CSI 5165 Combinatorial Algorithms — Fall 2014
Computer Science — University of Ottawa

Homework Assignment #2 (100 points, weight 15%)
Due: November 4 at 11:30 p.m. (submit to blackboard)

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) **Backtracking for reduced Latin squares**
   (This is based on Exercise 4.13 of the textbook.)
   A *Latin square* of order n is an n by n array A, whose entries are chosen from an n-element set $\mathcal{Y}$, such that each symbol in $\mathcal{Y}$ occurs in exactly one cell in each row and in each column of A. A Latin square on the n-element set $\mathcal{Y} = \{1, 2, \ldots, n\}$ is said to be a *reduced Latin square* if the elements in the first row and in the first column occur in the natural order $1, 2, \ldots, n$. The reduced Latin squares of order 4 are as follows:

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

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

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

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

   (a) (25 points) Write a backtracking algorithm to determine the number l_n of reduced Latin squares of order n. Show pseudocode and program. Efficiency and effectiveness counts, and these types of discounts will be applied in part (a), even though their effects will be noticed in later parts.

   (b) (5 points) Run your program for $n = 2, 3, 4, 5, 6$, reporting l_n, as well as the number of backtrack tree nodes visited and the total running time (excluding the time for I/O!).

   (c) (10 points) Based on the backtracking algorithm above, using Knuth's method for estimating the size of a backtracking tree (see Section 4.4 of textbook), design an algorithm to estimate the size of your backtracking tree. Show pseudocode and program. Using your program above, estimate the number of nodes in your backtracking tree for $2 \leq n \leq 8$. For each n, show your estimate for different sample sizes, and report your results for $2 \leq n \leq 8$, showing how the estimation compares to the exact values obtained for the range $2 \leq n \leq 6$. Using the exact and estimated data obtained, can you estimate the running time for inputs of size $n = 7, 8$? Justify your reasoning.

   (d) (10 points) Use Knuth's generalized method (which allows for weights on the nodes of the backtracking tree), to estimate the number of reduced Latin squares of order n, for $n = 4, \ldots, 7$. For more details see page 113 of the textbook by Kaski and OIstergard. The idea is to put weight 1 on nodes that correspond to a reduced Latin square and weight zero on all the other nodes. Use a large enough number of repetitions of the estimator algorithm to get good estimate for $n = 7$; show your results for $n = 7$ and 5 different values of repetitions.

   The number of reduced Latin squares of order n is known up to $n = 11$ and given in the following table:[1]

n	1	2	3	4	5	6	7	8
l_n	1	1	1	4	56	9408	16942080	535281401856

n	9	10	11
l_n	377597570964258816	7580721483160132811489280	5363937773277371298119673540771840

[1] Double check numbers in: McKay and Wanless, On the number of Latin squares, *Ann. Combin.* 9, 2005, 335-344.

2. (50 points) **Backtracking program for nonlinear codes.**

If $x, y \in \{0,1\}^n$, then recall that $\textsc{Dist}(x,y)$ denotes the Hamming distance between x and y. 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.

(a) (35 points) Describe a backtracking algorithm to compute $A(n,d)$ (give pseudocode and any other pertinent explanation).
Implement your algorithm and compute $A(n,4)$ for $4 \leq n \leq 8$. The correct values for $A(n,d)$ for small values of n and d can be found on the 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, the number of backtracking nodes visited and CPU time. Efficiency and clarity count.

(b) (5 points) Show a pseudocode and give a program 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).

(c) (10 points) Show a pseudocode and give a program for Knuth's generalized method to estimate the size of $A(n,4)$ for $n = 4, \ldots, 16$. The exact values can be found on the web site given in part a.