Enumeration School **September 28, 2011**
Fundamentals & Basic Algorithms: Exercises **Yoshio Okamoto**

Legend: (∗) Recommended; (−) Easy; (+) Hard
Notice 1: You need to prove the correctness and the efficiency of algorithms.
Notice 2: For some of the problems, the problem poser doesn't have complete solutions

Exercise 1 Consider the subset enumeration algorithm by binary partition given in the lecture, and modify the algorithm so that it outputs the objects with difference output. Prove that this modification yields an amortized constant-time delay algorithm.

Exercise 2 Prove Proposition 2 from the lecture (a rule to determine the next object in a Gray code for subset enumeration).

Exercise 3 Prove Proposition 4 from the lecture (that the number of edges on the path from one output to the next output passed by prepostorder traversal is constant).

Exercise 4 (∗−) Design an amortized polynomial-time delay algorithm to output all subsets of $\{1,\ldots,n\}$ with even cardinality, when given a natural number n as input. Here "polynomial" means a polynomial in n. If possible, design an amortized linear-time delay algorithm or a worst-case linear-time delay algorithm.

Exercise 5 Design an amortized polynomial-time delay algorithm to output all subsets of $\{1,\ldots,n\}$ that has cardinality of multiples of q, when given two natural numbers n and q, $q \leq n$, as input. Here "polynomial" means a polynomial in n. If possible, design an amortized linear-time delay algorithm or a worst-case linear-time delay algorithm.

Exercise 6 (∗) Design an amortized polynomial-time delay algorithm to output all subsets of $\{1,\ldots,n\}$ that sum up to s, when given two natural numbers n and s as input. Here "polynomial" means a polynomial in n and $\log_2 s$. If possible, design an amortized linear-time delay algorithm or a worst-case linear-time algorithm. Note: for example, the sum of the elements of $\{3,7,8\}$ is $3+7+8=18$.

Exercise 7 Design an amortized polynomial-time delay algorithm to output all subsets of $\{1,\ldots,n\}$ in the lexicographic order, when given a natural number n as input. Here "polynomial" means a polynomial in n. If possible, design an amortized linear-time delay algorithm or a worst-case linear-time algorithm. Note: For two subsets $X, Y \subseteq \{1,\ldots,n\}$, we say that X is *lexicographically smaller* than Y if $\min X \setminus Y < \min Y \setminus X$. For convenience, we set $\min \emptyset = -\infty$.

Exercise 8 (∗) A permutation $(a_1,\ldots,a_n)$ of the set $\{1,\ldots,n\}$ is a *derangement* if $a_i \neq i$ for all $i \in \{1,\ldots,n\}$. Design an amortized polynomial-time delay algorithm to output all derangements of $\{1,\ldots,n\}$, when given a natural number n. Here "polynomial" means a polynomial in n. If possible, design an amortized linear-time delay algorithm or a worst-case linear-time algorithm.

Exercise 9 A permutation $(a_1,\ldots,a_n)$ of the set $\{1,\ldots,n\}$ is *alternating* if it satisfies $a_1 > a_2 < a_3 > a_4 < \cdots$. Design an amortized polynomial-time delay algorithm to output all alternating permutations of $\{1,\ldots,n\}$, when given a natural number n. Here "polynomial" means a polynomial in n. If possible, design an amortized linear-time delay algorithm or a worst-case linear-time algorithm.

Exercise 10 (∗) For natural numbers $r_1,\ldots,r_m$ and $c_1,\ldots,c_n$, a 2-*way contingency table* with row sums $(r_1,\ldots,r_m)$ and column sums $(c_1,\ldots,c_n)$ is an $n \times m$ matrix $A = (a_{ij})$ with natural number entries such that $\sum_{j=1}^{n} a_{ij} = r_i$ for all $i \in \{1,\ldots,m\}$ and $\sum_{i=1}^{m} a_{ij} = c_j$ for all $j \in \{1,\ldots,n\}$. Design an amortized polynomial-time delay algorithm to output all 2-way contingency tables with row sums $(r_1,\ldots,r_m)$ and column sums $(c_1,\ldots,c_n)$, when given $r_1,\ldots,r_m$ and $c_1,\ldots,c_n$ as input. Here "polynomial" means a polynomial in $\sum_i \log_2 r_i + \sum_j \log_2 c_j$. If possible, design an amortized linear-time delay algorithm or a worst-case linear-time algorithm. Hint: As a first step, consider how to find one contingency table. If you find a simple

construction, then you may try to construct an enumeration tree with this simple construction as a root, and apply reverse search.

Exercise 11 Design an amortized polynomial-time delay algorithm to output all subsets of X that sum up to even numbers, when given a finite set $X \subseteq \{1, \ldots, n\}$ as input. Here "polynomial" means a polynomial in the size of X and $\log_2 n$. If possible, design an amortized linear-time delay algorithm or a worst-case linear-time algorithm.

Exercise 12 A *increasing subsequence* of a permutation $\pi = (a_1, \ldots, a_n)$ of the set $\{1, \ldots, n\}$ is a subsequence $(a_{i_1}, a_{i_2}, \ldots, a_{i_k})$ of π such that $a_{i_1} < a_{i_2} < \cdots < a_{i_k}$. Design an amortized polynomial-time delay algorithm to output all increasing subsequences of π, when given a permutation π of the set $\{1, \ldots, n\}$ as input. Here "polynomial" means a polynomial in n. If possible, design an amortized linear-time delay algorithm or a worst-case linear-time algorithm. Note: By defintion, an empty sequence and a sequence of length one are also increasing subsequences.

Exercise 13 (You may need a familiarity with graph theory and graph algorithms.)
A *clique* in an undirected graph $G = (V, E)$ is a vertex subset $S \subseteq V$ of G such that every pair of two vertices in S are joined by an edge. Design an amortized polynomial-time delay algorithm to output all cliques in a given undirected graph. If possible, design an amortized linear-time delay algorithm or a worst-case linear-time algorithm. Note: By definition, an empty set and a subset of size one are also cliques.

Exercise 14 (+) (You may need a familiarity with graph theory and graph algorithms.)
A *spanning forest* in an undirected graph $G = (V, E)$ is a subgraph of G that is maximal with respect to the property of containing no cycle. Design an amortized polynomial-time delay algorithm to output all spanning forests in a given undirected graph. If possible, design an amortized linear-time delay algorithm or a worst-case linear-time algorithm.

Exercise 15 (+) (You may need a familiarity with discrete geometry and computational geometry.)
For a finite point set $P \subseteq \mathbb{R}^2$ on the 2-dimensional plane, a subset $S \subseteq P$ is *linearly separable* if there exists a line $\ell \subseteq \mathbb{R}^2$ such that all points in S lie on one side of ℓ and all points in $P \setminus S$ lie on the other side of ℓ. Design an amortized polynomial-time delay algorithm to output all linearly separable subsets of a given point set P, assuming that no three points in P lie on a single line. If possible, design an amortized linear-time delay algorithm or a worst-case linear-time algorithm.

Exercise 16 (+) (You may need a familiarity with discrete geometry and computational geometry.)
For a finite point set $P \subseteq \mathbb{R}^2$ on the 2-dimensional plane, a subset $S \subseteq P$ lies in *convex position* if there exists a convex polygon with S as its vertex set. Design an amortized polynomial-time delay algorithm to output all subsets of a given point set P that lie in convex position, assuming that no three points in P lie on a single line. If possible, design an amortized linear-time delay algorithm or a worst-case linear-time algorithm. Note: By definition, the empty set and a subset of size one are also in convex position.