

Randomized Algorithms, Spring 2009: Project

version 1 – January 19, 2009

This project consists of two problems that each have both theoretical and practical aspects. The subproblems outlines a possible approach. If you follow the suggested approach, you must answer all subproblems to get full credit. However, you may choose a different approach, but then you should make a similarly thorough analysis and documentation to get full credit.

You are allowed to work in groups (of at most 3 students) and make a single hand-in per group. The report should be made using LaTeX or similar. Handwriting is not acceptable.

The evaluation of your report will put weight on simple short precise answers/documentation. For the programming subproblems, the evaluation will put weight on

- a precise description of the purpose of the experiment
- a well motivated design of the experiment
- a clear description of the outcome of the experiment, and
- a careful analysis and discussion of the outcome compared to theoretical predictions.

For the experiments you will need a source of random / pseudo-random numbers. In your report, you must explain carefully what source you have used. You may use the links on the course web page.

The hand-in should include program source code files in addition to the report in pdf format. All files are packaged in a zip-file and submitted via the course web page.

Deadline for hand-in: Friday March 6th at 12 noon.

Problem 1

This problem considers robustness of a score (grade) given to students based on a multiple choice test. The problem set-up is based on the paper Frandsen and Schwartzbach, *A singular choice for multiple choice*, [<http://doi.acm.org/10.1145/1189136.1189164>].

Assume a multiple choice test consists of n questions, each having 4 choices. For each question precisely one choice is correct. Students are allowed to make 0 or 1 “check” (cross) for each question. The score for a question is 1 if the student has checked the correct choice, $-\frac{1}{3}$ if the student has checked a wrong choice and 0 if no choices are checked. The score for the test is computed as the sum of the scores for all questions. The maximal score is therefore n . We assume that the test is used only to decide pass/fail, and the threshold for passing is a 50% score, i.e. a score $\geq \frac{n}{2}$.

Define a *challenged* student to be a student that knows the answers to at most 40% of the questions.

Clearly, a challenged student has nothing to lose by guessing the answers to the questions he does not know. Let us assume that he accordingly puts down checks at random choices (one per question), so he leaves no questions unanswered.

Define a multiple choice test to be *good*, if the probability that a challenged student passes is at most 5%.

A teacher has to make a test, and naturally he wants it to be *good*. He suspects that if he has enough questions in the test then it will be good. This is indeed correct.

In this problem you are required to find a (small) size n of the test that ensures it is *good*. You are required to do this both theoretically using Chernoff bounds (see Motwani and Raghavan, ch.4) and experimentally with high confidence.

You may find the following notation helpful: Assume the challenged student guesses $m = \frac{3}{5}n$ questions, define

$$X_i = \begin{cases} 1, & \text{if the } i\text{th guess is correct} \\ 0, & \text{otherwise} \end{cases}$$

Define $X = \sum_{i=1}^m X_i$

1.1 Subproblem Determine $E[X]$.

1.2 Subproblem Show that the challenged student only passes if $X \geq \frac{3}{2}E[X]$.

1.3 Subproblem Determine a size n of the test for which the challenged student only passes with probability at most .05 using the Chernoff bound technique.

The teacher suspects that a much smaller n than the one resulting from the Chernoff bound really suffices to make the test good. He decides to run an experiment, where he simulates challenged students making guesses. For a candidate n , he simulates R challenged students. Define the following notation

$$Y_{n,i} = \begin{cases} 1, & \text{if the } i\text{th simulated challenged student passes the test} \\ & \text{with } n \text{ questions} \\ 0, & \text{otherwise} \end{cases}$$

Let $p_n = \Pr(Y_{n,i} = 1)$, i.e. the test is good precisely when $p_n \leq .05$. Let $Y_n = \sum Y_{n,i}$.

1.4 Subproblem Determine $E[Y_n]$.

The teacher decides to accept a test as good, if the outcome $Y_n \leq 0.04R$.

1.5 Subproblem Determine a value of R that is sufficiently large to ensure that a non-good test is accepted as good with probability at most 0.05. You should use the Chernoff bound technique.

