6.5240 Sublinear Time Algorithms November 2, 2022

Lecture 15

Lecturer: Shyan Akmal *Scribe: Danny Mittal*

In this lecture, we study linear functions from the perspective of sublinear time algorithms, aiming to test linearity as well as evaluate linear functions with errors. We also revisit edge sampling, showing a lower bound and a new runtime.

1 Linear Functions

1.1 Introduction

1.1.1 Definitions

Definition 1 *$\mathbb{F}_2$ is the finite field with two elements, 0 and 1, so that $0+0=1+1=0$ and $0+1=1+0=1$.*

Definition 2 *A function $l : \mathbb{F}_2^n \to \mathbb{F}_2$ is* linear *if for all $x, y \in \mathbb{F}_2^n$, $l(x+y) = l(x) + l(y)$.*

In general, we will use l to refer to linear functions, and f to refer to functions that may not be linear. Additionally, we will use $N = |\mathbb{F}_2^n| = 2^n$ to refer to the size of the domain $\mathbb{F}_2^n$.

Definition 3 *A function $f : \mathbb{F}_2^n \to \mathbb{F}_2$ is* ϵ-far from linear *if more than ϵN point values of f need to be changed in order for f to be linear. f is ϵ-close to linear if it is not ϵ-far from linear.*

1.1.2 Model

In our model, we are given a function $f : \mathbb{F}_2^n \to \mathbb{F}_2$, and can access it using point value queries, i.e., we can query $x \in \mathbb{F}_2^n$ and $f(x)$ will be returned.

1.1.3 Problems

In this model, we will look to provide algorithms to solve two problems.

Problem 1 *Distinguish f being linear from f being ϵ-far from linear.*

Problem 2 *Under the assumption that f is ϵ-close to some linear l, given $x \in \mathbb{F}_2^n$, compute $l(x)$.*

As relates to the second problem, note that we are not given l nor do we have any information about it other than f itself. Given this, it may seem that the problem is improperly defined, as f could theoretically be ϵ-close to multiple linear l. However, as long ϵ isn't too big, l is unique. In particular, we will proceed with the assumption that $\epsilon < \frac{1}{6}$, under which l is guaranteed to be unique.

We will explore the second problem first.

1.2 Computing values of l given f that is ϵ-close to l

1.2.1 Motivation

Consider the following. Say that we have some algorithm A that solves some problem P but with some errors, so that, for example, A produces the correct answer to P on $\frac{9}{10}$ of inputs, but is always incorrect on the other $\frac{1}{10}$ of inputs (i.e., its correctness is not random, so it can't simply be repeated to have a high probability of producing the correct answer). Is it possible to produce an $\tilde{A}$ that corrects A, so that it is correct with probability $1-\delta$ for small δ?

Problem 2 can be thought of as an instance of this situation. We have f, which we know to be ϵ-close to some linear l. We could think of this as f being an imperfect or inaccurate version of l that we have access to when we instead want to know the values of l itself. In this instance, we will see that it is possible to produce an error correcting algorithm using f that computes l correctly with high probability.

1.2.2 The Algorithm

The algorithm is as follows.

> Given $x \in \mathbb{F}_2^n$ such that we want to compute $l(x)$, choose y uniformly at random from $\mathbb{F}_2^n$, then query the values of $f(y)$ and $f(x-y)$ and return the result $f(y) + f(x-y)$.

The motivation for this algorithm is that for linear l, $l(x) = l(y) + l(x-y)$, so if f were actually just l, the algorithm would correctly compute the result. We will argue that the algorithm still computes the correct result with high probability.

Theorem 1 *The above algorithm correctly computes $l(x)$ with probability at least $1 - 2\epsilon$; since $\epsilon < \frac{1}{6}$, this is more than $\frac{2}{3}$.*

Proof Since $l(x) = l(y) + l(x-y)$, in order for the result of the algorithm to be correct it suffices for $f(y)$ and $f(x-y)$ to match the respective values of l.

