Probability and Computing Hilary 2017

Exercise Sheet 2

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

1. Let A be a randomised algorithm and F a function such that A either returns $F(x)$ or `timeout` on any input x. Assume that there exists $p > 0$ such that for all inputs x, A returns $F(x)$ with probability at least p. Give an algorithm A' that almost surely (i.e., with probability 1) returns $F(x)$ on input x and whose expected running time is within a constant of the expected running time of A.

2. Let A be a randomised algorithm and F a function such that A returns $F(x)$ on any input x. Furthermore suppose that the expected running time of A is $O(n)$, where n denotes the input size. Note that we only know the expected running time; the actual running time may vary arbitrarily. For example, the running time may be exponential in n for some inputs. Give an algorithm that is guaranteed to terminate in time $O(n)$ for every input, and which on input x outputs $F(x)$ with probability at least 0.99 and otherwise returns `timeout`.

3. Suppose there are two integer multisets respectively stored in arrays $A[1..n]$ and $B[1..n]$. We want to determine whether the two sets are identical, i.e., each element has the same multiplicity in both A and B.

   (a) Describe a deterministic algorithm for testing equality of multisets with complexity $O(n \lg n)$.

   (b) Give a reduction of the multiset-equality problem to polynomial identity testing.

   (c) Over which field would you define and evaluate your polynomials in Part (b)?

4. Let $a_1, a_2, \ldots, a_n$ be a list of n distinct numbers. We say that a_i and a_j are *inverted* if $i < j$ but $a_i > a_j$. The **Bubblesort** algorithm works by swapping adjacent inverted numbers until there are no inverted numbers. Suppose that the input to **Bubblesort** is a permutation chosen uniformly at random from any of the $n!$ permutations of the n distinct numbers. Determine the expected number of swaps performed by **Bubblesort**.

5. Consider the following algorithm **RandomSelect** for finding the kth smallest element of an unsorted set S of size n:

   RandomSelect(S, k)

   Pick an element $p \in S$ at random

   By comparing p to each element of S, compute

   $S_1 := \{x \in S \mid x < p\}$

   $S_2 := \{x \in S \mid x > p\}$

   If $|S_1| = k - 1$ then output p

   If $|S_1| > k - 1$ then output **RandomSelect**(S_1, k)

If $|S_1| < k - 1$ then output **RandomSelect**$(S_2, k - |S_1| - 1)$

Let $T(n, k)$ denote the expected time (number of comparisons) required by **RandomSelect** to find the kth smallest element of a set of size n, and let $T(n) = \max_k T(n, k)$. Show that $T(n)$ is at most $4n$.

[**Hint**: Establish a recurrence for $T(n)$.]

6. The analysis of the algorithm for MAX-3-SAT showed that a random truth assignment satisfied a 7/8-fraction of the clauses in expectation. Using Markovs inequality, show that for $0 < \epsilon \leq 7/8$, repeating the randomized algorithm $t = O(1/\epsilon)$ times and taking the best of the t solutions satisfies at least $(7/8 - \epsilon)$-fraction of the clauses with probability at least $1/2$.