

## 2 Algoritmos probabilistas

1. Let  $A[1 \dots n]$  be an array with  $n$  different integers. Let  $ran(n)$  a randomized function that outputs an integer  $i$ ,  $1 \leq i \leq n$  under the uniform distribution. Let us consider the following algorithms

**Input:**  $A[1 \dots n]$   
**from**  $i = 1$  **to**  $n$  **do**  
    swap values  $A[i]$  and  $A[ran(n)]$

**Input:**  $A[1 \dots n]$   
**from**  $i = n$  **downto**  $1$  **do**  
    swap values  $A[i]$  and  $A[ran(i)]$

Do these algorithms output a uniform random permutation? Justify your answer.

2. Consider the set  $S = \{1, \dots, n\}$ .

- (a) We generate  $X \subseteq S$  as follows: A fair coin is flipped independently for each element of  $S$ , if the coin lands H, the element is added to  $X$ , otherwise it is not. Prove that the resulting set  $X$  is equally likely to be any one of the  $2^n$  possible subsets.
- (b) Suppose  $X, Y \subseteq S$  are chosen independently and u.a.r. from all  $2^n$  subsets from  $S$ . Compute  $\Pr[X \subseteq Y]$  and  $\Pr[X \cup Y = S]$

3. Consider the following algorithm to generate an integer  $r \in \{1, \dots, n\}$ : We have  $n$  coins labelled  $m_1, \dots, m_n$ , where the probability that  $m_i = \text{head}$  is  $1/i$ . Toss the in order the coins  $m_n, m_{n-1}, \dots$  until getting the first head, if the first head appears with coin  $m_i$ , the  $r = i$ . Prove that the previous algorithm yield an integer  $r$  with uniform distribution. i.e. the probability of getting any integer  $r$  is  $1/n$ .)

4. The following approach is called **reservoir sampling**. Suppose we have a sequence of items passing by one at a time. We want to maintain a sample of one item with the property that it is uniformly distributed over all the items that we have already seen at each time step. Moreover, we want to accomplish this without knowing the total number of items in advance or storing all of the items that we see.

Consider the following algorithm, which stores just one item in memory at all times. When the first item appears, it is stored in the memory. When the  $k$ -th item appears, it replaces the item in memory with probability  $1/k$ .

Explain if and why this algorithm solves the problem.

5. We have a function  $F : \{0, \dots, n-1\} \rightarrow \{0, \dots, m-1\}$ . We know that, for  $0 \leq x, y \leq n-1$ ,  $F((x+y) \bmod n) = (F(x) + F(y)) \bmod m$ . The only way we have for evaluating  $F$  is to use a lookup table that stores the values of  $F$ . Unfortunately, an Evil Adversary has changed the value of  $1/5$  of the table entries when we were not looking.

Describe a simple randomised algorithm that, given an input  $z$ , outputs a value that equals  $F(z)$  with probability at least  $1/2$ . Your algorithm should work for every value of  $z$ , regardless of what values the Adversary changed. Your algorithm should use as few lookups and as little computation as possible.

6. Donat un graf  $G = (V, A)$ , el volem acolorir amb 3 colors  $c : V \rightarrow \{0, 1, 2\}$ . Diem que una aresta  $(u, v) \in A$  esta acolorida correctament si  $c(u) \neq c(v)$ . Sigui  $c^*$  el nombre de arestes acolorides correctament a una 3-acoloració de  $G$  que maximitza el nombre d'arestes acolorides correctament. Dissenyeu un algorisme aleatori que utilitze 3 colors per acolorir  $G$  i tal que el nombre esperat d'arestes acolorides correctament sigui  $\geq \frac{2}{3}c^*$ .

