Lecture 1

Course Introduction

Scribe: Tore Frederiksen

1.1 Overview

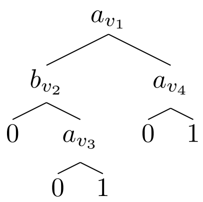
Assume we are in a situation where several people need to work together in order to solve a problem they cannot solve on their own. The field of communication complexity is then concerned with how much communication (bits) needs to be communicated between the parties in order to solve the problem. For simplicity we generally assume that there are only two parties who need to communicate, we call these two parties for Alice and Bob. As an example assume that Alice gets $x \in \{0,1\}^n$ and Bob gets $y \in \{0,1\}^n$. They now wish to determine if $x = y$. The question is now how many bits they need to communicate in the worst case in order to find the answer to this question with certainty. To answer this question let us first consider the trivial solution in order to get an upper bound. Clearly, if Alice sends the whole bitstring x to Bob, then Bob will be able to check if it is equal to his bitstring y and then return 1 if that is the case and 0 otherwise. For this approach we clearly need to communicate $n + 1$ bits. Since x contains n bits and Bob returns a single bit. Now the question is if we can do better? Maybe Alice can compress the bitstring x and send the compression to Bob who then expands it and checks its equality with his own bitstring y? Unfortunately this is not possible. To see this notice that there are 2^n possible values for x. Assume that we compress x by just a single bit, that is, to a string of size $n - 1$. This means that there are 2^{n-1} possible compressed strings. Thus there are at least two of the 2^n strings that must compress to the same 2^{n-1} string. Now, how can Bob know which of these two possible strings is Alices original string? He can't. So if we compress by even a single bit then Bob will be wrong in some cases. From this observation it turns out that it is actually not possible to solve the problem with less than the trivial $n + 1$ bits of communication.

Looking at this example we can now consider how to define a communication protocol for computing a function such that it is always clear whose turn it is, how many bits are sent each time and so on. It turns out that this can be specified elegantly using a tree as in the following definition:

Definition 1 (Protocol Tree)**.** A protocol for computing a function $f : X \times Y \to Z$ is a labeled binary tree with the following properties:

- The leaves are labeled with $z \in Z$.
- Internal nodes are labeled with either $a_v : X \to \{0,1\}$ or $b_v : Y \to \{0,1\}$. Notice that the subscript $_v$ is a unique identifier for a given node. Also notice that an inner node is actually a map, mapping our element x or y (depending on whether the node is of type a_v or b_v) to 0 or 1. If our element is mapped to 0 we go to the left child, and if it is mapped to 1 we go to the right child.

Example 2. We let $X = Y = \{1,2,3,4\}$ and $Z = \{0,1\}$ and define the following communication tree along with a map for each node:



a_{v_1}:

x	$a_{v_1}(x)$
1	0
2	0
3	1
4	1

b_{v_2}:

y	$b_{v_2}(y)$
1	1
2	0
3	1
4	1

a_{v_3}:

x	$a_{v_3}(x)$
1	0
2	1
3	1
4	0

a_{v_4}:

x	$a_{v_4}(x)$
1	1
2	0
3	1
4	0

From this example we notice that we make quite a few assumptions in order for the communication tree to specify a communication protocol for a function. These assumptions are:

- The output of the protocol is determined only through the communication sent.
- The bits sent are always received and never corrupted.
- The players deterministically choose their messages.
- Who sends the next bit is determined deterministically only by the previous communication.

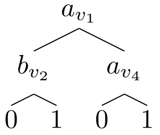
Again there are also some assumption we **don't** make:

- We do not make any assumptions about how long Alice and Bob have to decide on what message to send next.
- We do not make any assumption on the computational resources (e.g., CPU power, memory, disk space) available to Alice and Bob.

The lack of these assumptions makes the use of the communication tree as specification for communication protocols very unrealistic. However, it also makes the communication model very general. This means that the communication tree will give the ultimate lower bound for communication needed to decide the answer to a function using a specific protocol.

Now given a communication tree for a specific protocol we define the *cost a of the protocol* to be the depth of the tree, which is equivalent to the amount of bits sent in the worst case over all possible inputs. Similarly, we define the *cost of the protocol on a particular instance* of $x \in X$ and $y \in Y$ to be the length of the path from the root to the correct leaf in the communication tree.

Example 3. Consider the communication tree from our last example. Does this communication tree describe an optimal protocol for the function we wish compute? It turns out that it does not. To see this notice that node v_3 is actually not needed, since if $x = 1$ then the function will always output 0 no matter what y is. So if we change node v_1 and v_4 correctly we can eliminate node v_3 as it will always output 1. Thus we can cut off one layer in the tree and reduce the cost of the protocol by 1. So we can construct a communication tree for the same function as follows:



	x	$a_{v_1}(x)$		y	$b_{v_2}(y)$		x	$a_{v_4}(x)$
	1	1		1	1		1	0
a_{v_1}:	2	0	b_{v_2}:	2	0	a_{v_4}:	2	0
	3	1		3	1		3	1
	4	1		4	1		4	0

1.2 Fooling Sets

We start this section by formally defining what we mean by communication complexity for computing a given function.

Definition 4 (Communication Complexity)**.** For a function $f : X \times Y \to Z$ the deterministic communication complexity, $D(f)$, is defined as

$$D(f) := \min\{\text{cost}(P) : P \text{ computes } f\}$$

where P is any protocol.

Now assuming that the function we wish to compute is binary, we can then describe all possible outputs of the function using a binary matrix in the following manner:

Definition 5 (Communication Matrix)**.** We call the communication matrix of a binary function $f : X \times Y \to \{0, 1\}$ for M_f and define it to be a binary matrix of size $|X| \times |Y|$ where the rows corresponds to the input from Alice and the columns corresponds to the input from Bob. The value of entry (x, y) is then the result of function f on input $f(x, y)$, or more formally

$$M_f[x, y] := f(x, y)$$

Using such a matrix we can again define smaller submatrices with special properties that will make complexity analysis easier.

Definition 6 (Combinatorial Rectangle)**.** We define $R \subseteq X \times Y$ to be a combinatorial rectangle of a communication matrix M_f for function $f : X \times Y \to \{0, 1\}$ if $R = A \times B$ for some $A \subseteq X$ and $B \subseteq Y$.

Definition 7 (Mono-chromatic)**.** A combinatorial rectangle $R \subseteq X \times Y$ for function $f : X \times Y \to \{0, 1\}$ is called mono-chromatic if $M_f[x, y] = M_f[x', y']\ \forall (x, y), (x', y') \in R$.

This means that a rectangle is mono-chromatic if all of its entries are the same. So from these definitions we notice the following facts:

Fact 8. A protocol for a function f partitions the corresponding communication matrix M_f into mono-chromatic rectangles, with each rectangle corresponding to a leaf in the communication tree for this protocol.

Fact 9. If any partition of a communication matrix M_f requires t or more mono-chromatic rectangles then $D(f) \geq \log t$.

From these observations we get the following proposition:

Proposition 10. *A rectangle $R \subseteq X \times Y$ is a combinatorial rectangle if and only if $(x_1, y_1) \in R$ and $(x_2, y_2) \in R \Rightarrow (x_1, y_2) \in R \wedge (x_2, y_1) \in R$*

Using these informations we can now define a fooling set:

Definition 11 (Fooling set)**.** A set $S \subseteq X \times Y$ is a fooling set for a function $f : X \times Y \to Z$ if there exists a $z \in \{0, 1\}$ such that $f(x, y) = z$ for all $(x, y) \in S$ and for all $(x_1, y_1), (x_2, y_2) \in S$ then either $f(x_1, x_2) \neq z$ or $f(x_2, x_1) \neq z$.

From this definition we get the following interesting fact:

Fact 12. If a function $f : X \times Y \to Z$ has a fooling set of size t then $D(f) \geq \log t$.