As f is ϵ-close to l, at most ϵ fraction of the point values of f differ from l, and so as y is selected uniformly at random from the domain $\mathbb{F}_2^n$, the probability that $f(y) \neq l(y)$ is at most ϵ.

The same logic applies to $x-y$. For any possible value $z \in \mathbb{F}_2^n$, there is exactly one y so that $x-y=z$, that being the value $y = x - z$. Therefore, since y is chosen uniformly at random from $\mathbb{F}_2^n$, the value $x-y$ is also chosen uniformly at random from $\mathbb{F}_2^n$, and so the probability that $f(x-y) \neq l(x-y)$ is also at most ϵ.

Thus, by using union bound, we can see that the probability that $f(y) \neq l(y)$ or $f(x-y) \neq l(x-y)$ is at most $\epsilon + \epsilon = 2\epsilon$. It follows that the algorithm succeeds with probability at least $1 - 2\epsilon$. ■

By using Chernoff bounds as in the first problem from problem set 0, we can perform $O(\lg(\frac{1}{\delta}))$ iterations of this algorithm which correctly computes $l(x)$ with probability at least $\frac{2}{3}$ to obtain an algorithm which correctly computes $l(x)$ with probability at least any desired $1 - \delta$

Also notice that the fact that we can use f to calculate l correctly with high probability implies that l is unique.

1.2.3 A Note on Naming

The idea used in the above algorithm is sometimes called "worst case to average case reduction" in the literature. To explain this naming, consider an algorithm which simply returns $f(x)$ as its answer. This algorithm is guaranteed to be wrong in the worst case (assuming $f \neq l$), but in the average case succeeds with probability at least $1 - \epsilon$. You could therefore view our above algorithm as applying that algorithm in a way that brings the average case behavior to all cases.

1.3 Testing Linearity

We now move to the first problem, where we want to distingush f being linear from f being ϵ-far from linear.

1.3.1 The Algorithm

The algorithm is as follows.

```
Repeat the following $t$ times:
    Choose $x, y$ uniformly at random from $\mathbb{F}_2^n$, and check whether $f(x+y) = f(x) + f(y)$. If not, fail.
At the end, if we haven't failed yet, pass.
```

The correctness of this algorithm on linear f is easy to see.

Theorem 2 *The above algorithm always passes for linear f.*

Proof If f is linear, then it satisfies $f(x+y) = f(x) + f(y)$ for all $x, y \in \mathbb{F}_2^n$, so it is impossible for the algorithm to fail. ■

The more difficult task is then showing that, for an appropriately chosen t, the algorithm fails for f that is ϵ-far from linear with high probability.

Theorem 3 *If f is ϵ-far from linear, then the probability that the above algorithm failing on some given iteration is at least ϵ.*

Given theorem 6, it suffices to choose $t = \Theta(\frac{1}{\epsilon})$ for the algorithm to fail on functions ϵ-far from linear with probability $\Theta(1)$. The majority of the rest of the lecture will now be devoted to proving theorem 6.

1.3.2 The Framework for the Proof of Theorem 6

In this section we will more deeply rely on linear algebra. $\mathbb{F}_2^n$ can be thought of as a vector space over the field $\mathbb{F}_2$, with its dimension being n. It therefore has a basis $b_1, b_2, \ldots, b_n$ of size n, and so every element x of $\mathbb{F}_2^n$ can be thought of as a linear combination $x_1b_1 + x_2b_2 + \cdots + x_nb_n$ of the basis elements.

