Topics in Algorithms

Exercises

1 Generation of Basic Combinatorial Objects

1.1 Generation of Subsets

1. Consider the sum:

$$\sum_{(\varepsilon_1,\dots,\varepsilon_n)\in\{0,1\}^n} f(\varepsilon_1,\dots,\varepsilon_n)$$

(a) Show that it may be calculated in $O(n)$ time in each of the following cases:

(i) $f(\varepsilon_1,\dots,\varepsilon_n) = c_1\varepsilon_1+\ldots+c_n\varepsilon_n$ for some constants $c_1,\dots,c_n$.

(ii) $f(\varepsilon_1,\dots,\varepsilon_n) = a^{c_1\varepsilon_1+\ldots+c_n\varepsilon_n}$ for some constants $a, c_1,\dots,c_n$.

(iii) $f(\varepsilon_1,\dots,\varepsilon_n) = \cos(c_1\varepsilon_1+\ldots+c_n\varepsilon_n)$ for some constants $c_1,\dots,c_n$. (Hint: $\cos\theta = (e^{i\theta}+e^{-i\theta})/2$.)

(b) How fast can you calculate the sum above if $f(\varepsilon_1,\dots,\varepsilon_n) = P(c_1\varepsilon_1+\ldots+c_n\varepsilon_n)$ for some polynomial P and constants $c_1,\dots,c_n$?

2. Put:

$$M_n = \frac{1}{2^n}\sum_{(\varepsilon_1,\dots,\varepsilon_n)\in\{0,1\}^n}\sqrt{\sum_{i=1}^n \varepsilon_i}\,.$$

(a) Write $M_n = E(h(X))$, where X is a random variable whose distribution belongs to some well-known family of distributions and $h:\mathbf{R}\longrightarrow\mathbf{R}$ is a suitable function.

(b) Use the inequality $E^2(Y)\le E(Y^2)$, which holds for any random variable Y, to deduce that $M_n\le\sqrt{n/2}$.

(c) Employ the subadditivity of the mapping $x \to \sqrt{x}$ (i.e., the property $\sqrt{a+b} \leq \sqrt{a} + \sqrt{b}$ for $a, b \geq 0$) to obtain the same conclusion.

(d) Show that $M_n = \sqrt{\frac{n}{2}} \cdot (1 - o(1))$. (Hint: Use Chebyshev's inequality to show that for "most" n-tuples $(\varepsilon_1, \ldots, \varepsilon_n)$ we have $\sum_{i=1}^{n} \varepsilon_i > \frac{n}{2} - n^{2/3}$.)

3. We need to go over a certain family of subsets of $\{a_1, \ldots, a_n\}$. The following algorithm has been suggested: Go over all subsets using the binary expansion approach, and for each of them test whether it belongs to the required family or not. Find the time complexity of the algorithm, and suggest improvements to the algorithm if possible, if the family consists of all

(a) subsets of even size;

(b) subsets of size $\lfloor n/2 \rfloor$;

(c) subsets not containing adjacent elements (i.e., if a_i belongs to the subset for some i, then a_{i+1} does not).

4. In our construction of a Gray code we may add the coordinates in any way, which gives in principle $n!$ Gray codes. Are the paths obtained in this way all different from each other?

5. Consider the Tower of Hanoi Problem. Let M_n be the sequence of length $2^n - 1$ recording which disk moves at each step of the process if there are n disks. Prove that $M_n = T_n$, where T_n is the sequence, introduced in class, of bits changing when going over all elements of $\{0,1\}^n$ by the Gray code.

6. Consider the set

$$B = \{0, 1, \ldots, d_1 - 1\} \times \{0, 1, \ldots, d_2 - 1\} \times \ldots \times \{0, 1, \ldots, d_k - 1\},$$

where $d_1, d_2, \ldots, d_k \geq 2$ are integers. Two elements $(x_1, \ldots, x_k)$ and $(y_1, \ldots, y_k)$ in B are *adjacent* if they are at a distance of 1 apart, i.e., for some $1 \leq l \leq k$ we have $|y_l - x_l| = 1$ and $y_i = x_i$ for every $i \neq l$. A *Gray code* for B is a sequence of elements of B, containing a unique occurrence of each element of B, in which consecutive entries are adjacent.