Example 13. As an example of this fact let us consider a communication matrix that is equal to the identity matrix, that is $M_f = I$. We notice that this is in fact the communication matrix of the equality function (EQ). That is, the function $f : X \times Y \to \{0,1\}$ where $f(x,y) = 1 \iff x = y$. We also notice that there exists a fooling set $S := \{\alpha, \alpha\}\ \forall \alpha \in X = Y$. To see this consider a $\beta \in X = Y$ with $\beta \neq \alpha$ and notice that $f(\alpha,\alpha) = 1$ and $f(\beta,\beta) = 1$, but $f(\alpha,\beta) = 0$ and $f(\beta,\alpha) = 0$ as all entries in M_f are 0 except for the cases when $x = y$. This means that S is a 2^n size fooling set and thus that $D(\text{EQ}) = n + 1$.

From this example we notice the following fact:

Fact 14. Assume, without loss of generality, that $|X| \leq |Y|$ for a function $f : X \times Y \to Z$ then $D(f) \leq \log|X| + \log|Z|$.

Example 15. Let us consider another example. In this example we consider the median function (MED). Alice gets $x \subseteq \{1, ..., n\}$ and Bob gets $y \subseteq \{1, ..., n\}$. We then wish to compute the function $f(x,y) = \text{MED}(x,y) :=$ median of the multi set $x \cup y$.

A trivial solution to this problem might again be for Alice to send the entire set x for which we need n bits[1], and then let Bob send back the value of the median element using $\log n$ bits. Thus, using this approach the communication needed is $n + \log n$.

Let us now consider another approach where we try to do a binary search. Our approach is to, after each communication, be able to eliminate elements as possible answers to the function $\text{MED}(x,y)$ until only one element is left, which must then be the correct answer. We do so by maintaining a range $[i;j]$ with $i \geq 0$ and $j \leq n$ with possible answers. What we actually do can most easily be described with the pseudocode in algorithm 1. Using this approach we see that the upper bound of communication needed to compute MED

Algorithm 1 Binary search for result of $\text{MED}(x,y)$

```
$i := 1$
$j := n$
Alice and Bob exchange $|X|$ and $|Y|$
$t := |X| + |Y|$
**repeat**
    $k := (i + j)/2$
    Alice sends amount of elements in $x < k$ to Bob.
    Bob sends amount of elements in $y < k$ to Alice.
    **if** $t/2$ elements have value $< k$ **then**
        $j := k$
    **else**
        $i := k$
    **end if**
**until** $i = j$
**return** $i$
```

is $\mathcal{O}(\log^2 n)$, as we have $\log n$ iterations, where we send $\mathcal{O}(\log n)$ bits in each of these iterations.

Now the trivial lower bound is $\Omega(\log n)$ since there are n possible outputs and this is the absolutely minimum amount of information needed to be able to decide on any one of these values as the correct answer.

Some theory shows that the lower bound is in fact also the upper bound! We can solve MED by using no more than $\mathcal{O}(\log n)$ bits of communication in the worst case. (Note: this upper bound is nontrivial).

[1] This is so since we can encode Alices subset as a binary string of size n where a 1 in position i of this string implies that element i is in the subset and a 0 implies that it is not.

Lecture 2

Rank lower bound and covers

Scribe: Jesper Sindahl Nielsen

2.1 Recap

Last time we defined a protocol tree and discussed how it relates to the communication matrix. We also introduced combinatorial rectangles and mono-chromatic combinatorial rectangles. We looked at the problem EQ where Alice and Bob both get an n-bit string and have to compute whether they are the same and it turned out that $D(f) \geq n+1$, this was proved using the "fooling sets" technique.

2.2 Rectangle size lower bound

The general idea is to prove that the size of every monochromatic rectangle is "small"; hence, we need lots of rectangles to cover the entire matrix.

Proposition 16. *Let μ be a probability distribution of $X \times Y$. If any f-monochromatic rectangle R has meassure $\mu(R) \leq \delta$ then $D(f) \geq \log 1/\delta$*

Proof. We know that $\mu(M_f) = 1$ so there must be at least $1/\delta$ rectangles in any f-monochromatic partition of M_f. □

This is in fact a generalization of the fooling-sets technique, as illustrated by the following example. Consider the EQ problem again. Now define

$$\mu(x, y) = \begin{cases} 2^{-n} & \text{if } x = y \\ 0 & \text{if } x \neq y \end{cases}$$

It should now be clear that for any monochromatic rectangle R we have $\mu(R) \leq 2^{-n}$ which yields $D(f) = \log 1/2^{-n} = n$.

2.3 Rank lower bound

The rank lower bound was invented by Kurt Mehlhorn and Erik Meineche Schmidt. This result relies on linear algebra. A large portion of the techniques for proving lower bounds in communication complexity relies on knowledge from linear algebra and this is only scratching the surface. The rank lower bound works by giving a lower bound on the number of monochromatic rectangles in any partition of $X \times Y$.

Definition 17. The rank of a matrix A is the number of linearly independent rows in A.

Theorem 18. *For any function $f\colon X \times Y \to \{0,1\}$ the communication complexity $D(f) \geq \log(\mathit{rank}(f))$.*

Proof. Fix protocol P for f. Look at the leaves where the output is 1. Let $L_1 = \{\text{leaves in P where output is 1}\}$. For each $\ell \in L_1$ define

$$M_\ell[x,y] := \begin{cases} 1 & \text{if } (x,y) \in R_\ell \\ 0 & \text{if } (x,y) \notin R_\ell \end{cases}$$

where R_ℓ is the rectangle containing all inputs that reach ℓ. Now we can observe that the sum of the M_ℓs are exactly M_f because every entry (x,y) where $f(x,y) = 1$ is in a single matrix M_ℓ and if the entry (x,y) is 0 then it will be 0 in all the M_ℓs. We get that

$$M_f = \sum_{\ell \in L_1} M_\ell$$

And because of some nice properties of the rank function we get

$$\text{rank}(M_f) \leq \sum_{\ell \in L_1} \text{rank}(M_\ell)$$

If we look at the M_ℓ matrices we see that they all have rank 1

$$\sum_{\ell \in L_1} \text{rank}(M_\ell) = |L_1| \leq 2^{D(f)}$$

In the end we get $D(f) \geq \log \text{rank}(f)$ □

As an example of the proof, consider the following:

$$M_f = \begin{pmatrix} 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} + \begin{pmatrix} 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} + \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

The matrices on the right hand side are the M_ℓs and we can see that they all have rank 1.

As an additional note, the proof gave a lower bound on the number of rectangles that have the number 1 in them. We can do the same proof for the 0-rectangles which can be used to prove that

$$D(f) \geq \log(2\text{rank}(f) - 1)$$

2.4 Covers

To introduce the concept we start with a couple of examples.

Example 19. Consider the matrix in Figure **??**. Can this partitioning be represented by some protocol tree?

No. Let $\mathcal{P}$ be any protocol representing f. Suppose (wlog) that Alice sends the first message. Alice cannot send the same message for all inputs. Say the inputs are x_1, x_2, x_3 for Alice and y_1, y_2, y_3 for Bob. The message Alice sends must either be different on x_1 and x_2 or it must be different on x_1 and x_3. In the first case the upper left rectangle in the figure will not be induced by $\mathcal{P}$ and in the second case the lower right rectangle in the figure will not be induced by $\mathcal{P}$. A similar argument can be made if Bob sends the first message.

Example 20. Consider the matrix:

$$M_{f_2} = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{pmatrix}$$

How many monochromatic rectangles does it take to partition M_{f_2}?

How many monochromatic rectangles does it takes to cover M_{f_2}?

Note that fooling set arguments give lower bounds on the *latter*—elements of the fooling set must be in different monochromatic rectangles. Hence any *cover* of M_{f_2} by monochromatic rectangles must include at least the number of elements in the fooling set. The answers to the questions are 5 and 4 respectively.

Definition 21. Let $f\colon X \times Y \to \{0,1\}$ be a function.

1. *Protocol partition number* $C^P(f)$ is defined as the smallest number of leaves in a protocol tree.
2. *The partition number* $C^D(f)$ is defined as the least number of monochromatic rectangles in a partition of $X \times Y$.
3. *Cover number* $C(f)$ is defined as the smallest number of monochromatic rectangles needed to cover $X \times Y$.
4. For $z \in \{0,1\}$, $C^z(f)$ is the number of monochromatic rectangles needed to cover the z-inputs of f.