It follows that any linear function l is determined by its values at the basis elements, because for any $x = x_1b_1 + x_2b_2 + \cdots + x_nb_n \in \mathbb{F}_2^n$,

$$l(x) = l(x_1b_1 + x_2b_2 + \cdots + x_nb_n)$$

$$l(x) = x_1l(b_1) + x_2l(b_2) + \cdots + x_nl(b_n).$$

This means that we can think of $l(x)$ as being an inner product of some a with x written as $\langle a, x\rangle$, where we can express x as $(x_1, x_2, \ldots, x_n)$ and a as $(l(b_1), l(b_2), \ldots, l(b_n))$. Thus, the linear functions $\mathbb{F}_2^n \to \mathbb{F}_2$ are precisely the inner products $\langle a, x\rangle$ for $a \in \mathbb{F}_2^n$. Note that inner products are symmetric, and so are linear in both variables (i.e., both $\langle a, b\rangle + \langle a, c\rangle = \langle a, b+c\rangle$ and $\langle a, c\rangle + \langle b, c\rangle = \langle a+b, c\rangle$ hold), which will be useful later.

It will be convenient mathematically for us to, instead of dealing with functions f that have outputs in $\mathbb{F}_2$, consider $F(x) = (-1)^{f(x)}$, which has as its outputs the real numbers 1 and -1. Similarly, we make the following definition.

Definition 4 *For all $a \in \mathbb{F}_2^n$, define $\chi_a(x) = (-1)^{\langle a,x\rangle}$*

These χ_a are called character functions. We will explore the properties of character functions in the below claims.

Claim 1 *For all $a, b, x, y \in \mathbb{F}_2^n$, we have the following identities:*

$$\chi_a(x)\chi_a(y) = \chi_a(x+y)$$

$$\chi_a(x)\chi_b(x) = \chi_{a+b}(x)$$

These identities can be proven in essentially the same way using linearity, and the fact that exponents add ($a^b a^c = a^{b+c}$ for a, b, c in the right domains).
Proof

$$\chi_a(x)\chi_a(y) = (-1)^{\langle a,x\rangle}(-1)^{\langle a,y\rangle} = (-1)^{\langle a,x\rangle+\langle a,y\rangle} = (-1)^{\langle a,x+y\rangle} = \chi_a(x+y)$$
$$\chi_a(x)\chi_b(x) = (-1)^{\langle a,x\rangle}(-1)^{\langle b,x\rangle} = (-1)^{\langle a,x\rangle+\langle b,x\rangle} = (-1)^{\langle a+b,x\rangle} = \chi_{a+b}(x)$$

■

Claim 2 *For nonzero $a \in \mathbb{F}_2^n$, there are an equal number of x such that $\chi_a(x) = 1$ as there are such that $\chi_a(x) = -1$. Equivalently,*

$$\sum_{x\in\mathbb{F}_2^n} \chi_a(x) = 0.$$

The elegant formulation $\sum_{x\in\mathbb{F}_2^n} \chi_a(x) = 0$ is an example of how transitioning from $f(x)$ to $(-1)^{f(x)}$ is convenient mathematically.
Proof First note that $\sum_{x\in\mathbb{F}_2^n} \chi_a(x) = 0$ is equivalent to $\mathbb{E}_x[\chi_a(x)] = 0$, i.e., the expected value of $\chi_a(x)$ for x chosen uniformly from $\mathbb{F}_2^n$ is 0. We will show the latter statement.

Now, for some $y \in \mathbb{F}_2^n$, consider $\chi_a(y)\mathbb{E}_x[\chi_a(x)]$. We can rewrite this as $\mathbb{E}_x[\chi_a(y)\chi_a(x)]$, which by claim 1, is equal to $\mathbb{E}_x[\chi_a(x+y)]$. However, just as in the proof in 1.2.2 for the algorithm solving problem 2, if x is chosen uniformly at random from $\mathbb{F}_2^n$ and y is a constant, then $x + y$ is also chosen uniformly at random from $\mathbb{F}_2^n$. It follows that $\mathbb{E}_x[\chi_a(x + y)] = \mathbb{E}_z[\chi_a(z)] = \mathbb{E}_x[\chi_a(x)]$. Thus, we have the equality

