Radomized Algorithms Problemes 1 Fall 2018.

1.- Suppose 3 coins are tossed. Each coin has an equal probability of head or tail, but are not independent.

(a) What are the minimum and maximum values of the probability of three heads?

(b) Now assume that all pairs of coins are mutually independent. What are the minimum and maximum values of the probability of three heads?

2.- Consider the following balls-and-bin game. We start with one black ball and one white ball in a bin. We repeatedly do the following: choose one ball from the bin uniformly at random, and then put the ball back in the bin with another ball of the same colour. We repeat until there are n balls in the bin. Show that the number of white balls is equally likely to be any number between 1 and $n-1$.

3.- (*) Consider the set $S = \{1, \ldots, n\}$.

(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. Proof 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 of S. Compute $\mathbf{Pr}[X \subseteq Y]$ and $\mathbf{Pr}[X \cup Y = S]$

4.- Suppose you choose an integer uniformly at random from the range $[1, 1000000]$. Using the inclusion-exclusion principle, determine the probability that the number chosen is divisible by one or more of $4, 6$, and 9.

5.- (*) Consider the following algorithm to generate an integer $r \in \{1, \ldots, n\}$: We have n coins labelled $m_1, \ldots, m_n$, where the probability that $m_i =$ head is $1/i$. Toss the in order the coins $m_n, m_{n-1}, \ldots$ until getting the first head, if the fist 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 $1/n$.

6.- We have a function $F : \{0, \ldots, n-1\} \to \{0, \ldots, m-1\}$. We know that, for $0 \leq x, y \leq n-1$, $F((x+y) \mod n) = (F(x) + F(y)) \mod m$. 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 were 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.

7.- (*) Consider a standard Poker deck with 52 cards, therefore without jokers, four suits, each one 13 cards, which ordered by descending value are: A, K, Q, J, 10, 9, 8, 7, 6, 5, 4, 3, 2 (the Ace A can be considered as 1 for making a straight, but it has maximal value. Each player gets 5 cards. Notice the number of possible sets of 5 cards is $\binom{52}{5} = 2598960$, so the probability of a player having a specific combination is 3.84823^{-7}.
In the figure below you have all important hands in Poker, ordered by decreasing value (notice we can not achieve 5 of a kind without having jokers).

(a) Prove that for each of the following hands, the probability of having the hand is the given one:
   - Having *four of a kind* (four cards of the same value, the other one does not matter). Prob. =0.000240096
   - Having *flush*: (five cards of the same suit, not consecutive values). Prob.= 0.00198079
   - Having *straight flush*: (five cards of the same suit with consecutive values) Prob.= 0.0000153908

(b) Suppose you are playing with 3 other players, and you have a *four of a kind*. What is the probability one of the other players has a better hand (i.e. a straight flush).

8.- Given a text $T = x_1 x_2, \cdots x_n$ (w.l.o.g in binary) and a pattern $S = s_1 \cdots s_m$, with $m << n$ and both chains over the same alphabet $\Sigma = \{0, 1\}$, we want to determine if S occurs as a contiguous substring of T. For ex., if $T = 101101100010101110$ and $S = 1011$ then 101$\boxed{1011}$0010$\boxed{1011}$10. The standard greedy takes $O(nm)$ steps. There are deterministic algorithms that work in worst-time $O(n + m)$ (Knuth-Morris-Pratt), but they are complicated, difficult to implement and with large implementation constants. The following simple probabilistic algorithm does the job using the fingerprint technique, with a small probability of error. The algorithm computes the fingerprint of S and compares with the fingerprints of successive sliding substring of T, i.e. with $T(j) = x_j \cdots x_{j+m-1}$, for $1 \leq j \leq n-m+1$.

Matching P, T
Express S as an integer $D(S) = \sum_{i=0}^{m-1} x_{i+1} 2^i$
  so $D(S)$ is a m-bit integer
Choose a prime $p \in [2, \ldots, k]$,
  where $k = cmn \ln(cmn)$, for suitable $c > 1$
Compute $\phi(S) = D(S) \bmod p$
for $j = 1$ to $n - m + 1$ **do**
  Compute $D(T(j)) = \sum_{i=0}^{m-1} x_{j+i} 2^i$
  Compute $\phi(T(j)) = D(T(j)) \bmod p$
  if $\phi(T(j)) = \phi(S)$ **then**
    output match at position j
  endif
endfor

Prove,

(a) This algorithm is one-side, it may output match when there is no match. Prove the $\mathbf{Pr}\left[\textit{output match, when no match}\right] \leq 1/c$, for suitable $c > 0$.

(b) Prove that the algorithm can be implemented in $O(n+m)$ steps.