We now have some basic facts.

Fact 22. For all functions $f : X \times Y \to \{0,1\}$, we have

1. $C(f) = C^0(f) + C^1(f)$
2. $C(f) \leq C^D(f) \leq C^P(f) \leq 2^{D(f)}$

If we translate the techniques we have seen to the definitions we get the following.

1. Rank lower bound corresponds to $C^D(f) \geq 2\text{rank}(f) - 1$
2. Let μ be a probability distribution on z-inputs. If $\mu(R) \leq \delta$ then $C^z(f) \leq 1/\delta$.

2.4.1 A note on nondeterministic communication complexity

One can define nondeterministic communication complexity in terms of $C^z(f)$.

Definition 23. Nondeterministic communication complexity $N^1 := \log C^1(f)$ Co-nondeterministic communication complexity $N^0 := \log C^0(f)$. We also define $N(f) = \log C(f)$.

NP is the class of decision problems whose "yes" instances (i.e., $x \in L$) have proofs that can be verified in polynomial time. Similarly, coNP is the class of decision problems whose "no" instances (i.e., $x \notin L$) have proofs that can be verified in polynomial time.

Nondeterministic and co-nondeterministic communication complexity can be defined in a similar matter. In this setting, nondeterministic communication complexity considers problems where a prover sees x and y and sends a "proof" that $f(x,y) = 1$ to both Alice and Bob. The players then talk to each other and verify that indeed $f(x,y) = 1$. The cost is then defined as the length of the "proof" and the communication between Alice and Bob. Conondeterministic communication complexity is defined similarly, except the prover supplies a proof that $f(x,y) = 0$.

Example 24. We look at the EQ problem again.

$$EQ(x,y) = \begin{cases} 1 & \text{if } x = y \\ 0 & \text{if } x \neq y \end{cases}$$

M_{EQ} is then the identity matrix. The fooling set technique gave us that $C^1(f) \geq 2^n$ which gives $N^1(f) \geq n$.

In the co-nondeterministic proof the prover sends bit i where $x_i \neq y_i$. Alice and Bob then both send their i'th bit and verify that they are distinct. In total this requires $\log n + 2$ bits.

It is also possible to prove that for all $f\colon X \times Y \to \{0,1\}$ and $z \in \{0,1\}$, $D(f) \leq C^z(f) + 1$. Combining the lower bound and upper bound there might be a gap of exponential size between deteministic and nondeterministic communication complexity but no more that this.

Lemma 25. *For all $f\colon X \times Y \to \{0,1\}$ and $z \in \{0,1\}$ $D(f) \leq C^z(f) + 1$.*

Proof. Recall that $\log C^z(f) \leq \log C(f) \leq D(f)$.

Fix a z-covering of M_f: $R_1, R_2, \ldots, R_{C^z(f)}$. Say Alice and Bob both know the covering of M_f then Alice sends 1 for each R_i if x is consistent with R_i and 0 otherwise. Bob then checks all rectangles consistent with Alice's input. If there exists a rectangle also consistent with Bob's input then $f(x,y) = 1$ otherwise $f(x,y) = 0$. □

A final fact.

Lemma 26. $\log C^P(f) \leq D(f) \leq 3logC^P(f)$

Proof. Exercise. Hint: Rebalance an unbalanced tree. □

Today we also encountered an open problem. Is $D(f) = O(\log C^D(f))$? It is known that $D(f) = O(\log^2 C^D(f))$.

Lecture 3

Probability Theory + Nondeterministic Communication

Scribe: Navid Talebanfard

3.1 Probability Theory

Probability theory plays an important role in modern computer science, e.g. to model uncertainty in the incoming data or to study a whole other model of computation when the computers can choose their path according to the outcome of coin flips. In either case significant results have been obtained by employing only a small fraction of this vast theory. In this lecture we review this fraction which mainly consists of a few basic definitions and simple facts.

Definition 27. Let Ω be a non-empty finite set. A *probability distribution* on Ω is a function $\Pr : \Omega \to [0, 1]$ such that $\sum_{\omega \in \Omega} \Pr[w] = 1$. Then Ω is said to be the *sample space* of this function.

Example 28. Probability distribution generated by rolling a fair die could be represented by $\Omega = \{1, 2, 3, 4, 5, 6\}$ and $\Pr[\omega] = \frac{1}{6}$ for all $\omega \in \Omega$. In general for any Ω, if $\Pr[\omega] = \frac{1}{|\Omega|}$ for all $\omega \in \Omega$, the distribution is called *uniform.*

Definition 29. An *event* is a subset $A \subseteq \Omega$. The probability of the event A is then defined as $\Pr[A] = \sum_{\omega \in A} \Pr[\omega]$.

Example 30. Let $\Omega = \{1, 2, \ldots, 10\}$ and consider the uniform distribution on it. We define the following events: A is the event that a randomly picked number from Ω is odd, B is the event that this random number is a bigger than 5, and C is the event that this number is a prime. We can represent these events as follows: $A = \{1, 3, 5, 7, 9\}$, $B = \{6, 7, 8, 9, 10\}$ and $C = \{2, 3, 5, 7\}$. Now we have $\Pr[A] = \Pr[C] = \frac{1}{2}$ and $\Pr[B] = \frac{4}{10}$.

The following properties are immediate from the definition of the probability distribution.

Fact 31. Let Ω be a non-empty finite set and consider any probability distribution on it. We have

1. For any even A, $\Pr[\overline{A}] = 1 - \Pr[A]$.
2. (*Inclusion-Exclusion Principle*) For any two event A and B, $\Pr[A \cup B] = \Pr[A] + \Pr[B] - \Pr[A \cap B]$.
3. If $A \subseteq B$ then $\Pr[A] \leq \Pr[B]$.
4. (*Union Bound*) For any collection of events $A_1, \ldots, A_k$ we have $\Pr[\cup_{i=1}^{k} A_i] \leq \sum_{i=1}^{k} \Pr[A_i]$.

Among the above properties the union bound is of particular interest when bounding the probability of some “bad” event. Consider a randomized algorithm with a certain number of errors whose probabilities are known to us. To give an upper bound on the probability of some error happening, we can simply sum the individual properties by the union bound.

Definition 32. For any event $B \subseteq \Omega$ and $\omega \in \Omega$ the *conditional probability* of ω given B is defined as

$$\Pr[\omega|B] = \begin{cases} \frac{\Pr[\omega]}{\Pr[B]} & \omega \in B \\ 0 & \text{otherwise.} \end{cases}$$

We can then define the conditional probability of an even A given B as

$$\begin{aligned} \Pr[A|B] &= \sum_{\omega \in A} \Pr[\omega|B] \\ &= \sum_{\omega \in A \cap B} \frac{\Pr[\omega]}{\Pr[B]} \\ &= \frac{\Pr[A \cap B]}{\Pr[B]}. \end{aligned}$$

Intuitively an event A is independent from B if the occurence of B does not effect the probability of A. This notion is captured in the following definition.

Definition 33. We say two event A and B are *independent* if $\Pr[A \cap B] = \Pr[A]\Pr[B]$.

Definition 34. A *random variable* is a function $X : \Omega \to S$ for some set S. For $s \in S$ the event denoted by $X = s$ is defined as $\{\omega \in \Omega : X(\omega) = s\}$. Therefore $\Pr[X = s] = \sum_{\omega : X(\omega) = s} \Pr[\omega]$. An *indicator random variable* for an event A is defined as

$$X_A(\omega) = \begin{cases} 1 & \omega \in A \\ 0 & \text{otherwise.} \end{cases}$$

By combining random varibles we can get new ones. For instance given X and Y we can define $Z = X + Y$ by $Z(\omega) = X(\omega) + Y(\omega)$.