$$\chi_a(y)\mathbb{E}_x[\chi_a(x)] = \mathbb{E}_x[\chi_a(x)]$$

which we can write as

$$(\chi_a(y) - 1)\mathbb{E}_x[\chi_a(x)] = 0.$$

Given that $a \neq 0$, there must be some y such that $\chi_a(y) \neq 1$, i.e., $\langle a, y\rangle \neq 0$. For example, we can take $y = b_k$ where b_k is some basis element such that a's coefficient of b_k is 1, so that $\langle a, y\rangle = 1$. By fixing such a y, we see that it must be the case that $\mathbb{E}_x[\chi_a(x)] = 0$. ■

We will now define inner products for these functions as well.

Definition 5 *Define $\langle F, G\rangle = \sum_{x\in\mathbb{F}_2^n} F(x)G(x)$*

Note that this inner product is linear in both variables as inner products should be.

Claim 3 *For all $a \in \mathbb{F}_2^n$, $\langle \chi_a, \chi_a\rangle = N$ For all $a, b \in \mathbb{F}_2^n$ with $a \neq b$, $\langle \chi_a, \chi_b\rangle = 0$*

Proof The first part of the claim can be seen easily. χ_a has outputs that are ± 1, whose squares are all 1, so their sum will simply be the size of the domain being summed over:

$$\langle \chi_a, \chi_a\rangle = \sum_{x\in\mathbb{F}_2^n} \chi_a(x)\chi_a(x) = \sum_{x\in\mathbb{F}_2^n} \chi_a(x)^2 = \sum_{x\in\mathbb{F}_2^n} 1 = |\mathbb{F}_2^n| = N.$$

To see the second part of the claim, we can use claim 2:

$$\langle \chi_a, \chi_b\rangle = \sum_{x\in\mathbb{F}_2^n} \chi_a(x)\chi_b(x) = \sum_{x\in\mathbb{F}_2^n} \chi_{a+b}(x).$$

(note that the last step uses the second part of claim 1). Since $a \neq b$, $a + b \neq 0$, and so by claim 2,

$$\sum_{x\in\mathbb{F}_2^n} \chi_{a+b}(x) = 0.$$

Therefore,

$$\langle \chi_a, \chi_b \rangle = 0.$$

■

We now move to our final claim.

Claim 4 *The set $S = \{\chi_a | a \in \mathbb{F}_2^n\}$ forms a basis for the space of functions $\mathbb{F}_2^n \to \mathbb{R}$ over the field $\mathbb{R}$.*

Proof First, $\mathbb{F}_2^n \to \mathbb{R}$ has dimension N, which can be seen by considering the trivial basis of indicator functions, i.e., the functions J_a such that $J_a(a) = 1$ and $J_a(x) = 0$ for $x \neq a$. Any function F can be expressed as the linear combination $\sum_{a \in \mathbb{F}_2^n} F(a) J_a$, and the indicator functions are linearly independent, so they must be a basis.

S has size $|S| = N$, so it has the correct size to be a basis.

It remains now to show that the members of S are linearly independent, i.e., that no nontrivial linear combination of the members of S sums to 0. To do this, consider any linear combination of members S of that sums to 0,

$$\sum_{a \in \mathbb{F}_2^n} c_a \chi_a = 0.$$

We wish to show that for all $b \in \mathbb{F}_2^n$, the coefficient c_b is 0. We can do so as follows. Since $\sum_{a \in \mathbb{F}_2^n} c_a \chi_a$ is 0, its inner product with anything is also 0, so:

$$0 = \left\langle \sum_{a \in \mathbb{F}_2^n} c_a \chi_a, \chi_b \right\rangle = \sum_{a \in \mathbb{F}_2^n} c_a \langle \chi_a, \chi_b \rangle = c_b \langle \chi_b, \chi_b \rangle = c_b N$$

where we can move out the sum as inner products are linear, and the $\sum_{a \in \mathbb{F}_2^n} c_a \langle \chi_a, \chi_b \rangle = c_b \langle \chi_b, \chi_b \rangle$ step follows from claim 3. As N is positive, we can conclude that $c_b = 0$. ■

With the framework for our proof of theorem 6 built up, we now return to the proof itself.