7. In cryptographic applications we often need to generate random integer  $r \in \{1, \dots, n\}$  together with the factorization of  $r$ . Note the obvious method of just generating  $r$  u.a.r. and then factoring  $r$ , is not efficient for large  $r$  as factoring is not known to be in P. Consider the following function:

```

Prime-fact( $n$ )
  %generate an integer sequence  $n \geq s_1 \geq s_2 \geq \dots \geq s_t = 1$ 
   $s_1 = n; i = 1$ 
  while  $s_i \neq 1$  do
    ++ $i$ 
    choose  $s_i$  u.a.r. from  $\{1, \dots, s_{i-1}\}$  using the method in 2.3
  select the set  $S = \{s_i | s_i \text{ is prime}\}$ 
  do  $r = \prod_{s_i \in S} s_i$ 
  if  $r \leq n$  return  $r$  with prob =  $r/n$ 
  FAIL

```

Notice you can choose repeated integers, i.e. it may happen that  $s_i = s_{i-1}$  (in step 3).

To test for primality we can either use the deterministic algorithm AKS (Agrawal–Kayal–Saxena) ( $O(\lg n)^6$ ) or the faster probabilistic algorithms from Miller-Rabin a Monte-Carlo, one side error that runs in ( $O(\ln n)^4$ ).

- Suppose we represent the sequence  $(s_i)$  obtained by the algorithm as an  $n$ -dim vector  $(m_1, \dots, m_n)$ , where  $m_j$  is the number of times the number  $j$  occurs in the sequence (for ex. if  $n = 10$  and the sequence is  $s_1 = 8, s_2 = 5, s_3 = 5, s_4 = 1$  the vector would be  $(1, 0, 0, 0, 2, 0, 0, 1, 0, 0)$ , and  $S = \{5, 5\}$ , i.e.  $r = 25$ , and the algorithm would FAIL.) Show that the probability of generating the sequence is given by  $\prod_{j=2}^n (\frac{1}{j})^{m_j} (1 - \frac{1}{j})$ . (*Hint: the entire generating process can be thought of as a tossing of a sequence of coins*)
- Show (from the previous item) that the algorithm outputs each  $r \in \{1, \dots, n\}$  with equal probability  $\alpha_n/n$  where  $\alpha_n = \prod_p (1 - 1/p)$ , and the product is over all primes  $p \leq n$ .
- A standard theorem from number theory says that  $\alpha_n^{-1} \sim 1.8 \ln n$ . Suppose we repeat the algorithm until it outputs a  $r$ . What is the expected number of repetitions needed?
- The running time of each trial of the algorithm is dominated by the primality test. Show that the expected number of primality test is bounded by  $H_n = \Theta(\ln n)$ .
- Show that the expected number of primality tests performed before an output  $r$  is produced is  $O(\lg^2 n)$ . Justify why it is not enough to multiply the expectations obtained in items (c) and (d).

8. Considereu el següent algorisme per a trobar el mínim, d'una seqüència  $A$  de caràcters que pertanyen a un conjunt totalment ordenat

**Min**( $A$ )

Ordena  $A$  d'acord amb una permutació aleatòria  $\text{min} := A[1]$

**De**  $j = 1$  **fins a**  $n$  **fer**

**si**  $\text{min} > A[j]$  **fer**

$\text{min} := A[j]$

**tornar**  $\text{min}$ .

Evidentment aquest algorisme té complexitat  $T(n) = n$ . Sigui  $X$  una variable aleatòria que compta el nombre de cops que s'executa la instrucció 4 (la nova assignació a  $\text{min}$  dintre del bucle). Quin serà el valor de  $\mathbf{E}[X]$ ?

9. Sigui  $\mathcal{P}$  un programa on els dissenyadors diuen que calcula la inversa d'una matriu, i.e. donada com entrada una matriu  $A$  de dimensions  $n \times n$ ,  $\mathcal{P}$  produeix una matriu  $B$  que hauria de complir  $B = A^{-1}$ . Dissenyeu un algorisme que comprove que  $\mathcal{P}$  fa el que ha de fer. (En altres paraules, dissenyeu un algorisme que comprove si  $B = A^{-1}$ . Quina és la probabilitat d'error?. Quina és la complexitat del vostre algorisme?)