Definition 35. Let X and Y be random variables with ranges S and T, respectively. We say that X and Y are *independent* if for all $s \in S$ and all $t \in T$, $\Pr[X = s, Y = t] = \Pr[X = s]\Pr[Y = t]$. Random variables $X_1, \ldots, X_n$ are said to be *pairwise independent* if for any i and j, X_i and X_j are independent. They are *k-wise independent* if for any $i_1, \ldots, i_k$ and all $t_1, \ldots, t_k$, it hold that $\Pr[X_{i_1} = t_1, \ldots, X_{i_k} = t_k] = \prod_{i=j}^{k} \Pr[X_{i_j} = t_j]$. They are called *mutually independent* if they are k-wise independent for any $1 \leq k \leq n$.

Definition 36. The *expectation* of a random variable X is defined as

$$\begin{aligned} \mathbf{E}[X] &= \sum_{\omega} \Pr[\omega] X(\omega) \\ &= \sum_{s} \Pr[X = s] \cdot s. \end{aligned}$$

Proposition 37. (**Linearity of Expectation**) *For any two real-valued random variables X and Y and any $\alpha \in \mathbb{R}$ we have $\mathbf{E}[X + Y] = \mathbf{E}[X] + \mathbf{E}[Y]$ and $\mathbf{E}[\alpha X] = \alpha\,\mathbf{E}[X]$.*

Proof.

$$\begin{aligned} \mathbf{E}[X] + \mathbf{E}[Y] &= \sum_{\omega \in \Omega} \Pr[\omega] X(\omega) + \sum_{\omega \in \Omega} \Pr[\omega] Y(\omega) \\ &= \sum_{\omega \in \Omega} \Pr[\omega](X(\omega) + Y(\omega)) \\ &= \sum_{\omega \in \Omega} \Pr[\omega](X + Y)(\omega) \\ &= \mathbf{E}[X + Y] \end{aligned}$$

$$
\begin{aligned}
\alpha\,\mathbf{E}[X] &= \alpha \sum_{\omega\in\Omega} \Pr[\omega] X(\omega) \\
&= \sum_{\omega\in\Omega} \Pr[\omega](\alpha X(\omega)) \\
&= \mathbf{E}[\alpha X]
\end{aligned}
$$

□

We can use expectation to bound the probability that a random variable is at least a constant.

Proposition 38. *(***Markov's Inequality)** *Let X be real-valued random variable taking non-negative values. Let $\alpha > 0$ be any constant. Then $\Pr[X \geq \alpha] \leq \frac{\mathbf{E}[X]}{\alpha}$.*

Proof. Let I be the indicator random variable for the event $X \geq \alpha$. Then $\alpha \cdot I \leq X$. Taking expectations from both sides and noting that $\mathbf{E}[I] = \Pr[I = 1]$ proves the result. □

Example 39. There are 40 students in a class. The average weight of the students is 65 Kg. Then by Markov's inequality at most 26 students can weigh at least 100 Kg. To see this we take the uniform distribution on the set of students and let X be the random variable denoting the weight of the randomly chosen student. Then $\mathbf{E}[X] = 65$. By Markov's inequality we have $\Pr[X \geq 100] \leq \frac{65}{100}$. Therefore at most 65 % of the students can weigh at least 100 Kg, and that would be 26 of them.

To measure how far a random variable can deviate from its expectation we need variance.

Definition 40. Let X be a real-valued random variable and let $\mu = \mathbf{E}[X]$. The *variance* of X is defined as $\mathbf{Var}[X] = \mathbf{E}[(X - \mu)^2]$.

Proposition 41. (Chebyshev's Inequality) *Let X be a real-valued random variable and let $\alpha > 0$ be any constant. We have $\Pr[|X - \mathbf{E}[X]| \geq \alpha] \leq \frac{\mathbf{Var}[X]}{\alpha^2}$.*

Proof. Apply Markov's inequality on $(X - \mathbf{E}[X])^2$ and α^2 and note that $\Pr[|X - \mathbf{E}[X]| \geq \alpha] = \Pr[(X - \mathbf{E}[X])^2 \geq \alpha^2]$. □

Markov's inequality gave us a linear dependence on the constant and Chebyshev's inequality gave a quadratic one. In many cases we need more than this, we actually need exponentially small probablities in some constant. Taking more information about the random variables makes this possible.

Theorem 42. (Chernoff Bounds) *Let $X_1, \ldots, X_n$ be independent identically distributed (i.i.d) random variables such that for all i, $\Pr[X_i = 1] = p$ and $\Pr[X_i = 0] = 1 - p = q$. Let $\overline{X} = \frac{1}{n}\sum_{i=1}^{n} X_i$. For all $\epsilon > 0$ we have*

1. $\Pr[\overline{X} - p \geq \epsilon] \leq e^{\frac{-n\epsilon^2}{2q}}$
2. $\Pr[\overline{X} - p \leq -\epsilon] \leq e^{\frac{-n\epsilon^2}{2p}}$
3. $\Pr[|\overline{X} - p| \geq \epsilon] \leq 2e^{\frac{-n\epsilon^2}{p}}$

Proof. Let $\alpha > 0$ be a parameter that we will fix later. Define a new random variable $Z = e^{\alpha n(\overline{X} - p)}$. Note that since the exponential function is increasing, we have

$$
\begin{aligned}
\overline{X} - p \geq \epsilon &\Leftrightarrow \alpha n(\overline{X} - p) \geq \alpha n\epsilon \\
&\Leftrightarrow Z \geq e^{\alpha n\epsilon}
\end{aligned}
$$

Therefore we can write

$$
\begin{aligned}
\Pr[\overline{X} - p \geq \epsilon] &= \Pr[Z \geq e^{\alpha n\epsilon}] \\
&\leq \mathbf{E}[Z] \cdot e^{-\alpha n\epsilon}, \qquad (3.1)
\end{aligned}
$$

by Markov's inequality.

Now define $Z_i = e^{\alpha(X_i - p)}$. It is easy to see that $\mathbf{E}[Z_i] = p \cdot e^{\alpha q} + q \cdot e^{-\alpha p}$. Note also that $Z = \prod Z_i$ and since Z_i's are mutually independent we have $\mathbf{E}[Z] = \prod \mathbf{E}[Z_i] = (pe^{\alpha q} + qe^{-\alpha p})^n$.

Fact 43. $pe^{\alpha q} + qe^{-\alpha p} \leq e^{\frac{\alpha^2 q}{2}}$

This implies $\mathbf{E}[Z] \leq e^{\frac{\alpha^2 qn}{2}}$. Now by 3.1 and the above fact we have $\Pr[\overline{X} - p \geq \epsilon] \leq e^{\frac{\alpha^2 qn}{2} - \alpha n\epsilon}$. Setting $\alpha = \frac{\epsilon}{q}$ completes the proof. □

Example 44. Let A be any randomized algorithm that makes an error with probability at most $\frac{1}{3}$. We would like to reduce this probability to $2^{-\Omega(k)}$. For simiplicity assume that the algorithm gives only yes or no answers. We run A for $O(k)$ times and take a majority vote among the answers. Chernoff bound then guarantees that the error is bounded by $2^{-\Omega(k)}$.

3.2 Non-deterministic Communication (continued)

Theorem 45. *For every function $f : X \times Y \to \{0, 1\}$ we have $D(f) = O(N^0(f)N^1(f))$.*

Proof. We will use the following simple property of rectangles. Let $R_1 = S_1 \times T_1$ be a 1-rectangle and $R_2 = S_2 \times T_2$ be a 0-rectangle. Then it is not the case that $S_1 \cap S_2 \neq \emptyset$ and $T_1 \cap T_2 \neq \emptyset$. To see this assume that there exist $s \in S_1 \cap S_2$ and $t \in T_1 \cap T_2$. This means that R_1 and R_2 overlap which is contradiction given their colors.

The idea behind the protocol is that Alice and Bob try to find a 0-rectangle that contains both x and y. If they succeed then $f(x, y) = 0$, otherwise they conclude that $f(x, y) = 1$. To implement this they keep a candidate set of 0-rectangles (which initially contains all 0-rectangles) that "might" contain both x and y. In each round they "kill" at least half of the candidate rectangles by exchanging $O(\log C^1(f))$ bits. Since there are initally $C^0(f)$ rectangles, there will be $O(\log C^0(f))$ stages and hence giving the desired bound. Each stage of the protocol goes as follows.

