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

Homework Assignment #1 (100 points, weight 15%)
Due: Exercise 0 due February 3 (as Quiz 1); Exercises 1-4 due February 11, 11:59PM, via Bightspace.

Generating elementary combinatorial objects

0. (10 points) **Simple practice with generation algorithms (please submit Quiz1 in brightspace)** Calculate the result for the various operations. Show your work by attaching a pdf with all your calculations at the last question of the quiz.

    - Subsets: successor and rank for Gray codes.
    - k-subsets: rank and successor in both lexicographic and revolving-door order
    - Permutations: rank, successor and unranking in both lexicographic and Trotter-Johnson order.

1. (20 points) **Generalized Lexicographical n-tuples for mixed basis**
When we generated all subsets of an n-set in lexicographical order of their characteristics vectors, our algorithms were really generating binary n-tuples in lexicographical ordering. We noticed that the successor algorithm was equivalent to adding 1 to an n-bit binary number; the ranking algorithm was nothing more than transforming a given n-tuple (considered as a binary representation of a number) to its numerical representation; and the unranking algorithm was equivalent to transforming a number into an n-tuple corresponding to its binary representation.

    In this exercise, you will develop similar algorithms, but we are not going to be looking at binary n-tuples, but a tuple in some "mixed-basis". Consider fixed $m_0, m_1, \ldots, m_{n-1}$, where $m_i \geq 1$, for all $1 \leq i \leq n$. We will be generating in lexicographical order all n-tuples $(a_{n-1}, \ldots, a_1, a_0)$ where each component a_i satisfies $0 \leq a_i < m_i$, for all $1 \leq i \leq n$. The case of binary n-tuples (subsets of an n-set) is a special case of this where $m_i = 2$ for all $1 \leq i \leq n$.

    For example, for $(m_2, m_1, m_0) = (3, 4, 2)$ the 24 tuples in lexicographic order are: 000, 001, 010, 011, 020, 021, 030, 031, 100, 101, 110, 111, 120, 121, 130, 131, 200, 201, 210, 211, 220, 221, 230, 231.

    Give successor, ranking and unranking algorithms (pseudocode is fine); let's name these algorithms tupleSuccessor, tupleRanking and tupleUnranking, respectively.

2. (20 points) **Correctness of** Successor **algorithm for Graycodes G^n**
Prove Theorem 2.2 of the textbook which states that Algorithm GrayCodeSucessor(n, T) (slide 15) correctly computes successor for the binary reflected Gray code. You need to state and prove several facts, which will help you prove that the procedure below correctly computes the sucessor of a binary tuple $A = (a_{n-1}, a_{n-2}, \ldots, a_1, a_0)$ with weight $w(A)$ in G^n:

    > If $w(A)$ is even, then the last bit of A (namely a_0) is flipped; if $w(A)$ is odd, then we find the first "1" from the right, and flip the next bit (to the left). The last vector in G^n, which has no successor, is $[1, 0, \ldots, 0]$.

    Hint: Use the the recursive definition of G^n given in the textbook and in slide 13. Prove the facts using induction on n.

3. (25 points) **Different order for subsets of an n-set**
Consider an ordering of subsets of an n-set that lists the subsets in non-increasing cardinality by listing sets of cardinality $k = n, n-1, \ldots, 1, 0$ according to the Revolving Door (minimum change) Ordering of k-subsets of an n-set denoted by $A^{n,k}$. In our notation, we can define the tuple C^n of subsets according to this order by concatenating the tuples of subsets given by $A^{n,k}$, $k = n, n-1, \ldots, 1$ as follows:

$$C^n = A^{n,n} || A^{n,n-1} || \cdots || A^{n,1} || A^{n,0}.$$

For example, for $n = 5$ the sets are given in the following order: (each line correspond to $A^{5,k}$, $k = 5, 4, 3, 2, 1, 0$)

$$\begin{gathered}
C^n = [\{1,2,3,4,5\}, \\
\{1,2,3,4\},\ \{1,2,4,5\}, \{2,3,4,5\},\ \{1,3,4,5\},\ \{1,2,3,5\}, \\
\{1,2,3\},\ \{1,3,4\},\ \{2,3,4\}, \{1,2,4\}, \{1,4,5\},\ \{2,4,5\},\ \{3,4,5\},\ \{1,3,5\},\ \{2,3,5\},\ \{1,2,5\}, \\
\{1,2\},\ \{2,3\}, \{1,3\}, \{3,4\},\ \{2,4\},\ \{1,4\},\ \{4,5\},\ \{3,5\},\ \{2,5\},\ \{1,5\}, \\
\{1\},\ \{2\},\ \{3\},\ \{4\},\ \{5\}, \\
\{\}]
\end{gathered}$$