1.3.3 The Proof of Theorem 6

Recall theorem 6, which can be stated as follows:

Given $f : \mathbb{F}_2^n \to \mathbb{F}_2$ that is ϵ-far from linear, if we choose x, y uniformly at random from $\mathbb{F}_2^n$, then the probability that $f(x + y) \neq f(x) + f(y)$ is at least ϵ.

Proof Recall $F : \mathbb{F}_2^n \to \mathbb{R}$ as defined before, i.e., $F(x) = (-1)^{f(x)}$. By claim 4, the χ_a are a basis for $\mathbb{F}_2^n \to \mathbb{R}$, so we can express F as a linear combination of them:

$$F = \sum_{a \in \mathbb{F}_2^n} \hat{F}(a) \chi_a$$

where $\hat{F}$ is a common notation for the coefficients in the linear combination.

For any $b \in \mathbb{F}_2^n$, as in the proof for claim 4,

$$\langle F, \chi_a \rangle = \sum_{a \in \mathbb{F}_2^n} \hat{F}(a) \langle \chi_b, \chi_a \rangle = \hat{F}(b) \langle \chi_b, \chi_b \rangle = N \hat{F}(b)$$

from which it follows that

$$\hat{F}(b) = \frac{\langle F, \chi_a \rangle}{N}.$$

By expanding the above inner product as

$$\langle F, \chi_b \rangle = \sum_{a \in \mathbb{F}_2^n} F(a) \chi_b(a)$$

we can see that $\frac{\langle F, \chi_a \rangle}{N}$ is actually just the expected value of $F(a)\chi_b(a)$ when a is chosen uniformly at random from $\mathbb{F}_2^n$, so

$$\hat{F}(b) = \mathbb{E}_a[F(a)\chi_b(a)].$$

We now make an important, though trivial, observation, which is that because F and χ_b have outputs in $\{\pm 1\}$, $F(a) \neq \chi_b(a)$ is equivalent to $F(a)\chi_b(a) = -1$. We can use this observation as follows. Let p be the probability that $F(a) \neq \chi_b(a)$. As f is ϵ-far from linear, $p \geq \epsilon$. Now, p is also the probability that $F(a)\chi_b(a) = -1$. Since the only other possible value for $F(a)\chi_b(a)$ is 1, we can write

$$\hat{F}(b) = \mathbb{E}_a[F(a)\chi_b(a)] = p(-1) + (1-p)(1) = -p + 1 - p = 1 - 2p.$$

Since $p \geq \epsilon$, we can thus say

$$\hat{F}(b) \leq 1 - 2\epsilon.$$

In a similar vein, let q be the probability that $f(x+y) \neq f(x)+f(y)$ when x, y are chosen uniformly at random, recalling that we aim to prove $q \geq \epsilon$. $f(x+y) \neq f(x)+f(y)$ is equivalent to $F(x+y) \neq F(x)F(y)$, which is equivalent to $F(x+y)F(x)F(y) = -1$. Just as before, since the only possible values of this expression are ± 1, we can write

$$\mathbb{E}_{x,y}[F(x+y)F(x)F(y)] = q(-1) + (1-q)(1) = 1 - 2q.$$

On the other hand, we can greatly simplify $\mathbb{E}_{x,y}[F(x+y)F(x)F(y)]$. Consider expanding each instance of F as a linear combination of the χ_a:

