DSC180A FA25: Roughgarden Notes

Ciro, Darren, Ivy, Ryan

November 24, 2025

Contents

1 Lower Bounds for One-Way Communication: Disjointness, Index, and Gap-Hamming

Remark 1.1 Recall that the one-way communication complexity for disjointness is $\Omega(n)$, even for randomized protocols.

Remark 1.2 Recall that all constant error probabilities yield the same communication complexity, up to a constant factor. This is because the sucess of a protocol can be boosted though amplification via repeated trials.

Lemma 1.3 Let D be a distribution over the space of inputs (x, y) to a protocol and $\epsilon \in (0, 0.5)$ be its error. Suppose every deterministic one-way protocol P with

$$\Pr_D[P \text{ wrong on } (x, y)] \leq \epsilon$$

has CC at least k. Then every randomized one-way protocol R with two sided error at most ϵ on every input has communication cost at least k.

Proof. The idea is that we transfer randomness from the input into the protocol itself. Write P as a distribution of deterministic protocols $P_1, \ldots, P_s$. Then on average the protocol is wrong at least epsilon times, completing the proof. $\square$

1.1 Index

Definition 1.4 The index problem is where Alice has a list of n bits and Bob has a $\log_2(n)$ bit index i and they wish to compute the ith bit of Alice's input.

Theorem 1.5 The randomized one-way communication complexity of index is $\Omega(n)$.

Proof. Let D be the uniform distribution of inputs. We show every deterministic one-way protocol that uses cn bits has error at least $1/8$, where c is some small constant. By Lemma 2.3, this implies that every randomized protocol has error at least $1/8$ on some input.

Fix a deterministic protocol P that uses cn bits. There are 2^{cn} distinct messages that Alice sends. We show that

$$\Pr[P \text{ is incorrect}] = \frac{d_H(x, a(z))}{n}$$

where d_H is the Hamming distance and $a(z)$ is Bob's guess; an **answer vector**. Think of each $a(z)$ as a ball of radius $n/4$ in the Hamming cube. Since there are only 2^{cn} balls of small radii, the union of all the balls is less than half of the Hamming cube. That is, there are at least 2^{n-1} bad inputs.

Fix some answer vector a. The number of inputs x with hamming distance at most $n/4$ from a is

$$1 + \binom{n}{1} + \binom{n}{2} + \binom{n}{n/4}$$

where we choose how many bits to change from 0 (a itself) to $n/4$. Recall that

$$\binom{n}{k} \leq \left(\frac{en}{k}\right)^k$$

so the above is bounded by $2^{.861n} \leq 2^{n-1}$ for sufficiently large n. $\square$

1.2 Gap-Hamming

Remark 1.6 Our current goal is to prove that every streaming algorithm that computes a $(1 \pm \epsilon)$ approximation of F_0 or F_2 needs $\Omega(\epsilon^{-2})$ space. We do this using reductions.

Definition 1.7 Let x, y be inputs interpreted as streams, i.e. vectors of numbers. Then the hamming distance $d_H(x, y) = 2F_0 - |x| - |y|$. So a one-way protocol that computer F_0 with CC c yields a communiction protocol that computes d_H with CC $c + \log_2 n$. Thus a $(1 \pm {}^1/\!\sqrt{n})$ approximation of F_0 yields a protocol that estimates d_H up to $2\sqrt{n}$ additive error with $\log_2 n$ extra communication.

Theorem 1.8 The randomized one-way communication complexity of gap-hamming is $\Omega(n)$.

Proof. Consider an input to index, where Alice has n bits and Bob $\log_2 n$ bits an index i. Suppose n is odd and large. Alice and Bob can generate wihtout communicating, an input (x', y') to gap-hamming using public randomness. $\square$

2

Remark 2.1 There are lots of ways to use CC to prove lower bounds for data structures. Here we limit ourselves to the approximate nearest neighbor problem with static (unmodifiable and only queriable) data structures.

2.1 Introducing Nearest Neighbors (NN)

Definition 2.2 The **nearest neighbor problem** takes in a set S of n points in a metric space (X, ℓ). Most commonly this is $\mathbb{R}^d$ with the ℓ_2 norm. Here we use the **hamming cube** $X = \{0,1\}^d$ and ℓ is the Hamming distance. Let d be large, say $d = \sqrt{n}$.

The goal is to build a data structure D (as a function of $S \subseteq X$) that prepares for all possible nearest neighbor queries. A query is a poiny $q \in X$, and the end goal is to return $p^* \in X$ that minimizes $\ell(p, q)$ over all $p \in S$.

