows that backtracking, logic backtrack and heuristic algorithm work for all type of sudokus. Logic backtrack is a bit more efficient than backtrack. It can also be seen that logic algorithm is not correct and effective at all and works only for easy Sudoku's.

By comparing backtrack and heuristic backtrack, we conclude that the prediction of efficiency of heuristic backtracking was not correct. Although it is true to say that this algorithm fills the Sudoku with fewer recursions, I believe that the inconsistency may come from the fact that I used bubble sort to sort the positions and their empty cells and bubble sort is not efficient at all and takes a long time.