(a) Show that, for any k-tuple $(d_1, \ldots, d_k)$, the set B admits a Gray code.

(b) Characterize those k-tuples for which B admits a Gray code with the additional property that the last element of the sequence is adjacent to the first.

7. The solution presented in class to the problem of selecting a random subset of an n-element set (or, equivalently, a sequence of length n over $\{0, 1\}$) involves n selections of random numbers. The following algorithm, which requires a single selection of a random number, has been suggested: Select a random number $r \in [0, 1)$, multiply it by 2^n and take the integer part $s = \lfloor 2^n r \rfloor$. The bits of s form a random sequence as required.

(a) Is the algorithm theoretically correct?

(b) What do you expect the algorithm to yield in practice for large n (say, $n = 100$)?

1.2 Generation of Permutations

8. Let P be an arbitrary fixed subset of the set of all permutations of $\{1, 2, \ldots, n\}$. We want to design an algorithm which, given a permutation σ, returns the smallest permutation (according to the lexicographic order) in P which is greater or equal to σ (and returns the smallest permutation in P if σ is greater than all elements of P).

(a) The following algorithm has been suggested: Start with $\pi = \sigma$. While $\pi \notin P$, replace π by its successor. Analyze this algorithm in the average case and the worst case if P is the set

(i) D_n of all derangements (permutations σ with $\sigma(i) \neq i$ for each i);

(ii) ND_n of all non-derangements.

(b) Suggest worst-case polynomial time algorithms for the problem for both D_n and ND_n. Analyze their performance.

9. A permutation $\sigma = (\sigma_1, \sigma_2, \ldots, \sigma_n)$ is *cup-shaped* if $\sigma_1 > \sigma_2 > \ldots > \sigma_k < \sigma_{k+1} < \ldots < \sigma_n$ for some $1 \leq k \leq n$. Design an algorithm which goes over all cup-shaped permutations in linear time (in the number of such permutations).

10. In the algorithm for traversing the set of all permutations with minimal changes, at each stage exactly two elements change their locations. How many elements on the average change their locations at each stage in the algorithm which traverses the permutations in lexicographic order? (Figure out how this average behaves as the number of elements grows to infinity rather than trying to obtain an exact formula for each n.)

11. Which permutation is the one to be encountered last when we traverse all permutations with minimal changes?

12. The three algorithms below have been suggested for selecting a random permutation of $1, 2, \ldots, n$. For each of them, determine whether it is correct (i.e., chooses each permutation with the same probability $1/n!$) and, if so, find the average number of selections of random integers required to obtain a random permutation. How does this average behave as $n \to \infty$?

(a) Choose n random integers between 1 and n until the chosen n-tuple forms a permutation.

(b) We need to select $\sigma_1, \sigma_2, \ldots, \sigma_n$. The numbers are selected one by one. At the k-th stage, $k = 1, 2, \ldots, n$, select a random integer between 1 and n repeatedly until it is distinct from all those selected before, and set σ_k as this integer.

(c) Start with the permutation $(1, 2, \ldots, n)$. Repeatedly select two random integers i and j between 1 and n, and swap σ_i and σ_j. Repeat this procedure l times, where l is sufficiently large.

1.3 Generation of Subsets of a Fixed Size

13. Suppose we want to go over the set of all subsets of size k of $\{1, 2, \ldots, n\}$ with minimal changes, where the notion of a minimal change is restricted to include only interchanges of consecutive numbers, i.e., where a number s is removed from the subset and $s+1$ is added or vice versa.

(a) Show that, if n is odd and $\binom{n}{k}$ is even, then it is impossible to go over the set with minimal changes.

(b) Show that there exist infinitely many pairs (n, k) for which it is possible to go over the set with minimal changes.