1. Alice looks at the 0-rectangles that are still alive. If there is no such rectangle then she outputs $f(x, y) = 1$. Otherwise she looks for a 1-rectangle Q consistent with x that intersects in rows with at least half of the 0-rectangles that are alive. If she found such rectangle she sends the name of it to Bob and kills all 0-rectangles that do not intersect in rows with Q. Otherwise she tells Bob that she did not succeed.

2. Bob looks for a 1-rectangle consistent Q with y that intersects in columns with at least half of the 0-rectangle that are alive. If he found such rectangle he sends its name to Alice and kills all 0-rectangles that do not intersect in columns with Q. Otherwise he outputs $f(x, y) = 0$.

It is not hard to see that the above protocol satisfies the required communication bound. It thus only remains to show that it is works correctly. If (x, y) is in some 0-rectangle R, then R never dies during the protocol. It is simply because any 1-rectangle consistent with x intersects in rows with R and any 1-rectangle consistent with y intersects with R in columns. Therefore if no 0-rectangle is alive then $f(x, y) = 1$. If neither Alice nor Bob finds the 1-rectangle Q in their turn, then $f(x, y) = 0$. The reason is that if (x, y) is in some 1-rectangle, then by the mentioned property it either intersects in rows with at most half of the 0-rectangles, or it intersects in columns with at most half of the 0-rectangles. Thus either Alice or Bob should find such a rectangle. □

Lecture 4

Randomized Communication Complexity

Scribe: Tore Frederiksen

4.1 Model

So far we have only talked about deterministic protocols and protocols which exhibit non-determinism. Today we'll discuss protocols in which randomness are used during the execution. For this purpose we will always use the notion of an unbiased coin, which will determine a random bit, eg. 1 if the outcome of a coin-flip is head and 0 if the outcome is tail. This bit can then be used to decide whether to send message 1 or message 2 at a specific node in a protocol tree. So to model randomness in protocols we assume that each player has knowledge of a, potentially infinite, string of uniform random bits, $r \in \{0,1\}^*$. Using such strings we now define the notion of a Private Coin Protocol.

Definition 46 (Private Coin Protocol)**.** We wish to compute a function $f(a,b) \to \{0,1\}$ with $a \in \{0,1\}^n$ and $b \in \{0,1\}^n$ using a protocol P taking (a,b) as input. If Alice gets her input to the protocol, a, along with an infinite random string, $r_A \in \{0,1\}^*$ and Bob gets his input, b, along with another infinite random string, $r_B \in \{0,1\}^*$, we then call this protocol for a Private Coin Protocol.

We notice that such a protocol can also be described using a protocol tree, only in this case a given node, v, will be labeled with $a_v(a, r_A)$ or $b_c(b, r_B)$. That is, each edge choice will depend on a given player's input along with this player's random string.

Example 47 (Equality)**.** Let us try to apply this paradigm when solving the equality function (EQ). The idea we will use is common in the general construction of random algorithms; We turn the input into polynomials and transfer a random input to such a polynomial along with its output. What we specifically do is first to pick a random prime p, such that $n^2 < p < 2n^2$. Now, letting $a = a_0 a_1 ... a_n$ and $b = b_0 b_1 ... b_n$ be the individual bits in Alice's, respectively, Bob's input, we construct the following polynomials:

$$A(x) := a_0 + a_1 x + a_2 x^2 + ... + a_{n-1} x^{n-1} \mod p$$
$$B(x) := b_0 + b_1 x + b_2 x^2 + ... + b_{n-1} x^{n-1} \mod p$$

The protocol then consists of the following steps:

- Assume, without loss of generality, that Alice sends the first message. Alice generates an $x \in_R \{1, 2..., p\}$.
- Alice then computes $A(x)$ for this x.
- Now Alice sends $(x, A(x))$ to Bob.
- Bob calculates $B(x)$ and, if $A(x) = B(x)$ outputs 1, otherwise he outputs 0.

First notice that $A(x) - B(x) = 0 \iff A(x) = B(x)$, so if Bob outputs 0 then this output can never be wrong. However, if Bob output 1 it might be the case that the polynomials are not the same, but only give the same output on a specific input. Now, what is the probability of this happening and thus the 1 output being wrong? We notice that the wrong case can only happen for at most $n-1$ elements out of p. This is so since the difference between two non-zero polynomials is another polynomial of degree at most $n-1$, which again can have not more than $n-1$ roots. From this we see that the output is wrong with probability at most $\frac{n-1}{p} \leq n/n^2 = 1/n$. Next notice that any $x \in \{1, 2...p\}$ has $\mathcal{O}(\log n)$ bits and it is the same for $A(x)$. Thus the communication is only $\mathcal{O}(\log n)$, unlike the $\Theta(n)$ in the deterministic protocol.

Definition 48 (Random Protocol Types)**.** Let P be a randomized protocol taking (x, y) as input, then:

1. P computes a function $f : \{x, y\} \rightarrow \{0, 1\}$ with zero error if
$$\forall x, y \; Pr_{r_A, r_B}\left[P(x, y) \neq f(x, y)\right] = 0$$

2. P computes $f : \{x, y\} \rightarrow \{0, 1\}$ with error ϵ if
$$\forall x, y \; Pr_{r_A, r_B}\left[P(x, y) \neq f(x, y)\right] \leq \epsilon$$

3. P computes $f : \{x, y\} \rightarrow \{0, 1\}$ with one-sided error ϵ if, for all (x, y) when $f(x, y) = 0$ then
$$Pr_{r_A, r_B}\left[P(x, y) = 1\right] = 0$$
and if $f(x, y) = 1$ then
$$Pr_{r_A, r_B}\left[P(x, y) = 0\right] \leq \epsilon$$

4.2 Worst-case Vs. Average-case

We will now define what we mean by the worst-case and average-case cost of randomized protocols.

Definition 49. Given a protocol P we say that the *worst-case cost* of the protocol is the maximum amount of communication needed for any pair (x, y) and any pair of random strings (r_A, r_B). That is,
$$\text{worst-case } \mathit{cost}(P) := \max_{x,y} \max_{r_A, r_B} [\text{amount of communication}]$$

Given a protocol P we say that the *average-case cost* of the protocol is the maximum amount of communication needed for any pair (x, y), using the expected amount of communication given a pair of random string (r_A, r_B). That is,
$$\text{average-case } \mathit{cost}(P) := \max_{x,y} E_{r_A, r_B}[\text{amount of communication}]$$

Given this definition we introduce the following notion of randomized communication complexity:

Definition 50. Given a function $f : \{x, y\} \rightarrow \{0, 1\}$ and a protocol P for computing the function, we introduce the following kinds of randomized communication complexity:

1. *Las Vegas Communication Complexity*: is denoted $R_0(f)$ and is defined to be the minimum of the average-case costs of any protocol computing f.

2. *Monte Carlo Communication Complexity:* is denoted $R_\epsilon(f)$ for some error probability $0 < \epsilon < 1/2$, and is defined to be the minimum of the worst-case costs of any protocol computing f with error probability ϵ. For simplicity we define $R(f) := R_{\frac{1}{3}}(f)$.

3. *One-sided error Communication Complexity:* is defined to be $R^1_\epsilon(f)$ for some error probability $0 < \epsilon < 1/2$, and is defined to be the minimum of the worst-case costs of any protocol computing f with a one-sided error probability of ϵ.

Using these definitions, the two following theorems apply:

Theorem 51. *The one-sided error communication complexity for any function f and any ϵ between 0 and 1/2 will always be at least as much as the non-deterministic communication complexity for the same function. That is,*

$$R^1_\epsilon(f) \geq N^1(f)$$

Theorem 52. *The Monte Carlo Communication Complexity for any function f and $\epsilon = 1/3$ is lower bounded by the logarithm of the deterministic complexity of f. That is,*

$$R(f) = \Omega(\log D(f))$$

Proof. First consider the following facts:

Fact 53. $f(x, y) = 1 \Rightarrow E[\text{ protocol output }] \geq 1 - 1/3 = 2/3$.

