CSI 7160 Adv. Topics in TCS: Combinatorial Algorithms and Structures — Fall 2002
Computer Science — University of Ottawa

Homework Assignment #1 (100 points, weight 15%)
Due: Friday, October 11, at 10:00 a.m. (in lecture)

---

1. (20 points) **Revolving door ranking.**
Prove that the ranking formula for the revolving door ordering given in the textbook pages 48-52 holds. More precisely, let $A^{n,k}$ be the sequence recursively defined in page 48.
Prove that:

$$\text{rank}(T) = \sum_{i=1}^{k}(-1)^{k-i}\left(\binom{t_i}{i}-1\right) = \begin{cases} \sum_{i=1}^{k}(-1)^{k-i}\binom{t_i}{i}, & \text{if } k \text{ is even} \\ \left(\sum_{i=1}^{k}(-1)^{k-i}\binom{t_i}{i}\right)-1, & \text{if } k \text{ is odd} \end{cases}$$

Hint: Prove the first equation by induction. Use the first equation to prove the second one.

---

2. (10 points) **Gray codes and revolving door ordering.** (Textbook exercise 2.9)
Suppose $1 \leq k \leq n$, and we delete all vectors in the binary reflected Gray code G^n that do not correspond to subsets of cardinality k. Prove that the vectors that remain comprise the vectors in the revolving door ordering $A^{n,k}$.

---

3. (30 points) **A different subset ordering.** (Textbook exercise 2.10)
Another way to order the subsets of an n-element set is to order them first in increasing size, and then in lexicographic order for each fixed size. For example, when $n = 3$, this ordering for the subsets of $S = \{1, 2, 3\}$ is:

$$\emptyset, \{1\}, \{2\}, \{3\}, \{1,2\}, \{1,3\}, \{2,3\}, \{1,2,3\}.$$

Develop ranking, unranking and successor algorithms for the subsets with respect to this ordering. Please, hand-in the pseudocode, as well as a program with the algorithm's implementation. Include a main program that does a complete test for the all the procedures when applied to all possible instances with $n = 4$ as well as a printout of the test results.

---

4. (30 points) **Backtracking for generating all reduced Latin squares** (Textbook exercise 4.14)
A Latin square of order n is an n by n array A, whose entries are chosen from $\{1, 2, \ldots, n\}$, such that each symbol of $\{1, 2, \ldots, n\}$ occurs in exactly one cell in each row and in each column of A.

A Latin square of order 4: $A =$

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

A Latin square is said to be a *reduced* Latin square if the elements in the first row and the first column appear in their natural order $1, 2, \ldots, n$. (The Latin square shown above is reduced.)
Write a backtracking algorithm to determine the number of reduced Latin squares of order n. Please, hand in the pseudocode, as well as a program implementing your algorithm. Run your program for $n = 2, 3, 4$ and 5, and show printouts for your results. Please, make your program also print statistics on the number of (recursive) calls to your backtrack algorithm, for each input. Efficiency and clarity count.

---

5. (10 points) **Simple practice with algorithms for generation of elementary objects**

Calculate the result for the following operations. Show your work.

- subsets:
  Give the Successor and the Rank of 01010110 in the Gray code G^8.

- k-subsets:
  Give Rank of $\{3, 6, 7, 9\}$ considered as a 4-subset of $\{0, 1, \ldots, 12\}$ in lexicographic and revolving-door order.
  What is the Successor in each of these orders?

- Permutations:
  Find the rank and successor of the permutation $[2, 4, 6, 7, 5, 3, 1]$ in lexicographic and Trotter-Johnson order.

  Unrank the rank $r = 56$ as a permutation of $\{1, 2, 3, 4, 5\}$, using the lexicographic and Trotter-Johnson order.

You may use one of various high-level programming languages, such as C, C++, Pascal, Java. Please, consult with me if you would rather use another programming language.

Assignments are supposed to be done strictly individually and with no search through related literature. If you use any other resource in addition to your brain, the textbook or class notes, please add appropriate references.