Give successor, ranking and unranking algorithms (pseudocode is fine).
Notes: As seen in class, we represent a set S with $|S| = k$ by a tuple $T = [t_1, t_2, \ldots, t_k]$, where $S = \{t_1, t_2, \ldots, t_k\}$ and $t_1 < t_2 < \ldots < t_k$. For a tuple T you can refer to its length by T.LENGTH; for example if T$= [1, 2, 5]$, T.LENGTH=3. In the example, RANK$(T = [1, 2, 3, 4, 5], n = 5) = 0$, RANK$(T = [\,], n = 5) = 31$, UNRANK$(r = 2, n = 5) = [1, 2, 4, 5]$, SUCCESSOR$(T = [1, 2, 5], n = 5) = [1, 2]$.
Hint: You should study and use the algorithms for $A^{n,k}$.

4. (25 points) **Signed permutations**
A signed permutation is a permutation with optional signs attached to the elements; therefore, there are $2^n n!$ such signed permutations of n elements. For example $31\bar{2}$ is a signed permutation.

   (a) Give an algorithm that generates all signed permutations of $\{1, 2, \ldots, n\}$ where each step either interchanges two adjacent elements or negates the first element.

   (b) Demonstrate that your algorithm works by implementing it, and showing the results for n=1,2,3,4. Please do a compact printout with $2^{(n-1)}(n-1)!$ lines with $2n$ permutations per line.

This resembles the Trotter-Johnson minimal change ordering for permutations.

Example for $n = 1$: $[1, \bar{1}]$, and for $n = 2$ and $n = 3$:

$$\begin{array}{l}
[12, 21, \bar{2}1, 1\bar{2}, \\
\ \bar{1}\bar{2}, \bar{2}\bar{1}, 2\bar{1}, \bar{1}2\ ],
\end{array}
\qquad\qquad
\begin{array}{l}
[123, 132, 312, \bar{3}12, 1\bar{3}2, 12\bar{3}, \\
\ 21\bar{3}, 2\bar{3}1, \bar{3}21, 321, 231, 213, \\
\ \bar{2}13, \bar{2}31, 3\bar{2}1, \bar{3}\bar{2}1, \bar{2}\bar{3}1, \bar{2}1\bar{3}, \\
\ 1\bar{2}\bar{3}, 1\bar{3}\bar{2}, \bar{3}1\bar{2}, 31\bar{2}, 13\bar{2}, 1\bar{2}3, \\
\ \bar{1}\bar{2}3, \bar{1}3\bar{2}, 3\bar{1}\bar{2}, \bar{3}\bar{1}\bar{2}, \bar{1}\bar{3}\bar{2}, \bar{1}\bar{2}\bar{3}, \\
\ \bar{2}\bar{1}\bar{3}, \bar{2}\bar{3}\bar{1}, \bar{3}\bar{2}\bar{1}, 3\bar{2}\bar{1}, \bar{2}3\bar{1}, \bar{2}\bar{1}3, \\
\ 2\bar{1}3, 23\bar{1}, 32\bar{1}, \bar{3}2\bar{1}, 2\bar{3}\bar{1}, 2\bar{1}\bar{3}, \\
\ \bar{1}2\bar{3}, \bar{1}\bar{3}2, \bar{3}\bar{1}2, 3\bar{1}2, \bar{1}32, \bar{1}23]
\end{array}$$

Note that for n you have $2^n n!$ signed permutations. When we obtain the ones for $(n+1)$ each of the signed permutations for n will show up $2(n+1)$ times as a nested permutation. For $n+1$ steps the element $n+1$ (with or without sign) moves from the end position until the first position, then its sign gets reversed and it moves from the first position to the last.
Note: Efficiency matters; make sure your memory usage is in $O(n)$.
Hint: It will be useful to study well how the Trotter-Johnson algorithm works for minimal change of regular permutations as it will be helpful here.