TDA251/DIT280, Period 2, 2016: Algorithms Advanced Course.

Exam Problems.

Instructions

(i) **Motivate all your answers.**

(ii) An answer, even a correct one, without motivation *does* not count. On the other hand, answer concisely and to the point, without digressions.

(iii) Avoid wordy essays. Extensive writing does not necessarily add to clarity, but it can make it harder to retrieve the relevant statements and logical steps.

(iv) Submit your answers via the FIRE system as PDF attachment strictly before the given **deadline: Jan. 14, kl 20:00**.

(v) Make sure there is your *name* and *id* printed on every page and solution to every porblem starts on a *new* page.

(vi) Make a first submission as soon as possible. Later, you can upload a revised version by clicking on *withdraw* for the previous one in the FIRE. You can repeat it any number of times. Only the last one will be considered.

(vii) If you have any problems to stick to the deadline for an *important* reason, inform us in good time, in order to avoid failure.

(viii) You *must* do the exam completely on your own. Neither group work nor external help is permitted.

(ix) Used literature beyond the course material *must* be cited.

(x) Questions may be directed to *azams@chalmers.se* only. (However, no solution hints will be given. Only questions about the interpretation of the exam problems will be answered.)

(xi) Utmost academic *honesty* is expected. Cheating can lead to *failure* on the entire course and further consequences.

Problem 1

An influential French scholar in the 18th century proposed a funny randomized algorithm to approximate the number $\pi \approx 3.14$ without complicated calculations. It works as follows. Draw several parallel lines on the floor, such that any two consecutive lines have distance $2L$. Throw a stick of length L on the floor. Do this n times and count how often the stick hits a line. Let k denote the number of hits.

One can prove that the probability to hit a line is $1/\pi$. Hence n/k should be close to π. One may be wondering how efficient and accurate this method is. It turns out that one gets only a few decimals of π in reasonable time, however these are correct with high confidence. In detail:

(a) It should be pretty obvious that, for any fixed number n, the expected value of k/n is $1/\pi$. Explain why this is true.

(b) Now let us fix k instead, and repeat the experiment until k hits are observed. Let n be the (random) number of trials needed. Show that the expected value of n/k is π. – But be careful: This does not follow from (a) by using $E[1/X] = 1/E[X]$ for some random variable X. In fact, this equation is, in general, plainly wrong! What is the correct reason for the claimed expected value π? Note the slightly different assumptions in (a) and (b).

(c) Despite (b), let us work with the inverse value and approximate $1/\pi$. Let $d \geq 0$ be a fixed integer. Give an upper bound for the probability that $k/n > 1/\pi$, and fewer than d decimals of k/n are correct. (In other words, at most the first $d-1$ digits after the decimal point are equal to those of $1/pi$). Use the Chernoff bound $e^{-\delta^2\mu/3}$, for relative error δ and expectation μ, as provided in the lecture notes. (Here we do not consider the other case $k/n < 1/\pi$ which is similar.)

(d) Say why it was correct to use the Chernoff bound in (c).

(e) Suppose that you want to keep a fixed error probability but get one further decimal ($d+1$ rather than d). How much do you have to increase n? Explain.

(f) Suppose that you are satisfied with d decimals but you want to make the error probability smaller. How does the error probability behave as a function of n?

Remark: These are a lot of questions, but each one can be answered in one or two lines.

Problem 2

An important and fundamental problem in *streaming* is the following. Suppose the stream consists of m elements each of which is an integer between 1 and n. Here we assume that

n is known but the stream can be arbitrarily long. We would like to estimate the number of distinct numbers in the stream. For instance if the stream is 1, 1, 10, 2, 2, 2, 1, 1, 10 the answer should be 3. Of course we can do this by maintaining n counters but this would require a huge amount of space. Here we outline a simple idea for this problem. Let the stream of numbers be $a_1, a_2, \cdots, a_m$. We want to estimate d, the number of distinct numbers in the stream. Use a random hash function $h : 1, 2, ..., n \to [0, 1]$ and keep track of $Z = \min\{h(a_1), h(a_2), ..., h(a_m)\}$ which is *only one number to store.*

(a) Compute the expectation of Z.

(b) Use this to describe an unbiased estimator for d.

Problem 3

The prefekt of the Chalmers CSE Department has a headache: he needs to form m administrative committees with committee i requiring at lest k_i members who need to be selected from amongst the n professors in the department. However, each professor j is only willing to serve in a selected subset of committees S_j where they think they have something useful to contribute and they have a limit of s_j different committees they could be part of given their time restrictions. Give a polynomial time algorithm to help the prefekt decide if there is a way to form such committees and if so how. State the running time of your algorithm.

Problem 4

Let's say we need to remove elements from a Bloom filter B with $|B| = b$ bits (recall, it is not a Bloom filter if the hashes are not uniform). For a deletion of item x we set the $\{h_1(x), \ldots, h_k(x)\}$ bits to zero (note, we assume, as in the lecture, that the $h_i(x)$ are pair-wise distinct). Deletions could obviously cause false negatives. The question is, how much of a problem is this?

(a) What is the probability that no false negatives caused by the deletion of one element, assuming that there are $n + 1$ elements in the Bloom filter before the deletion?

(b) What is the probability of d false negatives caused by the deletion of one element, assuming that there are $n + d$ elements in the bloom filter before the deletion?

(c) What is the expected number of items affected by one deletion?

(d) What is the expected number of false negatives caused by d deletions if there are $n + d$ elements in the bloom filter before the deletions? You might need to make further simplifying assumptions, or invoke a more complicated probability argument.

Problem 5

The space requirement of prefix trees depends greatly on the tree implementation. Recall, that in a prefix tree every node can have at most $|\Sigma|$ many descendants, one outgoing edge per character in the alphabet. The actual number of descendants will however very much depend on the depth of a node in the tree. Clearly, the root is likely to have all possible descendants, while nodes deeper in the tree might only have a few.

Assume that the only prefix tree implementation available to you implements a doubly-linked list in each node to store pointers to descendants and uses four 32-bit words (three pointers—previous, next, descendant—and one number for the character) per descendant and two 32 bit numbers for the first and last pointer.

Alternatively, one can store the pointer to descendants in an array, using a prefix as the index into the array. E.g., the root node could be replaced by an array of pointers indexed by the first character of the prefix which requires $|\Sigma|$ 32-bit words space.

The two approaches can be combined to form a "prefix forest" where an array with $|\Sigma|^k$ entries is used to store pointers to descendants (or null, if no such descendant exists) of all possible k-character prefixes in lexicographic order. A proper choice of k can reduce the memory footprint.

(a) Propose a criterion for choosing an optimal k given a prefix tree as input.

(b) Propose an algorithm for choosing an optimal k and converting a prefix tree to a prefix forest.