Fact 54. $f(x, y) = 0 \Rightarrow E[\text{ protocol output }] \leq 1/3$.

Notice that these are true as the output of the protocol is either 0 or 1 and since we only have 1/3 error probability, thus we expect the protocol to output 1 at least 2/3 of the times, and since the other alternative output is 0, we get the expected output to be 2/3.

Our idea is now to present a deterministic simulation of any randomized protocol, P, and then show that this protocol will not need more than $2^{R(f)}$ bits in communication. To do so we start by fixing a pair (x, y). Then for each leaf ℓ in the randomized protocol tree we let P_ℓ be the probability that we reach this specific leaf given the input (x, y). From this, along with our facts, we get that $E[\text{ output }] = \sum_\ell[\text{ output at leaf } \ell]$. Now let P^A_ℓ be the probability that all the responses Alice makes during the protocol will make her and Bob continue on the path to ℓ in the protocol tree. On the other hand, let P^B_ℓ be the probability that Bob's response at each step in the protocol would make him and Alice continue down the path to leaf ℓ.

We now have that $P_\ell = P^A_\ell \cdot P^B_\ell$ since the events P^A_ℓ and P^B_ℓ are independent. So now the deterministic protocol would go as follows:

- Alice sends $\{P^A_\ell\}$ for all leaves ℓ.
- Bob computes $\{P^B_\ell\}$ and all $\{P_\ell = P^A_\ell \cdot P^B_\ell\}$.
- Bob also computes $E[\text{ output }] = \sum_\ell P_\ell[\text{ protocol outputs at } \ell]$.
- If $E[\text{ output }] > 1/2$ then Bob outputs 1, otherwise he outputs 0.

Now notice that the communication cost is one real number (probability) for each leaf and that we have at most $2^{R(f)}$ leafs in our randomized protocol. However, we do not need more than a constant amount of bits precision, thus we end up needing to communicate $\mathcal{O}(2^{R(f)})$ bits in the deterministic protocol. From this it follows that $R(f) = \Omega(\log D(f))$. □

4.3 Public Coin Protocols

The notion of public coin protocols is very similar to that of the private coin protocols, except that we now assume that Alice and Bob have access to the same random string, that is $r_A = r_b = r \in \{0, 1\}^*$. We define it formally a little bit different:

Definition 55 (Public Coin Protocol)**.** A Public Coin Protocol is a distribution over deterministic protocols. Further, we say that a Public Coin Protocol, P, on input (x, y) along with a random string $r \in \{0, 1\}^*$ computes a function $f(x, y) \to \{0, 1\}$ with an error probability $\epsilon \in]0, 1/2[$ if

$$\forall x, y \; Pr_r[P(x, y) \neq f(x, y)] \leq \epsilon$$

We then say that $R^{pub}_\epsilon(f)$ is the minimum cost of any Public Coin Protocol, P, that computes f with an error probability less than or equal to ϵ.

We should here notice that $R_\epsilon(f) \geq R_\epsilon^{pub}(f)$ since Alice and Bob can always simulate the public coin protocol by pretending that $r = (r_A, r_B)$.

Example 56 (Equality)**.** We wish to compute the equality function, EQ. As usual Alice gets $a \in \{0,1\}^n$ and Bob gets $b \in \{0,1\}^n$, and we assume that Alice begins. The trick to this protocol is now to use an inner product modulo 2, IP. Now the protocol goes as follows:

- Alice picks $r \in_R \{0,1\}^n$ and computes $\langle a, r\rangle$ and sends it to Bob (We use $\langle \cdot, \cdot \rangle$ to denote the inner product modulo 2).
- Bob computes $\langle b, r\rangle$ and checks if $\langle a, r\rangle = \langle b, r\rangle$. If so, he outputs 1, otherwise he outputs 0.

The key is here to notice the following fact:

Fact 57. If a and b are bit-strings and $a \neq b$ then $Pr_r[\langle a, r\rangle = \langle b, r\rangle] = 1/2$.

Using this fact we see that Bob's output will be correct with probability 1/2. However, if we repeat the protocol for a different r the error probability is decreased to 1/4. We can do this as many times as we like, decreasing the error probability down to $1/2^i$ where i is the amount of iterations. Thus, the cost of the protocol is only constant!

Fact 58. The public coin communication complexity for a one-sided error protocol computing the 'not equal' function is constant. That is,

$$R^{1,pub}(NEQ) = 1$$

4.4 Newman's Theorem

Theorem 59 (Newman's Theorem)**.** *For all ϵ and δ greater than 0 we have that the Monte Carlo Communication Complexity for a function f is less than or equal to the public coin communication complexity with error ϵ along with a logarithm overhead. Or more formally:*

$$\forall \epsilon, \delta > 0 \quad R_{\epsilon+\delta}(f) \leq R_\epsilon^{pub}(f) + O(\log n/\delta)$$

Proof. Let P be the best protocol for computing $R_\epsilon^{pub}(f)$. The idea is now to reduce the randomness needed for the protocol to only $\mathcal{O}(\log \frac{n}{\delta})$ bits (which unfortunately causes the error to increase by δ) without increasing the complexity. Alice will then generate these random bits privately and send these random bits to Bob so he can complete the protocol using this $\mathcal{O}(\log \frac{n}{\delta})$-bit string as the public randomness. For this approach to work we wish to have a function taking a short random string and returning a long random string in order to account for the reduced randomness. Before we continue we need the following definition:

Definition 60 (Indicator Random Variable)**.** We define an indicator random variable Z on input x, y and random string r for a function f using a protocol P to be 1 if $P(x,y) \neq f(x,y)$ using the random string r and 0 otherwise. Or more formally:

$$Z(x,y,r) = \begin{cases} 1 & \text{if } P(x,y) \neq f(x,y) \text{ on a random string } r \\ 0 & \text{otherwise} \end{cases}$$

We now notice that for all fixed pairs (x,y) $E_r[Z(x,y,r)] \leq \epsilon$ as P computes f with error probability ϵ. To continue we must decide on a small fixed t and pick a small set of random strings, $\{r_1, ..., r_t\}$ i.i.d. from $\{0,1\}^*$. The new protocol then consists of Alice picking $i \in_R \{1, ..., t\}$ and using string r_i as the random string in the new protocol, which we will call $P_{r_1,...r_t}$. We now need to show that there exists such strings $r_1, ...r_t$ such that $E_i[Z(x,y,r_i)] \leq \epsilon + \delta$ for all (x,y). We do this by considering a specific input pair, (x,y), and compute the probability that $E_i[Z(x,y,r_i)] > \epsilon + \delta$ where i is uniformly distributed. We will call this event $Bad_{x,y}$.

We then define $BAD := \vee_{x,y} Bad_{x,y}$, that is the event that at least one $Bad_{x,y}$ event occurs. We have that $E_i[Z(x,y,r_i)] > \epsilon + \delta \iff \frac{1}{t}\sum_{i=1}^{t} Z(x,y,r_i) > \epsilon + \delta$. So by the Chernoff bound, since $E_r[Z(x,y,r)] \leq \epsilon$, we have the following

$$Pr[Bad_{x,y}] = Pr\left[\frac{1}{t}\sum Z(x,y,r_i) - \epsilon > \delta\right] \leq 2 \cdot e^{-2\delta^2 t}$$

Finally, using the union bound, this gives us:

$$Pr[BAD] \leq 2^{2n} \cdot 2e^{-\delta^2 t} \approx e^{\Theta(n) - \Theta(\delta^2 t)}$$

Now when $t = \mathcal{O}(\log(n/\delta^2))$ then $Pr[BAD] < 1$ which imply that there exists $r_1, ..., r_t$ such that $P_{r_1,...r_t}$ is an $\epsilon + \delta$ protocol. This again implies that there exists choices of $r_1, ..., r_t$ for every pair (x,y) such that the error of $P_{r_1,\ldots,r_t}$ is at most $\epsilon + \delta$. Now since the only communication overhead in $P_{r_1,\ldots,r_t}$ is sending the random t it will be $\log t = \mathcal{O}(\log(n/\delta))$ and so we are done. □

NOTE: The above proof uses the probabilistic method – we need to show that a suitable set of random strings $\{r_1, \ldots, r_t\}$ exist, so we generate them at random and show with nonzero probability this randomly generated set works for us. We can then include that there must exist at least one set $\{r_1, \ldots, r_t\}$ that works for our purposes.

