Your name
Your student number

CS 466/666: ALGORITHM DESIGN AND ANALYSIS
PROBLEM SET 4

DUE ON: NOVEMBER 1, 2017

ACKNOWLEDGEMENTS.

Acknowledge **all collaborators** with whom you discussed any of the problems in this problem set and **all external sources** that you may have used in the completion of the assignment. See the website for the full homework policy.

1. PROBLEM 1: BLOOM FILTERS (TAKE II)

A *(one-sided) δ-error algorithm* for the SET MEMBERSHIP problem is a randomized algorithm A that sees n objects $x_1, \ldots, x_n \in U$ and then a query $a \in U$, and

- A always outputs 1 when $a = x_i$ for some $i \leq n$; and
- A outputs 0 with probability at least $1 - \delta$ when $a \notin \{x_1, \ldots, x_n\}$.

In the midterm, you showed that the simple SET MEMBERSHIP algorithm is a δ-error algorithm for the problem which requires $\lceil n/\delta \rceil + \lceil \log |H| \rceil$ bits of memory when $H = \{h : U \to [\lceil \frac{n}{\delta} \rceil]\}$ is a universal family of hash functions.

(a) Show that there is a simple variant of the SET MEMBERSHIP algorithm that achieves the same guarantee but requires much less memory:

Theorem 1. *For every pairwise-independent family of hash functions $H = \{h : U \to [\lceil \frac{2n}{\sqrt{\delta}} \rceil]\}$, there is a δ-error algorithm for* SET MEMBERSHIP *that requires only $\left\lceil \frac{2n}{\sqrt{\delta}} \right\rceil + 2 \lceil \log |H| \rceil$ bits of memory.*

Proof. Enter your answer here. □

(b) In the midterm, you also showed that by using multiple hash tables and a universal family of hash functions $H = \{h : U \to [3n]\}$, we can also obtain a δ-error algorithm for SET MEMBERSHIP with memory complexity $\lceil \log \frac{1}{\delta} \rceil (3n + \lceil \log |H| \rceil)$. When is the algorithm from part (i) better than this algorithm?

Enter your answer here.

(c) **(Bonus.)** Can we further improve the result above to use even less memory? In general, what is the minimal amount of memory required to implement a δ-error algorithm for SET MEMBERSHIP?

2. Problem 2: Problems in disguise

Use results we have seen in class to provide efficient algorithms for the following problems. In these problems, we have n machines $M_1, \ldots, M_n$ as well as a separate machine M^* that is responsible for coordinating the n other machines.

(a) Consider the setting where each machine M_i is either *free* or *busy*. The machine M^* can ask any machine M_i whether it is free or busy. Fix some $\epsilon > 0$. How efficiently can it estimate the number of machines that are free up to additive error ϵn?

Enter your answer here.

(b) Consider now the setting where M^* must assign some jobs with IDs $j_1, \ldots, j_n$ to the machines $M_1, \ldots, M_n$ in a way that, if it is later asked about the status of a job ID j, it knows which machine M_i has the corresponding job. How efficiently can it do this, where by efficiently we want (i) the memory or M^* to be as small as possible, and (ii) to minimize the maximum number of jobs that any machine M_i receives.

Enter your answer here.

(c) Now we consider a different setup where the machines are in a ring network: M^* can only send messages to M_1, machine M_i can only send messages to M_{i+1} for any $i < n$, and M_n can only send messages to M^*. As in part (a), M^* wants to know approximately how many machines are free. How efficiently can we do this, where now efficiency is measured in terms of the length of the messages passed around the ring?

Enter your answer here.

3. Problem 3: Polynomial identity testing

In the study of algebra and number theory, one encounters a number of elegant and surprising polynomial identities. Take, for example,

$$x^3 + y^3 = (x + y)(x^2 - xy + y^2)$$

or

$$(x_1^2 + x_2^2)(x_3^2 + x_4^2) = (x_1x_3 - x_2x_4)^2 + (x_2x_3 + x_1x_4)^2.$$

How efficiently can we verify whether identities such as the ones above are true? This is the famous *polynomial identity testing* (PIT) problem, and if you can identify an efficient deterministic algorithm for the problem, you will get an automatic 100% in the course (among other distinctions!). In this question, you will show that with randomized algorithms there is an amazingly simple solution to the problem:

Algorithm 1: PIT algorithm(p, q)

Let $\mathbb{F}$ be a finite field of size $|\mathbb{F}| > 2d$;
Draw $x \in \mathbb{F}^n$ uniformly at random;
Accept iff $p(x) = q(x)$;

(a) Prove the following fundamental fact about the number of roots of multivariate polynomials.

Lemma 1. *Let p be a non-zero polynomial on n variables with degree at most d. Then for any field $\mathbb{F}$, when $x = (x_1, \ldots, x_n) \in \mathbb{F}^n$ is drawn uniformly at random we have*

$$\Pr_{x \in \mathbb{F}^n}[p(x_1, \ldots, x_n) = 0] \leq \frac{d}{|\mathbb{F}|}.$$

Proof. Enter your answer here. □

Hint. You can proceed by induction on n. You may use the fact that the lemma is true for univariate polynomials (i.e., when $n = 1$) without proof.

(b) Use the lemma above to show that the PIT algorithm above is valid when $q \equiv 0$.

Theorem 2. *When the PIT algorithm above is run with $q \equiv 0$ and p is any polynomial of degree at most d, then*

- *If $p \equiv 0$, the algorithm always accepts; and*
- *If $p \not\equiv 0$, the algorithm rejects with probability at least $\frac{1}{2}$.*

Proof. Enter your answer here. □

(c) Use the theorem above to show that the PIT algorithm is valid for *any* polynomials p and q.

Theorem 3. *When the PIT algorithm above is run with any two polynomials p and q of degree at most d, then*

- *If $p \equiv q$, the algorithm always accepts; and*
- *If $p \not\equiv q$, the algorithm rejects with probability at least $\frac{1}{2}$.*

Proof. Enter your answer here. □