10. Alice (A) maintain a large  $n$ -bit database  $X = \{x_{n-1}, \dots, x_0\}$  of information, while Bob (B) maintains a second copy  $Y = \{x_{n-1}, \dots, x_0\}$  of the database. Periodically, when  $X$  and  $Y$  are separately updated, they must check their copies for consistency, i.e. to assure both  $A$  and  $B$  are identical. Because  $A$  and  $B$  are far away, they would like to discover the presence of an inconsistency without transmitting the entire database between them, which is costly and it is prone to noise. Alice and Bob use the following fingerprinting technique:

- Interpret the data as  $n$ -bit integers  $N_X = \sum_{i=0}^{n-1} x_i 2^i$  and  $N_Y = \sum_{i=0}^{n-1} x_i 2^i$ .
- Choosing  $p$  u.a.r. among the primes  $\leq 2^n$ , A computes  $\phi(N_X) = N_X \bmod p$  and sends the result to B.
- B computes a fingerprint  $\phi(N_Y) = N_Y \bmod p$  and compares with the quantity he got from Alice.
- Then, if  $X \neq Y$ , we should get  $\phi(N_X) \neq \phi(N_Y)$ , but it is possible  $\phi(N_X) = \phi(N_Y)$  and  $X \neq Y$ .

To choose  $p$  uniformly at random: The prime number theorem says that if for an integer  $k$ ,  $\pi(k)$  is the number of primes  $< k$ , then  $\pi(k) \sim k / \ln k$ . Another algebraic theorem it may be useful is: The number of prime divisors of any integer  $m < 2^n$  is at most  $n$  (Any prime is  $\geq 2$ , Therefore, the number of distinct primes that divide  $m \leq 2^n$  is no greater than  $n$  because if we multiply together more than  $n$  numbers that are  $\geq 2$ , then we get a number greater than  $2^n$ .)

- (a) Prove that  $\Pr[\phi_p(N_X) = \phi_p(N_Y) | X \neq Y]$  is very small.
- (b) How many bits are needed for A to send  $\phi_p(N_X)$  to B?

11. Let  $h(x) = x \bmod m$  be a hash function defined as  $h(x) = x \bmod m$ , where  $m = 2^p - 1$ . Let  $w$  be a character string corresponding to the representation in radix  $2^p$  of a natural number. Prove that, if  $w'$  is a string obtained by permuting the symbols in  $w$ ,  $h(w') = h(w)$ .

12. There are different operations that we wish to implement on sets of integers. Take into account that in a set there are no repeated elements. Let  $a[n]$  and  $b[n]$  be arrays holding the elements in two sets  $A$  and  $B$  with  $n$  elements. Provide an exact and a randomized algorithm for each of the following problems.

(a) Given  $a$  and  $x \in \mathbb{N}$ , are there integers  $y, z \in A$  such that  $x = y + z$ ?

(b) Given  $a$  and  $b$ , is  $A = B$ ?

Your algorithms should solve the problems in time  $\Theta(n \log n)$  and in expected time  $O(n)$ , respectively.

13. Alice and Bob have a recorded they favorite songs. To access quickly the collection each of them have created a multiple Bloom filter using  $mk$  bits and  $k$  hash functions. Assume that  $X$  is Alice's Bloom filter and  $Y$  is Bob's filter. Using the Bloom filters, provide a way to estimate how far away their musical tastes are as a function of  $m, n, k$  and the number of common songs.

14. Let  $A$  and  $B$  be sets of  $n$  elements from the same universe  $\mathcal{U}$ . Let  $h$  be a hash function  $h : \mathcal{U} \rightarrow \{0, \dots, m - 1\}$ . Consider Bloom filters  $T_1 = h(A)$  and  $T_2 = h(B)$  obtained after hashing their elements into a  $m$ -bit array. Each one of the arrays can be view as an  $m$ -dimensional Boolean vector, therefore the it make sens considering Boolean operations. Define  $T_1 \vee T_2$  as the vector holding  $T_1[i] \vee T_2[i]$ , for  $0 \leq i \leq m - 1$ . In the same way we can define  $T_1 \wedge T_2$ . Is there a data set  $S$  for which  $T_1 \vee T_2$  ( $T_1 \wedge T_2$ ) are  $h(S)$ ?