14. Consider the set of all subsets of size either k or $k+1$ of a set of size n. A *minimal change* of a subset consists of either removing an element from the subset or adjoining an element to it.

(a) Prove that, if $n \geq 3$ and $n \neq 2k+1$, it is impossible to go over the set with minimal changes.

(b) Show that, for $(n, k) = (1, 0), (2, 0), (3, 1), (5, 2)$, it is indeed possible to go over the set with minimal changes.

(c) Let $n = 2k+1 \geq 3$. Suppose we go over all subsets of the given set using the Gray code, shown in class. Now omit all subsets of sizes other than k and $k+1$. Show that the remaining sequence does not yield a traversal of our set with minimal changes.

(d) Prove or disprove: If $n = 2k+1$, then it is possible to go over the set with minimal changes.

15. An algorithm for selecting a random subset of any size k of $\{1, 2, \ldots, n\}$ is provided. Use it for designing an algorithm of similar performance for finding a random subset of size k of $\{1, 2, \ldots, n\}$, containing no two adjacent integers.

16. A *legal expression* in parentheses is a word over the alphabet $\{(,)\}$ in which the total number of parentheses of the two types is equal and in every prefix of which the number of right parentheses does not exceed that of left parentheses.

(a) For any positive integer n, denote by a_n the number of legal expressions in parentheses of length $2n$. Prove that

$$a_n = a_0 a_{n-1} + a_1 a_{n-2} + a_2 a_{n-3} + \ldots + a_{n-1} a_0, \qquad n = 1, 2, \ldots .$$

(b) Employing generating functions, conclude from part (a) that $a_n = \binom{2n}{n}/(n+1)$ for each n.

(c) Note that a legal expression in parentheses of length $2n$ is uniquely determined by the set (of size n) of locations of the left parentheses. Thus, given any algorithm for traversing the set of all subsets of size k of a set of size n, we may use it (with $2n$ and n instead of n and k, respectively) to traverse the set of legal expressions in parentheses of length $2n$ (by going over all expressions with n left and n right parentheses and omitting the illegal ones). Suppose the given algorithm for traversing all subsets of size k is linear. Analyze the suggested algorithm for traversing all legal expressions in parentheses.

(d) Develop a linear time algorithm for traversing all legal expressions in parentheses of length $2n$ in lexicographic order.

(e) Using parts (a) and (b), develop an algorithm of linear expected time for selecting a random legal expression in parentheses of length $2n$.

17. A subset of $\{1, 2, \ldots, n\}$ is *sparse* if it contains no two adjacent numbers.

(a) Suppose a linear time algorithm for traversing the set of all subsets of $\{1, 2, \ldots, n\}$ of size k (for every n and k) is given.

Consider the following algorithm for traversing the set of all sparse subsets of $\{1, 2, \ldots, n\}$ of size k: Go over all subsets of size k and omit those which are not sparse. Is the suggested algorithm linear? If yes – prove it, if not – explain why not and suggest a linear time algorithm.

(b) Suppose a linear time algorithm for traversing the set of all subsets of size k with minimal changes is given. Use it to develop a linear time algorithm for traversing the set of all sparse subsets of size k with minimal changes.

(c) Develop an algorithm for selecting a random sparse subset of size k which works in time $O(k)$.

1.4 Generation of Partitions

18. Let $P(n, k)$ be the set of partitions of n whose maximal component is k.

(a) Modify the algorithm presented in class, for traversing the set of all partitions of n in lexicographic order, to a linear time algorithm for traversing $P(n, k)$ in lexicographic order.

(b) Consider the algorithm presented in class for traversing the set of all partitions of n in vocabulary order. Explain why it basically solves also the problem of traversing $P(n, k)$ in vocabulary order.

19. Denote by $p(n)$ the number of partitions of n (where we agree that $p(0) = 1$) and by $p(n, k)$ the number of those partitions of n whose maximal component is k.