Now comes the programming part.

1.6 Subproblem You should implement a method that given n, R uses a random/pseudorandom source to determine an experimental value for Y_n .

1.7 Subproblem Using that method you should construct and implement

an algorithm that determines a (small) n that ensures a good multiple choice test.

1.8 Subproblem Analyze what confidence you can have in the outcome from running the algorithm.

1.9 Subproblem Run the algorithm and compare the outcome of the experiment with the earlier theoretically determined minimal size of a good test.

Problem 2

This problem considers equality test of multisets. This can be used to check that a sorting algorithm works correctly. The problem set-up is based on the paper Blum and Kannan, *Designing programs that check their work*, [<http://doi.acm.org/10.1145/200836.200880>]. We phrase the problem in terms of databases.

Swat Ltd. has a profitable market in selling customer information. They have over the years collected a huge database of peoples hobbies, making it possible to direct ads towards the most likely customers. They suspect that one or more hackers has obtained a copy of their database. They notify the police who seize the computers of 8 known hackers. The thieves(s) have anticipated this move, so they only keep permuted versions of the database on their computers. In addition they have gotten all their friends to keep random databases on their computers, to confuse the police. As a result the police find database lookalikes on all 8 computers. The police is smart enough to realize that each computer either contain random records (not related to the Swat Ltd database) or a copy of the real database where records have been permuted.

As an example, if the real database looks like

```
[Egon Hausgaard,soccer]
[Vera Nicolaisen,poker]
[Holger Uggerhøj,movies]
```

then a permuted copy of the database may look like

[Holger Uggerhøj,movies]
[Egon Hausgaard,soccer]
[Vera Nicolaisen,poker]

and a list of random records may look like

[Niels Vestergaard,riding]
[Silius Thur,archery]
[Winona Møller,gardening]

Since it is not a crime to have random records on your computer, the police need to distinguish permuted databases from random records. They realize that sorting may do the job, but for efficiency reasons they decide to use variants of the randomized fingerprint technique (Motwani and Raghavan, ch.7 and 8) for this purpose.

Formally they consider the problem:

Multiset Equality

Input: $X = [x_1, \dots, x_n]$, $Y = [y_1, \dots, y_n]$, two lists of records
 k , a positive integer.

Output: Yes, if X and Y represent the same multiset of records.
No, otherwise.

The output is allowed to be wrong with probability $\leq 2^{-k}$

Assumption: A tuple is described by b bits. Since the choice between various algorithms depends on the size of b and n , we will assume that a tuple is represented as a string of max 40 characters (8 bits each), i.e. $b = 320$, and $n \leq 2^{19}$. These restrictions are satisfied by the testdata mentioned later.

We use $k = 10$, i.e. a judge will convict a hacker if a randomized algorithm that err with probability at most 2^{-10} has declared that the database on the said hackers computer has the same multiset of tuples as the genuine Swat Ltd database. (Is that reasonable?)

Define polynomials $f_X(z) = (z - x_1) \cdots (z - x_n)$ and $f_Y(z) = (z - y_1) \cdots (z - y_n)$. Here we have interpreted each tuple x_i or y_i as a b -bit integer.

Because of unique factorization in polynomial rings the polynomials f_X and f_Y are identical if and only if $X = [x_1, \dots, x_n]$ and $Y = [y_1, \dots, y_n]$ are

identical when regarded as multisets.

If f_X and f_Y are distinct, then the Schwartz-Zippel Lemma tells us that we have a high probability of discovering this when evaluating f_X and f_Y on a randomly chosen number.

This suggests a simple probabilistic algorithm for multiset equality test: Evaluate polynomials f_X and f_Y on a random number and look for equality of results. However, this simple algorithm is highly inefficient, taking quadratic time in the bit size of the input, when using “school” arithmetic.

This project has 2 parts. In the first part you should improve the above simple algorithm to become as efficient as you can. That includes a detailed description of your improved algorithm with time analysis and analysis of the error bound. You are free to use your creativity in combination with the tools from the course with one restriction. Your solution must not be significantly less efficient than the suggested approach below, and it should preferably be faster than sorting. The error and time analysis of your approach should also be as rigorous as the analysis in the suggested approach.

