CSI 5165 Combinatorial Algorithms | Fall 2013
Computer Science | University of Ottawa

**Homework Assignment #1** (100 points, weight 15%)
Due: Friday October 11, at 16:00 p.m.
(slide under office door STE5027, or handin on a lecture two days earlier)

---

## Generating elementary combinatorial objects

1. (33 points) Another way to order the subsets of an $n$-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 **unranking**, **ranking** and **successor** algorithms for the subsets with respect to this ordering.

   Hint: Adapt the ideas developed for the lexicographical order of $k$-subsets of an $n$-set to this situation. Note that efficiency will play a role in the evaluation.

2. (33 points) 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 a minimal change ordering for the $k$-element subset of an $n$-set.

   Suggested approach:

   (a) Prove the following Lemma: Let $1 \leq k \leq n$. Then,
   - the first vector of $G^n$ corresponding to a set of cardinality $k$ is of the form [0...01...1], with $(n - k)$ 0's followed by $k$ 1's.
   - the last vector of $G^n$ corresponding to a set of cardinality $k$ is of the form [10...01...1] with a 1 followed by $(n - k)$ 0's, followed by $(k - 1)$ 1's.

   (b) Prove the main result by induction on $n$, using the Lemma.

3. (34 points) A *derangement* is a permutation $[\pi[1], \pi[2], \ldots, \pi[n]]$ of the set $\{1, 2, \ldots, n\}$ such that $\pi[i] \neq i$, for all $1 \leq i \leq n$. Let $D_n$ denote the number of derangements of an $n$-element set. Note that $D_1 = 0$ and $D_2 = 1$. To show that $D_n = (n-1)(D_{n-1}+D_{n-2})$, for $n \geq 3$, we can use the following argument:

   We can set $\pi[1]$ in $n - 1$ ways, namely with $i = 2, 3, \ldots n$.
   Once $\pi[1] = i$ there are two possibilities:

   - $\pi[i] = 1$, in which case we list all derangements of $\{1, \ldots, n\} \setminus \{1, i\}$ (there are $D_{n-2}$ of them) in order to complete the current derangement.

- $\pi[i] \neq 1$, in which case we can rename value 1 as $i$ list all derangements of $\{1, 2, \ldots, n\} \setminus \{1\}$ (there are $D_{n-1}$ of them), and then change back $i$ to 1 in each of these derangements.

Use this recurrence relation (and its associate argument) to develop an algorithm to generate all the derangements. Note that you do not need to necessarily come up with a successor algorithm; indeed a recursive algorithm might be the easiest solution. Ideally, you would not store several derangements in main memory at the same time, that is, after a derangement has been generated it can be printed out; this would keep your memory requirements in $O(n)$ rather than exponential. You may have to keep some $n$-arrays in your program in order to deal with current permutations, indexes that are active and possible relabelings. Note that efficiency will play a role in the evaluation.

(a) Provide a **pseudocode** of your algorithm (with similar level of detail as the algorithms given in textbook). Please, also add any comments or extra **explanations** necessary to understand why your pseudocode works.

(b) **Implement** our algorithm, providing a **printout** of the code, as well as **outputs** for $n = 3, 4, 5$