Implementation of Sudoku Solver using Backtracking

For this project you will develop a Sudoku solver, a program that receives a Sudoku instance and solves it. Your program will use a backtracking approach as follows. At any given point, the solver has a partial assignment of values to some of the cells in the input grid. Initially, the partial assignment only has values for the preset cells that are specified in the input instance.

The solver repeatedly attempts to extend the current partial assignment, by selecting one of the unassigned cells and giving it a value that does not violate the Sudoku rules. If no good value exists for the selected cell, then this cell is unassigned and the solver backtracks to change the value assigned earlier to a previously chosen cell. If the assignment is extended so that all the cells are assigned a value consistent with the Sudoku rules, the input instance is solved.

Here is some recursive pseudocode for the crucial method.

```
// Returns true if the assignment has been extended to a complete
// assignment; if this is not possible, returns false.
// The incoming assignment may be a partial assignment.

boolean solve ( Assignment assignment ) {
   if (isComplete(assignment)) // base case: solution found return true
   else {
      // cell is chosen in some systematic way
      cell = chooseNextCell(assignment);
     // compute all the reasonable values that cell might take
     values = set values that do not appear is cell's row or column or box
     while (values is not empty) {
        // chosen in some systematic way
        chosenValue = the next value in values
        // cell gets a reasonable value
        setValue (cell, assignment, chosenValue)

       // try to assign values to any cells that remain unassigned
       if (solve(assignment))
          return true
       else {
```

```
            // setting cell to the chosen value did not work
            unset (cell, assignment)
            remove chosenValue from values
        }} // while

        // We tried each reasonable value for cell, and none worked
        // (if one had worked, we would have returned already)
        // so head back
        return false;

    } //else
} // solve
```

Your programming project

You will develop a solver for the Sudoku problem using the backtracking approach. Your solver receives a Sudoku instance as input and computes a (any one) solution for it. If no solution exists, your program will report this.

The input will be read in as the previous project so you will use your parser and board that you already have written.

After your own testing convinces you that your solver is correct, run and time your solver on all of the required input set. Preserve the runtimes since you will need them again to compare with future projects.

Submission instructions

By the deadline, submit your working solver, evidence of its correctness on the required input set, and a meaningful table with its runtimes.

This material is based upon work supported by the National Science Foundation under Grant No. 1140753.