In the second part you should implement the algorithm and run experiments to test its correctness and efficiency. That includes a comparison with sorting on the testdata. You may find the original Swat Ltd. database and the 8 hacker files here

<http://www.daimi.au.dk/~gudmund/randomF09/database/>

Which hackers had stolen the Swat database? Describe which experiments you have run to determine your answer, and how long time they took.

Suggested approach

The main reason for the inefficiency of the simple algorithm is that the evaluation of polynomials f_X and f_Y on a b -bit number results in an enormous number containing bn or more bits. So the total evaluation takes $\Omega(n)$ arithmetic operations, some of which are applied to $\Omega(bn)$ bit numbers.

To improve efficiency we may use fingerprinting to make sure we only have

to do arithmetic on numbers that are so small that they fit in a computer word.

Let $p = 2^{31} - 1$, a 31 bit prime number. Arithmetic modulo p may be done via 64 bit long arithmetic. Let F_p denote the finite field with p elements i.e. the numbers $\{0, 1, \dots, p - 1\}$ with arithmetic modulo p .

One might think that we just have to evaluate f_X and f_Y modulo p . Unfortunately this approach is too naive. We have to be a bit more sophisticated to get a sufficiently good error bound.

Let H be a 2-universal class of hash-functions from $\{0, 1\}^b$ into F_p . (Don't worry about the construction of H yet)

2.1 Subproblem Show that if $X = [x_1, \dots, x_n]$ and $Y = [y_1, \dots, y_n]$ are distinct multisets and h is selected uniformly randomly from H then multisets $h(X) = [h(x_1), \dots, h(x_n)]$ and $h(Y) = [h(y_1), \dots, h(y_n)]$ are identical with probability at most n/p .

In the following, let $h \in H$. It is a fact that if $h(X) = [h(x_1), \dots, h(x_n)]$ and $h(Y) = [h(y_1), \dots, h(y_n)]$ are distinct multisets then $f_{h(X)}(z) = (z - h(x_1)) \cdots (z - h(x_n))$ and $f_{h(Y)}(z) = (z - h(y_1)) \cdots (z - h(y_n))$ are also distinct polynomials over the field F_p .

2.2 Subproblem Show that if $f_{h(X)}$ and $f_{h(Y)}$ are distinct and w is chosen uniformly randomly from F_p then $f_{h(X)}(w)$ and $f_{h(Y)}(w)$ are identical with probability at most n/p .

We still need to construct a class of 2-universal hash functions H . One might think that we could simply use the class from section 8.4.3 in Motwani and Raghavan. However, that construction requires us to use arithmetic modulo a prime larger than 2^b , which is inefficient when b is larger than the word size of our computer. We really only want to do arithmetic modulo p as defined earlier. However, there is a way around this obstacle.

A bit string of length b may be divided into blocks of length 30, where each block can be interpreted as a number in F_p . This suggests defining a 2-universal class H_s of hash functions from F_p^s into F_p , and use $H_{\lceil b/30 \rceil}$ as our H .

Define $H_s = \{ h_{a_1, \dots, a_s} \mid a_1, \dots, a_s \in F_p \}$, where $h_{a_1, \dots, a_s}(x_1, \dots, x_s) =$

$$\sum_{i=1}^s a_i x_i.$$

2.3 Subproblem Show that H_s is 2-universal (Hint: use Principle of Deferred Decisions, MR section 7.1).

2.4 Subproblem Explain how to construct the desired 2-universal H such that $h \in H$ can be computed using arithmetic modulo p .

2.5 Subproblem Formulate and analyse an algorithm for testing whether two multisets are equal (based on your answers to the previous subproblems). The algorithm should have runtime $O(bn)$, i.e. linear in the input size, for all values b and n . You may assume that arithmetic operations modulo p take unit time. It should have error probability bounded by 2^{-10} when the input satisfies the restriction $n \leq 2^{19}$.