Lecture 5

Distributional Communication Complexity

Scribe: Jana Kunert

5.1 Distributional Complexity

Distributional Complexity is a model, where we consider a probability distribution over inputs. We use this model to prove lower bounds for randomised protocols.

Definition 61. Let μ be a distribution on $X \times Y$. A (μ, ϵ)-*protocol* for f is a deterministic protocol s.t.

$$Pr_{(X,Y)\sim\mu}\left[P(x,y) \neq f(x,y)\right] \leq \epsilon$$

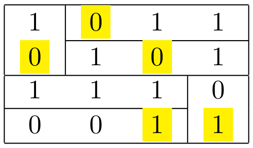
i.e. the probability of the protocol returning a value not equal to the value of the function on random input x and y is less or equal ϵ.

Definition 62. The *Distributional Complexity* $D^{\mu}_{\epsilon}(f)$ is the cost of the best (μ, ϵ)-protocol for f.

Example 63. Look at the following communication matrix:

$$\begin{array}{|cccc|}
\hline
1 & 0 & 1 & 1 \\
0 & 1 & 0 & 1 \\
1 & 1 & 1 & 0 \\
0 & 0 & 1 & 1 \\
\hline
\end{array}$$

We split it into rectangles, but the rectangles are not necessarily monochromatic:



$$\begin{array}{|c|ccc|}
\hline
1 & 0 & 1 & 1 \\
\cline{2-4}
0 & 1 & 0 & 1 \\
\hline
1 & 1 & 1 & \multicolumn{1}{|c|}{0} \\
\cline{1-3}
0 & 0 & 1 & \multicolumn{1}{|c|}{1} \\
\hline
\end{array}$$

When hitting a yellow field, we make a mistake.

Example 64. Consider a uniform distribution μ and identity matrix as communication matrix.
What is $D^{\mu}_{\epsilon}(EQ)$ for $\epsilon = \frac{1}{3}$?
Answer: $D^{\mu}_{\frac{1}{3}}(EQ) = 0$—when inputs are *uniformly* distributed, inputs are equal with probability only 2^{-n}. A protocol can output zero and be correct with probability $1 - 2^{-n}$.

5.2 Yao's Minimax Lemma

Lemma 65. $R_\epsilon^{pub}(f) = \max_\mu D_\epsilon^\mu(f)$.

Proof. (easy direction $\geq$)
Fix an input distribution μ. Let $(X,Y) \sim \mu$. Fix a public coin protocol P for f.
Define

$$Z(x,y,r) := \begin{cases} 1 & \text{if } P \text{ errs on } (x,y) \text{ given a random string } r \\ 0 & \text{otherwise} \end{cases}$$

Now, by the error guarantee of P, we know that

$$\forall x,y : E_R[Z(x,y,R)] \leq \epsilon$$

Since this is true for all (x,y), we can conclude that

$$\begin{aligned} \epsilon &\geq E_{(X,Y)\sim\mu} E_R[Z(X,Y,R)] \\ &= E_R E_{(X,Y)\sim\mu}[Z(X,Y,R)] \end{aligned}$$

Therefore, there must be some r, such that

$$E_{(X,Y)\sim\mu}[Z(X,Y,r)] \leq \epsilon$$

The other direction of the proof uses the Nash Equilibrium over 2-player zero sum games. The proof is beyond the scope of this course. □

The following two sections contain two different ideas to proof lower bounds.

5.3 Corruption

Find a distribution μ such that

(1) most inputs give 1 as answer:
$Pr_\mu[f(x,y)=1] \geq \epsilon$ for some $\epsilon > \frac{1}{2}$

(2) any large 1-rectangle contains many zeros.

Example 66. DISJ
Input: $x, y \in \{0,1\}^n$
Output: DISJ $(x,y) = \begin{cases} 1 & \text{if } x \wedge y = \emptyset \\ 0 & \text{otherwise} \end{cases}$
Look at X, Y as subsets $X, Y \subseteq \{1, \cdots, n\}$.

Fact 67. There is a distribution μ, constants $\alpha, \delta > 0$ and sets $A \subseteq \text{DISJ}^{-1}(1)$, $B \subseteq \text{DISJ}^{-1}(0)$, such that

(1) $\mu(A) \geq \frac{3}{4}$

(2) For every rectangle R: $\mu(R \cap B) \geq \alpha\mu(R \cap A) - 2^{-\delta n}$

Proof. The proof of this fact is an exercise. □

We can now proof the following theorem, that gives us a lower bound on the randomised communication protocol for DISJ:

Theorem 68. $R_\epsilon^{pub}(\text{DISJ}) = \Omega(n)$.

Proof. Fix some small ϵ. We want to show that $D^{\mu}_{\epsilon}(\text{DISJ}) \geq \delta_n - O(1)$.
Define $c := D^{\mu}_{\epsilon}(\text{DISJ})$. Then there is a (μ, ϵ) protocol Q for DISJ, that partitions the communication matrix into $\leq 2^c$ rectangles.
Let $R_1, \cdots, R_t$ be the 1-rectangles in Q. The error of the protocol is $\leq \epsilon$, so by using fact 7 (1) we get

$$\mu\left(\bigcup_{i=1}^{t} R_i \cap A\right) \geq \frac{3}{4} - \epsilon$$

The idea is that every point in A that is not contained in $\bigcup R_i$, is an error - and therefore there are at most ϵ of these points. Fact 7 (2) gives for each R_i:

$$\begin{aligned}
\mu(R_i \cap B) &\geq \alpha\mu(R_i \cap A) - 2^{-\delta n} \\
\Rightarrow \epsilon \geq \mu\left(\bigcup_{i=1}^{t} R_i \cap B\right) &\geq \sum_{i=1}^{t} \left(\alpha\mu(R_i \cap A) - 2^{-\delta n}\right) \\
&\geq \sum_{i=1}^{t} \left(\alpha\mu(R_i \cap A)\right) - 2^c 2^{-\delta n} \\
&= \alpha\mu\left(\bigcup_{i=1}^{t} R_i \cap A\right) - 2^{c-\delta n} \\
&\geq \alpha\left(\frac{3}{4} - \epsilon\right) - 2^{c-\delta n} \\
\Rightarrow \epsilon &\geq \alpha\left(\frac{3}{4} - \epsilon\right) - 2^{c-\delta n} \\
\Rightarrow 2^{c-\delta n} &\geq \alpha\left(\frac{3}{4} - \epsilon\right) - \epsilon > \Omega(1) \\
c &\geq \delta n - O(1)
\end{aligned}$$

□

The first proof for this theorem was gives by Kalyanasundaram and Schnitger in 1987. The above proof is due to Razborov and simplifies the [KS87] result.

5.4 Discrepancy

The basic idea of discrepancy is that all large rectangles have nearly 50% 1-inputs and 50% 0-inputs.

