CSI 5165 Combinatorial Algorithms — Winter 2021
Computer Science — University of Ottawa

Homework Assignment #2 (100 points, weight 15%)
Due: March 4, 2021 (11:55PM)

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, print 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. (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 given instances (33 test cases provided).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.

2. (50 points) **Backtracking program for non-linear codes.**

   If $x, y \in \{0,1\}^n$, then recall that $\textsc{Dist}(x,y)$ denotes the Hamming distance between x and y, that is the number of components i where $x_i \neq y_i$. A non-linear code of length n and minimum distance d is a subset $\mathcal{C} \subseteq \{0,1\}^n$ such that $\textsc{Dist}(x,y) \geq d$ for all $x, y \in \mathcal{C}$. Denote by $A(n,d)$ the maximum number of n-tuples in a length-n non-linear code of minimum distance d.

   Example: The following vectors give a nonlinear code of length $n = 6$ minimum distance 4 with 4 codewords: $\mathcal{C} = \{111000, 100110, 010101, 0010011\}$. You can inspect that any two codewords in $\mathcal{C}$ have distance at least $d = 4$. Note that in this example the distance between any pair of codewords is exactly equal to 4, but in general some of these distances could be larger (5, 6, etc).

   (a) (15 points) Describe a backtracking algorithm that given n and d compute $A(n,d)$ (give pseudocode and any other pertinent explanation).

   (b) (25 points) Implement your algorithm and compute $A(n,4)$ for $4 \leq n \leq 8$. You can find values for $A(n,d)$ for small values of n, d in following web page:

   `http://www.win.tue.nl/~aeb/codes/binary-1.html`

   For each of your tests, report the input values, the final answer (both $A(n,4)$ and the actual code obtained), the number of backtracking nodes visited and CPU time. Efficiency and clarity count.

   (c) (10 points) Show a pseudocode and give a program implementation for Knuth's method to estimate the size of the backtracking tree for your algorithm. Use this method to estimate the size of the backtracking tree for $4 \leq n \leq 11$. For each value of n, choose a suitably large number P of probes and show the estimate for at least 5 values of number of probes equally spaced within $[10, P]$.
   Does this estimate approximates well the number of nodes you found in the previous question? (If not, you may have to check correctness of the computations there or your estimation here).

   (d) (Bonus 10 points) Values for $A(n,4)$ are also known for $n \geq 9$. For example, it is known that $A(9,4) = 20$ and $A(10,4) = 40$, but these values are harder to obtain. If your algorithm can determine some of these higher values of n you can get a bonus up to 10 points.
   IMPORTANT: Never spend time working on a bonus question, if the other parts of the assignment have not been completed. There is no guarantee that this bonus question is attainable within a reasonable time.

   HINTS:
   1) In case it helps, it is possible to rephrase this problem of finding maximum non-linear codes as a maximum clique problem in a suitable graph.
   2) You are allowed to make your algorithm more efficient by fixing elements of your code, if this is possible to be done without loss of generality. If you do so, please explain and justify.