Heads and Tails, summer 2009

PROF. DR. JOACHIM VON ZUR GATHEN, DANIEL LOEBENBERGER

4. Exercise sheet
Hand in solutions until Sunday, 17 May 2009, 24:00h.

Exercise 4.1 (Complexity classes). (13 points)

In class there was a lot of confusion regarding some complexity classes. Your task is to fix that.

(i) Lookup the definitions for the complexity classes P, NP, PSPACE, RP, BPP and ZPP. $\boxed{6}$

(ii) Prove that BPP $\subseteq$ PSPACE. Hint: Enumerate all possible random coins of the algorithm. $\boxed{3}$

(iii) Verify the inclusions P $\subseteq$ RP $\subseteq$ NP $\subseteq$ PSPACE. $\boxed{3}$

(iv) Can you show that the set of primes is in RP? $\boxed{1}$

Exercise 4.2 (Repeated squaring). (8+4 points)

In class we discussed the Fermat test. For the test one needs to compute a power $a^k \pmod n$ for some parameters $a, k, n \in \mathbb{N}$ of roughly the same size. In this exercise we will explore how to compute this exponentiation efficiently.

(i) Assume you want to do the exponentiation like in school, i.e. you compute $a^k = a \cdot a^{k-1} \pmod n$ recursively until you hit $a^0 = 1$. Futher assume that each such step needs one nanosecond (10^{-9} seconds) on a standard computer. Estimate the time the computation would need if k is a 30bit, 60bit or 90bit integer, respectively. $\boxed{3}$

(ii) A much better approach, the so called *square and multiply algorithm*, can be described as follows: If k is even we compute $a^k = \left(a^{k/2}\right)^2 \pmod n$ recursively otherwise we compute $a^k = a \cdot a^{k-1} \pmod n$. Again the recursion is based on $a^0 = 1$. Do the same estimates as in (i). $\boxed{3}$

(iii) How many multiplications do you need to compute a^{382} using the square and multiply algorithm? $\boxed{2}$

(iv) How can you do better? $\boxed{+4}$

Exercise 4.3 (Shuffling cards). (4+5 points)

You are given a method that generates bits uniformly at random.

4 (i) Suggest an algorithm for generating a permutation of 2^k elements uniformly at random, where $k \in \mathbb{N}_{>0}$. Hint: Select successively elements while carefully checking that the element was not drawn before.

+5 (ii) Prove that every permutation is generated with the same probability and analyze the expected runtime of your algorithm. Hint: You may use the estimate $\sum_{j=1}^{n} \frac{1}{j} \approx \gamma + \ln n$ for some constant $\gamma \in \mathbb{R}_{>0}$.

Exercise 4.4 (Physical random generators). (4 points)

4 In class we discussed a random generator which was actually used in real life. Do some research on the internet, find some other ideas for (physical) random generators and describe how they work.

cosec computer security