$$\begin{aligned}\mathbb{E}_{x,y}[F(x+y)F(x)F(y)] &= \mathbb{E}_{x,y}\left[\left(\sum_{a\in\mathbb{F}_2^n} \hat{F}(a)\chi_a(x)\right)\left(\sum_{b\in\mathbb{F}_2^n} \hat{F}(b)\chi_b(y)\right)\left(\sum_{c\in\mathbb{F}_2^n} \hat{F}(c)\chi_c(x+y)\right)\right] \\ &= \mathbb{E}_{x,y}\left[\sum_{a\in\mathbb{F}_2^n}\sum_{b\in\mathbb{F}_2^n}\sum_{c\in\mathbb{F}_2^n} \hat{F}(a)\hat{F}(b)\hat{F}(c)\chi_a(x)\chi_b(y)\chi_c(x+y)\right].\end{aligned}$$

We can simplify the expression $\chi_a(x)\chi_b(y)\chi_c(x+y)$ using the properties from claim 1:

$$\chi_a(x)\chi_b(y)\chi_c(x+y) = \chi_a(x)\chi_b(y)\left(\chi_c(x)\chi_c(y)\right) = \left(\chi_a(x)\chi_c(x)\right)\left(\chi_b(y)\chi_c(y)\right) = \chi_{a+c}(x)\chi_{b+c}(y).$$

So,

$$\mathbb{E}_{x,y}[F(x+y)F(x)F(y)] = \mathbb{E}_{x,y}\left[\sum_{a\in\mathbb{F}_2^n}\sum_{b\in\mathbb{F}_2^n}\sum_{c\in\mathbb{F}_2^n} \hat{F}(a)\hat{F}(b)\hat{F}(c)\chi_{a+c}(x)\chi_{b+c}(y)\right].$$

By linearity of expectation, we can move the sums and coefficients out of the expectation to obtain

$$\sum_{a\in\mathbb{F}_2^n}\sum_{b\in\mathbb{F}_2^n}\sum_{c\in\mathbb{F}_2^n} \hat{F}(a)\hat{F}(b)\hat{F}(c)\mathbb{E}_{x,y}\left[\chi_{a+c}(x)\chi_{b+c}(y)\right].$$

Furthermore, we can separate the terms depending on x in the expectation from the terms depending on y:

$$\sum_{a\in\mathbb{F}_2^n}\sum_{b\in\mathbb{F}_2^n}\sum_{c\in\mathbb{F}_2^n} \hat{F}(a)\hat{F}(b)\hat{F}(c)\mathbb{E}_x\left[\chi_{a+c}(x)\right]\mathbb{E}_y\left[\chi_{b+c}(y)\right].$$

We can now greatly simplify this expression by using claim 3: for $a \neq c$, $a+c \neq 0$, so the first expectation will be 0, and similarly for the second expectation, so the only nonzero terms come from when $a = b = c$, in which cases the expectations are the expectation $\mathbb{E}_x[\chi_0(x)]$, which is 1 as $\chi_0(x)$ is always 1. Thus,

$$\mathbb{E}_{x,y}[F(x+y)F(x)F(y)] = \sum_{a\in\mathbb{F}_2^n} \hat{F}(a)\hat{F}(a)\hat{F}(a)(1)(1) = \sum_{a\in\mathbb{F}_2^n} \hat{F}(a)^3$$

$$1 - 2q = \sum_{a \in \mathbb{F}_2^n} \hat{F}(a)^3.$$

Finally, we will bound q from below by bounding $\sum_{a \in \mathbb{F}_2^n} \hat{F}(a)^3$ from above. First recall that $\hat{F}(a) \leq 1 - 2\epsilon$ for all $a \in \mathbb{F}_2^n$, so that we can say

$$\sum_{a \in \mathbb{F}_2^n} \hat{F}(a)^3 \leq \sum_{a \in \mathbb{F}_2^n} (1 - 2\epsilon)\hat{F}(a)^2 \leq (1 - 2\epsilon) \sum_{a \in \mathbb{F}_2^n} \hat{F}(a)^2.$$

