CMSC 421 Programming Assignment 1

Due Date: Part A: In class Tuesday, September 27
Part B: 6PM Friday, September 30

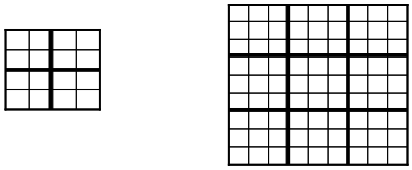
For this problem, you will apply search to solve *Su Doku* puzzles.

Su Doku

In this puzzle, you start with an initial partially filled in $n \times n$ game board. A simple 4×4 example is given below:

3			2
	1	4	
1	2		4
		2	1

At each step in the game, you can fill in the empty positions with a number from 1 to n. A solution to the puzzle is a fully filled in game board in which each number from 1 to n appears exactly once in each row, column and block. The blocks for a 4×4 board and a 9×9 board are shown in bold as below:



A solution to the simple 4×4 example is given below:

3	4	1	2
2	1	4	3
1	2	3	4
4	3	2	1

The difficulty of these puzzles varies greatly depending on the initial board. In some cases, a relatively straight-forward depth first search may work satisfactorily. In other cases, backtracking will be required.

Your Assignment

Su Doku can be solved in a number of ways – some options include using uninformed search, heuristic search, local search and constraint satisafaction.

The first part of your assignment, Part A, involves writing pseudo-code for a basic backtracking search algorithm to solve this constraint satisfaction problem for a 4x4 board. Your solution

should be about a page long, and should include details about how you will encode and maintain the current assignment as you search and possibly backtrack. You should do this part of the assignment as soon as possible, before you start coding!

The second part, Part B, of your assignment involves writing a C/C++ program to solve this puzzle. We have provided a simple shell for a program which reads in and writes out a Su Doku game board. Your job is to fill in the innards. You may use whatever search algorithm you like, but beware that naive search algorithms may not be able to solve the harder puzzles.

Code Infrastructure

Code and infrastructure for the course is available from `/afs/csic.umd.edu/class/cmsc421/0101/common` For this assignment, a simple shell of a C program which reads in and outputs a board is provided. Copy the source code from `/afs/csic.umd.edu/class/cmsc421/0101/common/pa1/src` and use the provided Makefile with `make` to compile the program.

Sample input and output files are available in `/afs/csic.umd.edu/class/cmsc421/0101/common/pa1/input` `/afs/csic.umd.edu/class/cmsc421/0101/common/pa1/output`. Your code will be tested on other inputs, so it is probably a good idea to try some others. You should check for incorrect input.

Evaluation

1. [**25 points**] The psuedo-code for the two different algorithms.
2. [**50 points**] Your Su Doku program
3. [**25 points**] A 1 page write-up describing your Su Doku code development.

Deliverables

To receive full credit, you must 1) Turn in **Part A** your psuedo-code writeup in class on Tuesday, September 26. 2) You must turn in your program as explained in below 3) Submit a 1 page writeup with your program, describing the search approach you used, any other approaches you tried, and successes or failures you had along the way. 2) and 3) comprise **Part B** and are both due at 6PM on Friday, September 30.

Submission Guidelines for Part B

Please remember that your code must work on the class linuxlab system. Please double-check this before turning in your submission.

To submit electronically, execute the following command:
`/afs/csic.umd.edu/submit/fall2005/cmsc421/bin/submit 1 project1.tar`
project1.tar is a tar file explained below. `project1.tar` should be in your current directory.

Note that you will see a message saying that your submission is successful after submitting your tar file. In case of multiple submissions for the same assignment, only the last submission will be saved. The previous ones are overwritten. (Note: use electronic submission only for programming assignments. Homework assignments should be submitted in class or physical copies should be given to the TA or Professor.)

- You should include all the files you need to submit in one tar file called `project1.tar`. This is the file that you will submit using the above submit command. Please do not zip the tar file. Your tar file should include following:

  1. Source code: ONLY source files for your Su Doku program. Do not submit object files, executables, and input files.
  2. Makefile: This is your makefile that builds your source code on typing *make* to create an executable called *sudoku* .

     The TA will run your executables with an input file supplied on the commandline, i.e. `sudoku inputfile`. This means that your program should take the input file name as a parameter using (argc and argv) as in the sample.
  3. A report file: A page write-up describing your Su Doku code development. See the above Evaluation 3. (.txt(a plain-txt file), .ps, or .pdf will be accepted.)
  4. README: This should contain
     - Your name.
     - Your login id.
     - The list of files that you are submitting as a tar file.
     - Anything special that you want the TA to keep in mind when grading your project. PLEASE DO NOT EXPLAIN YOUR ALGORITHM HERE. YOU SHOULD DO THAT IN YOUR REPORT FILE.

Extra Credit

[20 points] Particularly fast, well-engineered solutions or thorough comparisons of alternate methods will receive up to 20 points extra credit.