Remark 2.3 We use the Hamming cube because it is easier to map into communication complexity, though the key high-level ideas are the same for analyzing Euclidean and other metric spaces.

Lemma 2.4 If computing $\ell(p, q)$ take $O(d)$ time, then the brute force algorithm takes $O(dn)$ time. Alternatively, precomputing the answer to all possible queries takes $\Theta(d2^d)$ space. The exact NN problem thus suffers from an exponential-size data structure in terms of d. This is known as the **curse of dimensionality**.

2.2 Approximate Nearest Neighbors (ANN)

Definition 2.5 The $(1+\epsilon)$-approximate NN, where we only need to return p where $\ell(p, q) \leq (1+\epsilon)\ell(p^*, q)$, we can do much better. Let $\epsilon = 1$ or 2, so it is not too small but still practically relevant.

The high-level idea is to hash nearby points into the same bucket, then precompute the answer for each bucket. Therefore the key challenge is to make sure nearby points hash to the same bucket.

2.2.1 Decision ANN

Definition 2.6 Consider the **decision ANN** version, where given $L \in \{0, 1, \ldots, f\}$, we wish to distinguish between

1. There exists a point $p \in S$ with $\ell(p, q) \leq L$.
2. $\ell(p, q) \geq (1+\epsilon)L$ for every $p \in S$.

If neither applies, both answers are correct.

Note that if $|S| = 1$, this is oddly similar to equality: Equality is the same as deciding between hamming distance 0 or not. The public coin protocol where Alice takes the inner product mod 2 of x with r_1 and r_2 (2 bits) and sends it to Bob always accepts $x = y$ and accepts $x \neq y$ with probability 1/4.
It is also similar to gap-hamming, which we showed was hard in Chapter 2 using a reduction from index.

Definition 2.7 Consider the CC problem where Alice and Bob want to decide if $\ell_H(x, y)$ between inputs $x, y \in \{0, 1\}^d$ is at most L or at least $(1 + \epsilon)L$. Call this **ϵ-Gap Hamming**. Define the following protocol:

1. Alice and Bob create $s = \Theta(\epsilon^{-2} \log(?))$ random strings $R = \{r_1, \ldots, r_s\} \in \{0, 1\}^d$ where $d = \theta(\epsilon^{-2})$ from public coins. Each entry is independent but not uniform: there is a $1/2L$ probability of each entry in r_i being equal to 1.

2. Alice sends s inner products of r_i with x. This is the "hash value" $h_R(x) \in \{0, 1\}^s$.

3. Bob accepts if any only if the Hamming distance between $h_R(y)$ and $h_R(x)$ differs by some small number of bits.

We defer the analysis of the small number of bits to the book. The key idea is as follows:

1. When selecting a string r_i, we are choosing a coordinate to be relevant (be a 1, and thus contribute to the inner product later) with probabiliy $1/L$. Consider it irrelevant otherwise. Think of this as selecting relevant coordinates for this r_i hash.

2. The second stage, the inner product, is a modified equality with these coordinates only. We see that

$$\Pr_j[\langle r_j, x\rangle \not\equiv \langle r_j, y\rangle \mod 2] = \frac{1}{2}\left(\left(1 - \left(1 - \frac{1}{L}\right)^{l_H(x,y)}\right)\right).$$

This results in CC $\Theta(\epsilon^{-2})$.

2.2.2 Data Structure Decision ANN

Definition 2.8 We now convert the above algorithm into a data strcuture for the $(1 + \epsilon)$ decision ANN problem.

1. Given a point set P of n points in $\{0, 1\}^d$, choose a set of R of $s = \Theta(\epsilon^{-2} \log n)$ as above.

2. Define $h_R : \{0, 1\}^d \to \{0, 1\}^s$ by setting the jth coordinate to $\langle r_j, x\rangle$ mod 2.

3. Make a table with $2^s = n^{\Theta(\epsilon^{-2})}$ buckets, indexed by s-bit strings.

4. For all $p \in P$, insert p into every bucket b which $\ell(h_R(p), b) \leq (t + {}^1/_2 h(\epsilon))s$ where $h(\epsilon) = \frac{1}{2\epsilon^2}(1 - \epsilon^{-\epsilon})$.

This data structure has probability at least $1 - \frac{1}{n}$ to give the correct answer in $n^{O(\epsilon^{-2})}$ space.

Remark 2.9 We will show later that with this level of accuracy this space is optimal. Note that it is possible to do better by inreasing the query time.

