Probability and Computing Hilary 2017

Exercise Sheet 1

Elias Koutsoupias (with thanks to Stefan Kiefer and Stanislav Živný)

1. Suppose that two sets X and Y are chosen independently and uniformly at random from all 2^n subsets of $N = \{1, ..., n\}$. Determine $\Pr(X \subseteq Y)$ and $\Pr(X \cup Y = N)$.

2. Consider the following variation of the coupon collector's problem. Each box of cereal contains one of $2n$ different coupons. The coupons are organised into n pairs: $\{1, 2\}$, $\{3, 4\}$, etc. To win the prize you must obtain at least one coupon from each pair. Assuming that the coupon in each cereal box is chosen uniformly at random from the $2n$ coupons, what is the expected number of boxes you have to buy before you can claim the prize?

3. We have a function $F : \mathbb{Z}_n \to \mathbb{Z}_m$. 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 to evaluate F is to use a look-up table that stores the values of F. Unfortunately an Evil Adversary has changed the value of $1/5$ of the table entries–but we don't know which ones.

   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 use as few lookups and as little computation as possible.

   Hint: The (deterministic) algorithm that looks up the value of $F(z)$ in the look-up table and returns that value is not a correct solution, because it returns the correct value with probability 0 if the Adversary has changed the table entry for z.

   Suppose I allow you to repeat your algorithm three times. What should you do in this case, and what is the probability that your enhanced algorithm returns the correct answer?

4. Consider the following program:

```
Input: integer array A[1..n]
z := infinity
for i = 1 to n
  do if A[i] < z then z:=A[i]
return z
```

   If the input array is chosen uniformly at random from all permutations of $\{1, \ldots, n\}$ find the expected number of times that variable z changes its value over a run of the algorithm.

   [Hint: use linearity of expectation.]

5. Suppose there is a stream of items passing by one at a time. We can either save or discard each item as it passes, but we cannot recover a discarded item and we only have room to store k items at any one time. We don't know beforehand the length of the stream. Describe a randomised procedure so that after the whole stream has passed the set of items held in memory is uniformly distributed among all k-element subsets of all items seen. Carefully justify the correctness of your proposed solution.

6. Consider the following procedure for deciding whether there exists a simple path of length k in an undirected graph $G = (V, E)$: (i) choose a total ordering $v_1 < v_2 < \ldots < v_n$ of the vertices V uniformly at random; (ii) turn G into a directed acyclic graph by orienting each edge in E from the lesser to the greater vertex with respect to the chosen ordering; (iii) compute a longest path in the resulting dag.

   (a) Show that one can decide whether a dag has a length-k path in polynomial time.

   (b) Show that the probability that a given length-k path in G yields a length-k path in the resulting randomly generated dag is at least $1/k^k$.

   (c) Conclude that the above procedure yields a polynomial-time randomised algorithm for computing a length-k path in case $k = O\left(\frac{\lg n}{\lg \lg n}\right)$.

7. A matching in a graph is a set of edges without common vertices. In the Maximum Bipartite Matching problem, we are given a bipartite graph $G(L \cup R, E)$ and we want to find a matching of maximum cardinality. Consider the following randomised algorithm for this problem: Each edge is selected independently with probability p. All edges that have common points are discarded.

   Assume that the bipartite graph has $|L| = |R| = n$ and that every vertex has degree 3.

   - What is the expected cardinality of the matching returned by the algorithm as a function of p?
   - Find the value of p that maximises the expected cardinality of the matching. What is the approximation ratio of this algorithm?
   - Show how to derandomise the algorithm.