Though the expressions look similar, sums of squares tend to be convenient, as they are in this case, because

$$\sum_{a \in \mathbb{F}_2^n} \hat{F}(a)^2 = \frac{\langle F, F \rangle}{N}.$$

We can see this by again expanding F into a linear combination of the χ_a:

$$\langle F, F \rangle = \sum_{a \in \mathbb{F}_2^n} \hat{F}(a) \langle \chi_a, F \rangle = \sum_{a \in \mathbb{F}_2^n} \sum_{b \in \mathbb{F}_2^n} \hat{F}(a)\hat{F}(b) \langle \chi_a, \chi_b \rangle.$$

By applying claim 3, this reduces to

$$\sum_{a \in \mathbb{F}_2^n} \hat{F}(a)\hat{F}(a)N = N \sum_{a \in \mathbb{F}_2^n} \hat{F}(a)^2.$$

Thus, $\sum_{a \in \mathbb{F}_2^n} \hat{F}(a)^2 = \frac{\langle F, F \rangle}{N}$. However, $\langle F, F \rangle$ is also N, because, as in the proof of claim 3, it is the sum of $F(a)^2 = 1$ over all $a \in \mathbb{F}_2^n$. We can therefore conclude that $\sum_{a \in \mathbb{F}_2^n} \hat{F}(a)^2 = 1$, and so

$$1 - 2q = \sum_{a \in \mathbb{F}_2^n} \hat{F}(a)^3 \leq (1 - 2\epsilon) \sum_{a \in \mathbb{F}_2^n} \hat{F}(a)^2 = 1 - 2\epsilon$$

$$2q \geq 2\epsilon$$

$$q \geq \epsilon$$

■

2 Edge Sampling

In the remainder of the lecture, we briefly show a lower bound on the time complexity of edge sampling as well as a modification of the solution to problem 1 in problem set 3 whose query complexity more closely matches our lower bound.

2.1 Introduction

2.1.1 Model

We are working in the adjacency list model, where we have a graph G with n node and m edges, and can do the following:

- Sample a node in G uniformly at random
- Given a node u, query its degree $\deg(u)$
- Given a node u and an index j, query the jth neighbor of u; this query fails if $\deg(u) < j$

2.1.2 Goal

We seek an algorithm to sample edges in G ϵ-close to uniformly at random, which we define as obeying the condition that, if the probability of sampling an edge e in G is $p(e)$, for any two edges e_1, e_2 in G,

$$1 - \epsilon \leq \frac{p(e_1)}{p(e_2)} \leq 1 + \epsilon.$$

In problem set 3, we saw how to do this in $O(\frac{n}{\sqrt{m\epsilon}})$ queries. Is it possible to do better?

2.2 Lower Bound by Construction

Theorem 4 *An algorithm that samples edges ϵ-close to uniformly at random needs $\Omega(\frac{n}{\sqrt{m}})$ queries.*

Definition 6 *A* clique *is a graph where there is an edge between any two distinct nodes.*

Proof Consider a graph G constructed as follows. We choose $\sqrt{2m}$ nodes and make them into a clique. Other than that, we add no edges (so that the number of edges is m as expected). Uniformly sampling an edge requires us to sample an edge to begin with, which requires us to find one of the $\sqrt{2m}$ nodes in the clique, as no other nodes have edges. A random node has a $\frac{\sqrt{2m}}{n}$ probability of being in the clique, so we need $\Omega(\frac{n}{\sqrt{m}})$ queries to find a node in the clique. We can thus give $\Omega(\frac{n}{\sqrt{m}})$ as a lower bound on the query complexity of a uniform edge sampler. $\blacksquare$

2.3 Approaching the Lower Bound

This lower bound differs from the complexity we achieved in the problem set by a multiplicative factor of $\sqrt{\frac{1}{\epsilon}}$. Clearly, we shouldn't be able to remove the dependence on ϵ completely, but we can ask if it is possible to move it to an additive factor, and the answer is yes.