(a) Denote by f the generating function of the double sequence $(p(n, k))_{n,k=0}^{\infty}$, namely

$$f(x, y) = \sum_{n=0}^{\infty} \sum_{k=0}^{\infty} p(n, k) x^n y^k.$$

Prove that f satisfies the functional equation

$$f(x, xy) = (1 - xy) f(x, y).$$

(b) Let g be the generating function of the sequence $(p(n))_{n=0}^{\infty}$. Express g in terms of f.

(c) Show that $p(n + 1) > p(n)$ for $n \geq 1$.

(d) Show that $p(n+1,k) \geq p(n,k)$ for each n and k.

(e) For each fixed k, find a polynomial Q such that $p(n,k) = \Theta(Q(n))$.

(f) Prove that $p(n,k) \leq \binom{n}{k}$ for each n and k.

(g) Use the preceding part to obtain an upper bound on $p(n)$.

(h) Obtain the upper bound on $p(n)$, obtained in the preceding part, directly (i.e., without using $p(n,k)$).

(i) Prove that $p(n) \geq 2^{C\sqrt{n}}$ for every $n \geq 1$ for an appropriate constant $C > 0$. (Hint: Restrict yourself to partitions using only some of the possible integers, and such that each component, except perhaps for 1, appears at most once.)

1.5 Generation of Set Partitions

20. For integers $n \geq k \geq 1$, denote by $\mathcal{P}(n)$ the set of all partitions of the set $\{1, 2, \ldots, n\}$ and by $\mathcal{P}(n,k)$ the set of all partitions into k components. Put $a_n = |\mathcal{P}(n)|$ and $a_{n,k} = |\mathcal{P}(n,k)|$.

(a) Prove that for every fixed k we have $a_{n,k} = \Theta(k^n)$.

(b) Conclude from the previous part that the generating function of the sequence (a_n) converges only at the point 0.

(c) Prove that for every fixed k we have $a_{n,n-k} = \Theta(Q(n))$ for an appropriate polynomial Q.

(d) Find an explicit formula for $a_{n,2}$.

(e) Same for $a_{n,3}$.

(f) Same for $a_{n,n-1}$.

(g) Same for $a_{n,n-2}$.

(h) Prove that $n^{an} \leq a_n \leq n^{bn}$ for suitable constants $b > a > 0$ for all sufficiently large n.

(i) Consider the order on $\mathcal{P}(n)$ according to which the algorithm given in class produces the partitions. Design an algorithm which, given a partition in $\mathcal{P}(n,k)$, produces the next partition belonging to $\mathcal{P}(n,k)$ (or reports that the given partition is the last in $\mathcal{P}(n,k)$). The algorithm should work in time $O(n)$, with the implicit constant independent of k.

1.6 Generation of Random Variates from Discrete Distributions

21. Following are several distributions, defined in terms of probability functions, supported on $\{1, 2, \ldots, n\}$. In each case, p_k denotes the probability of k, and the constant c is determined so that $\sum_{k=1}^{n} p_k = 1$. We use the following method for choosing random variates from the distribution. A random number $x \in [0, 1)$ is selected, and is compared with the number p_1, then with $p_1 + p_2$, then with $p_1 + p_2 + p_3$, and so forth, until for the first time the accumulated sum of p_i's exceeds x. If $\sum_{k=1}^{r-1} p_k \leq x < \sum_{k=1}^{r} p_k$, then we select the number r. Determine the order of magnitude of the expected time required by the algorithm for selecting a single random variate. (Ignore any calculations performed once. We refer only to those which need to be done each time we select another variate.)

(a) $p_k = c/k$.

(b) $p_k = c/k^2$.

(c) $p_k = c/k^3$.

(d) $p_k = c/\log(k+1)$.

22. For each of the following distributions, defined similarly to those of the preceding exercise, design a linear expected time algorithm for selecting random variates.

(a) $p_k = ck^2$.

(b) $p_k = c/k(k+1)$.

(c) $p_k = ck2^{-k}$.

(d) $p_k = cF_k$, where $(F_k)_{k=0}^{\infty}$ is the Fibonacci sequence.

(e) $p_k = c(1 + \sin k)$.

(f) $p_k = c\cos^2 k$.