Note that in the real problem we do not have this value L. We would want it to be the actual NN distance of a query point q (since L counts the number of differences / coordinates checked), but this can change from one q to another. Furthermore, the data strcuture may be wrong on as many as $2^d/n$ queries. Knowing the coin flips allow an adversary to exhibit queries that will be wrong.

Lemma 2.10 Fix 1: Make d copies of the data structure for each value of L^2. Answering takes $O(\log d)$ lookups. Space blows up by $\Theta(d \log d)$. Query time blows up by $\Theta(\log \log d)$.

Lemma 2.11 Fix 2: Make $\Theta(d)$ copies and take the majority vote. This is wrong in at most inverse exponential of d. Here not even an adversary who knows the coin flips knows whether a particular query will be incorrect ahead of time. This blows up space and query time by $\Theta(d)$.

Lemma 2.12 Fix 3: Make $\Theta(d)$ copies and choose uniformly randomly one of the copies to answer. The space blows up by $\Theta(d)$ but query time is unaffected. The probability of a correct answer is at least $1 - \Theta(\frac{1}{n})$.

Combining the three lemmas:

- Space: $O(d^2 \log d) \cdot n^{\Theta(\epsilon^{-2})}$
- Query time: $O(\epsilon^{-2} d \log n \log \log d)$.

That is, polynomial space for logarithmic query time.

3 The Cell Probe Model

We start by motivating the cell probe model. The goal is to create a database D to answer Q queries, known beforehand. The encoding scheme must work for all possible databases: In ANN, P is the database (unknown beforehand) and Q are elements of $\{0,1\}^d$ (known). Another example is set membership.
A parameter of the cell probe model is **word size** w. The design space if the number of ways to encode D as s cells of w bits each. We say s is the space used in the encoding. It must correctly answer all $q \in Q$; it does this by reading cells one at a time, giving w bits each. The query time of this algorithm is the max, over all D and all $q \in Q$, number of accesses to answer the queries.

Remark 3.1 w is typically large so a single element of the database can be stored in a cell, and not much larger than this.

For ANN, take w to be polynomial in $\max\{d, \log_2(n)\}$. The previous construction solves ANN cell probe with space $n^{\Theta(\epsilon^{-2})}$ and query time 1. We now show a matching lower bound in the cell-prove model; thus constant query time can only be achieved by encodings that us $n^{\Omega(\epsilon^{-2})}$ space.

3.0.1 Query Database

We start with a contrived problem and build connections later.

DEFINITION 3.2 Consider the **query database** problem. Alice has a query q and Bob gets database D. We wish to compute the answer to q on the database D.

LEMMA 3.3 Let there be a cell-probe encoding of query database with word size w, space s, and query time t. Then there is a CC protocol with CC at most

$$\underbrace{t\log_2(s)}_{Alice} + \underbrace{tw}_{Bob}$$

Alice simulates the query-answering algorithm, sending $\log_2(s)$ to Bob for every cell requested. Bob sends back w bits to describe the contents of the selected cell. They need to go back and forth only t times (since the query time is also the number of $\Theta(1)$ queries, i.e. t queries).

LEMMA 3.4 (**RICHNESS LEMMA**) Let $f : X \times Y \to \{0,1\}$ be a Boolean function with corresponding $X \times Y$ boolean matrix $M(f)$. Assume that

1. $M(f)$ has at least v columns that each have at least u 1-inputs.
2. There is a deterministic protocol that computes f in which Alice and Bob send a and b bits respectively.

Then $M(f)$ has a 1-rectangle $A \times B$ with $|A| \geq u/2^a$ and $|B| \geq u/2^{a+b}$.

THEOREM 3.5 For every $\epsilon, \delta > 0$ and sufficiently large n, in every CC protocol that solves $(\frac{1}{\epsilon^2}, n)$-disjointness with a universe of size $2n$, either

1. Alice sends at least $\delta/\epsilon^2 \log_2 n$ bits, or
2. Bob sends at least $n^{1-2\delta}$ bits.

THEOREM 3.6 $(\frac{1}{\epsilon^2}, n)$-disjointness reduces to the query database problem for the decision problem of ANN with $(1+\epsilon)$.

THEOREM 3.7 Every data structure for the decision version of $(1+\epsilon)$-ANN with query time $t = \Theta(1)$ and word size $w = O(n^{1-\delta})$ for $\delta > 0$ uses space $s = n^{\Omega(\epsilon^{-2})}$.

The proof is quite involved and is deferred for later.

Term Index