Theorem 5 *There is an algorithm for sampling edges ϵ-close to uniformly at random that uses $O(\frac{n}{\sqrt{m}} + \frac{1}{\epsilon^2})$ queries*

This algorithm is in fact a modification of the algorithm from the problem set, so we will first briefly remind ourselves of that algorithm, omitting the details of the analysis.

2.3.1 The Original Algorithm, Briefly

The idea of the original algorithm was to choose some degree threshold Δ, considering nodes with degree at most Δ to be "light", and nodes with degree greater than Δ to be "heavy". We then say that an edge (a, b) is light if a is light and heavy if a is heavy.

We first have the following procedure for sampling light edges:

1. Sample a node u in G uniformly at random.
2. Fail if u is heavy.
3. Sample a random index k in $[1, \Delta]$.
4. Query v, the kth neighbor of u, failing if u does not have k neighbors.
5. Return the edge (u, v).

The probability of a particular light edge being sampled by this procedure is $\frac{1}{n\Delta}$ because of steps 1 and 3. The idea used in steps 3 and 4 where we essentially only pass for a given u with probability proportional to its degree is known as rejection sampling, and is what allowed us to obtain a uniform probability of sampling each light edge.

We then have the following procedure for sampling heavy edges:

4. Do the same steps 1 to 4 from the previous page to sample a light edge (u, v).

5. Fail if v is light.

6. Sample a random neighbor w of v.

7. Return the edge (v, w).

As seen in the problem set, a heavy edge (v, w) will be chosen with probability $\frac{1}{n\Delta}$ if all of v's neighbors are light, and by choosing $\Delta = \sqrt{\frac{m}{\epsilon}}$, you can argue that at most $\frac{1}{\epsilon}$ of any heavy node's neighbors are heavy (since there are so few heavy nodes), so the probability of (v, w) being chosen is at least $(1-\epsilon)\frac{1}{n\Delta}$, which means this procedure samples heavy edges ϵ-close to uniform.

Because both procedures output a given edge with probability close to $\frac{1}{n\Delta}$, we can combine them by simply choosing one of them at random to run to obtain a procedure for sampling edges in G ϵ-close to uniform at random.

The runtime of this algorithm then comes from the fact that it succeeds with probability

$$O\left(\frac{m}{n\Delta}\right) = O\left(\epsilon\frac{\sqrt{m}}{n}\right)$$

meaning that we need to repeat the procedure $O(\frac{n}{\epsilon\sqrt{m}})$ times to have an $\Omega(1)$ probability of finding an edge.

2.3.2 The Modification

We weren't able to go into detail about the modification in lecture, but we describe the idea here.

In order to improve the runtime, we want to make Δ smaller. The reason for Δ's size comes from the heavy edges – Δ needs to be $\sqrt{\frac{m}{\epsilon}}$ so that the heavy nodes's neighbors are mostly light nodes, which is needed so that the heavy edges don't vary too much in their probabilities of being chosen.

We can alleviate this by using rejection sampling for the heavy edges just as we did for the light edges. We insert a new step between steps 6 and 7 of our heavy edge sampling procedure that fails with a certain probability depending on how many of v's neighbors are heavy; this then brings the heavy edges' probabilities of being sampled close to uniform, meaning that we can relax the size of Δ somewhat.

We don't cover the details of how this is done, but essentially the decreased Δ leads to the $O(\frac{n}{\sqrt{m}})$ term in the runtime, and we need to perform sampling to approximate a given v's proportion of heavy neighbors to within a something like ϵ factor, which adds the $O(\frac{1}{\epsilon^2})$ term, attaining a runtime of

$$O\left(\frac{n}{\sqrt{m}} + \frac{1}{\epsilon^2}\right)$$

as stated in theorem 5.