CSI 5165 Combinatorial Algorithms — Fall 2009
Computer Science — University of Ottawa

Homework Assignment #2 (100 points, weight 15%)
Due: Tuesday November 3, at 11:30 a.m. (in lecture)

Backtracking

Backtracking for reduced Latin squares

1. (55 points) (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) (15) 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.

   (d) (10) 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.

   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.

Travelling Salesman Problem (TSP)

2. (10 points) (Exercise 4.2 of the textbook)
   Find a formula for the number of nodes in the state space tree that results when Algorithm 4.10 is run on an instance of the Travelling Salesman problem having n vertices.

3. (35 points) **Improving textbooks TSP algorithms**
   (Based on Exercise 4.8 of the textbook.) Read the textbook exercise 4.8 regarding additional pruning needed so that the TSP algorithms do no consider each Hamiltonian cycle twice. In addition, use the following extra specifications:

   - Consider all 5 given algorithm variations for your basis of comparison, namely: Algorithm 4.10, Algorithm 4.13 with B1 and B2, Algorithm 4.23 with B1 and B2, where B1 is MinCostBound and B2 is ReduceBound.
   - To run the given algorithms, use C code for the textbook available under TSP.tar.gz at: `http://www.math.mtu.edu/~kreher/cages/Src.html`
   - Create 3 new variations (one for each) of the given 3 algorithms, so that each algorithm considers each hamiltonian cycle only once. Make sure you apply the pruning closer to the top of the backtracking tree, so that the number of visited backtracking tree nodes is considerably reduced.
   - Run your comparison on the graph for Exercise 4.7 (we will call this graph G_1), using the 5 original algorithms and the 5 enhanced algorithms.

   This exercise consists of the following parts:

   (a) (15) Give explanation and pseudocode that incorporates the pruning of equivalent hamiltonian cycles. Your methods will be judged based on effectiveness of pruning the backtracking tree size; pruning high on the tree is likely to be more effective; discounts related to effectiveness will be applied here, even though it will be detected in later parts (be sure to revise your method if it is not effective in pruning the tree).

   (b) (15) Provide your program highlighting with a marker the modifications to existing code and where new code has been added.

   (c) (5) Show the output of the textbook code algorithms for graph G_1. Run your new algorithms for graph G_1, providing the same output information that is being provided by the original code. Results and discussion: Show a table of results listing stats for each new algorithm next to the ones for the corresponding original algorithm. For each algorithm provide the number of nodes in the backtracking tree. Briefly discuss the results obtained.