CSI 5165 Combinatorial Algorithms Winter 2018
Computer Science University of Ottawa

Homework Assignment #2 (100 points, 15%)

Due: Tuesday March 20, 11:59PM

Guidelines for programming parts: Write your program in some high level programming language such as C, C++, Java. Hand in pseudocode, program and output results (note if too many tests are done, submit only a sample of output results and summarize results in tables). Please, specify the platform you run your tests on (machine speed, machine RAM and operating system).

1. (25 points) **Backtracking for self avoiding walks** (written question)
   A self-avoiding walk is described by a sequence of edges in the Euclidean plane, beginning at the origin, such that each of the edges is a horizontal or vertical segment of length 1, and such that no point in the plane is visited more than once. There are precisely 4 such walks of length 1, 12 walks of length 2, and 36 walks of length 3. Define choice sets and describe a backtracking algorithm for the problem of finding all self-avoiding walks of length n.

2. (25 points) **Estimating backtracking tree size** (written question)
   Write an algorithm in pseudocode that uses the method of estimating the size of a backtrack tree described in Section 4.4 of the textbook (Knuth's method), in order to estimate the total number of cliques of a given graph. The input for your algorithm consists of a graph G and the number P of probes, and the output is the estimated number of cliques of the graph based on P probes.

3. (50 points) **SUDOKU by backtracking**

   SUDOKU is a placement puzzle in which symbols from 1 to 9 are placed in cells of a 9×9 grid made up of nine 3×3 subgrids, called regions. The grid is partially filed with some symbols (the "givens"). The grid must be completed so that each row, column and region contains exactly one instance of each symbol.

Example1: easy for humans

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

Example 2: medium for humans

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

Example 3: hard for humans

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

In this exercise you will write two backtracking algorithms for solving **SUDOKU**:

(a) The first algorithm will try to fill out the first available table position in order (say left to right, top to bottom).

(b) The second algorithm will try to fill out the table position that has the smallest number of values allowed.

Efficiency and clarity count!

For each of the algorithms:

- Write a **pseudocode** for a backtracking algorithm that solves **SUDOKU**.
- Implement your algorithm and test the instances given in the course web site (report any input errors):

  `http://www.site.uottawa.ca/~lucia/courses/5165-18/a2data/`
  `http://www.site.uottawa.ca/~lucia/courses/5165-18/a2data.zip`

  The input for your program consists of a 9×9 matrix representing the SUDOKU puzzle, where empty spaces in the grid are entered as 0s.

  The output of your programs should consists of:

  - the input grid;
  - the solution grid;
  - statistics on the algorithm performance such as: total number backtracking nodes and running time (CPU time for the solution, not including I/O), etc.
- Compare the results of both algorithms by displaying a table with the statistics for each algorithm on the same input values. Discuss the results.