Definition 69. Let $f : X \times Y \rightarrowtail \{0,1\}$. For any rectangle R and a distribution μ on $X \times Y$ the discrepancy is defined as

$$\text{Disc}_{\mu}(R, f) := |Pr_{\mu}[f(x,y) = 0 \wedge (x,y) \in R] - Pr_{\mu}[f(x,y) = 1 \wedge (x,y) \notin R]|$$

Definition 70. The discrepancy of f with respect to μ is defined as

$$\text{Disc}_{\mu}(f) := \max_{R} \text{Disc}_{\mu}(R, f)$$

Theorem 71. *For all f, all μ, all $\epsilon > 0$:*

$$D^{\mu}_{\frac{1}{2}-\epsilon}(f) \geq \log\left(\frac{2\epsilon}{\text{Disc}_{\mu}(f)}\right)$$

Proof. Let P be a c-bit, $\left(\frac{1}{2}-\epsilon\right)$-protocol for f. Then

$$\begin{aligned}
2\epsilon &= \left(\frac{1}{2}+\epsilon\right) - \left(\frac{1}{2}-\epsilon\right) \\
&= Pr[P(x,y) = f(x,y)] - Pr[P(x,y) \neq f(x,y)] \\
&= \sum_{\text{leaves l}} Pr[P(x,y) = f(x,y) \wedge (x,y) \in R_l] - Pr[P(x,y) \neq f(x,y) \wedge (x,y) \in R_l] \\
&\leq \sum_{\text{leaves l}} \big| Pr[P(x,y) = f(x,y) \wedge (x,y) \in R_l] - Pr[P(x,y) \neq f(x,y) \wedge (x,y) \in R_l] \big| \\
&= \sum_l \big| Pr[f(x,y) = 0 \wedge (x,y) \in R_l] - Pr[f(x,y) = 1 \wedge (x,y) \in R_l] \big| \\
&\leq \sum_l \mathrm{Disc}_\mu(f) \\
&\leq 2^c \mathrm{Disc}_\mu(f) \\
\Rightarrow 2\epsilon &\leq 2^c \mathrm{Disc}_\mu(f) \\
c &\geq \log\left(\frac{2\epsilon}{\mathrm{Disc}_\mu(f)}\right)
\end{aligned}$$

□

Look at the following example:

Example 72. Inner product (IP)
Input: $x, y \in \{0,1\}^n$
Output: $\mathrm{IP}(x,y) := \sum_{i=1}^n x_i y_i \mod 2$

Recap from Linear Algebra

To proof the next theorem, we need a little recap from Linear Algebra:

vector norm $\|\vec{x}\|_2 = \sqrt{\sum_{i=1}^n x_i^2}$

matrix norm $\|M\| = \max_{\vec{v}:\|\vec{v}\|_2=1} \|M\vec{v}\|_2$

eigenvector $\vec{v}$ is a eigenvektor if $M\vec{v} = \lambda\vec{v}$ for some scalar λ

eigenvalue λ is a eigenvalue if $M\vec{v} = \lambda\vec{v}$ for some vektor $\vec{v}$

Cauchy-Schwarz $\vec{a} \cdot \vec{b} \leq \|\vec{a}\|\|\vec{b}\|$

We also need Fact 13, which we will use without a proof.

Fact 73. $|\vec{v} \cdot A\vec{w}| \leq \|\vec{v}\|_2 \cdot \|A\| \cdot \|\vec{w}\|_2$

The last thing we need is the following claim:

Claim 1. $H \cdot H^T = 2^n I$

Proof. $H \cdot H^T[x,y] = \sum_Z H(x,z) \cdot H(z,y)$
if $x = y$ then $HH^T[x,y] = \sum_z 1 = 2^n$
if $x \neq y$ then $HH^T[x,y] = \frac{2^n}{2} \cdot (+1) + \frac{2^n}{2} \cdot (-1) = 0$ □

Theorem 74. *Let μ be an uniform distribution. Then* $\mathrm{Disc}_\mu(\mathrm{IP}) = 2^{-\frac{n}{2}}$.

Proof. Define $2^n \times 2^n$ matrix H as $H[x,y] = \begin{cases} 1 & \text{if } \mathrm{IP}(x,y) = 0 \\ -1 & \text{if } \mathrm{IP}(x,y) = -1 \end{cases}$ Look at a rectangle $S \times T$.

$$\begin{aligned}
\mathrm{Disc}_\mu(S \times T, \mathrm{IP}) &= |Pr[\mathrm{IP}(x,y) = 0 \wedge (x,y) \in S \times T] - Pr[\mathrm{IP}(x,y) = 1 \wedge (x,y) \in S \times T]| \\
&= \frac{\sum_{(x,y)\in S\times T} H[x,y]}{2^{2n}} = \frac{\vec{1}_S \cdot (H \cdot \vec{1}_T)}{2^{2n}} \\
&\leq \frac{\|\vec{1}_s\| \cdot \|H\| \cdot \|\vec{1}_T\|}{2^{2n}} \\
&\leq \frac{\sqrt{2^n} \cdot \sqrt{2^n} \cdot \sqrt{2^n}}{2^{2n}} \\
&= \frac{2^{\frac{3}{2}n}}{2^{2n}} = 2^{-\frac{n}{2}}
\end{aligned}$$

□

Now we can proof the following corollary, that gives us a lower bound for the randomised protocol for IP:

Corollary 75. $R^{pub}_{\frac{1}{2}-\epsilon}(\mathrm{IP}) \geq D^{\mu}_{\frac{1}{2}-\epsilon}(\mathrm{IP}) \geq \frac{n}{2} - \log\frac{1}{\epsilon}$

Proof. This follows immediately from theorem 14 in combination with theorem 11:

$$\begin{aligned}
D^{\mu}_{\frac{1}{2}-\epsilon}(\mathrm{IP}) &\geq \log\left(\frac{2\epsilon}{\mathrm{Disc}_\mu(\mathrm{IP})}\right) \\
D^{\mu}_{\frac{1}{2}-\epsilon}(\mathrm{IP}) &\geq \log\left(\frac{2\epsilon}{2^{-\frac{n}{2}}}\right) = \log\left(\frac{2^{\frac{n}{2}}}{2\frac{1}{\epsilon}}\right) \\
&\geq \frac{n}{2} - \log\frac{1}{\epsilon}
\end{aligned}$$

Lemma 5 gives the rest of the proof:

$$\begin{aligned}
R^{pub}_{\epsilon'}(f) &= \max_{\mu} D^{\mu}_{\epsilon'}(f) \geq D^{\mu}_{\epsilon'}(f) \\
R^{pub}_{\frac{1}{2}-\epsilon}(\mathrm{IP}) &\geq D^{\mu}_{\frac{1}{2}-\epsilon}(\mathrm{IP}) \geq \frac{n}{2} - \log\frac{1}{\epsilon}
\end{aligned}$$

□

Lecture 6

Crash Course in Information Theory

Scribe: Jana Kunert, Lisbeth Andersen, Jakob Friis

6.1 Entropy

Consider the World Cup Quarter finals with the teams in Table 6.1. We want to tell who won the World Cup using as few bits as possible.

Team	Chances of winning	Prefix 1	Prefix 2
Team 1	1/2	000	1
Team 2	1/4	001	01
Team 3	1/8	010	001
Team 4	1/16	011	0001
Team 5	1/64	100	000000
Team 6	1/64	101	000001
Team 7	1/64	110	000010
Team 8	1/64	111	000011

Table 6.1: World Cup Quarter Finals

The naive solution is illustrated as 'Prefix 1' in the third column: No matter which team wins, we always send 3 bits.
'Prefix 2' takes acount of the fact, the Team 1 is much more likely to win than any other team. If we use these prefixes, we compute the expected message length as

$$1 \cdot \frac{1}{2} + 2 \cdot \frac{1}{4} + 3 \cdot \frac{1}{8} + 4 \cdot \frac{1}{16} + 4 \cdot \left(6 \cdot \frac{1}{64}\right) = 2$$

thus, we get a shorter expected message length. We looked at the variable and used its distribution to reduce the expected message length. To do so, we need to measure the amount of randomness in a variable.

Shannon Entropy

We will use the following notation to define the Entropy of a random variable:

$\mathfrak{X}$ - a finite set

X - a random variable on $\mathfrak{X}$

p(x) - the probability the x happens: $Pr[X = x]$

The (Shannon) Entropy $H(X)$ is now defined as

$$H(X) = \sum_{x \in \mathfrak{X}} p(x) \cdot \log \frac{1}{p(x)}$$

If there is no uncertainty in X, the entropy $H(X) = 0$. Clearly, the value is maximized when X is a uniformly distributed.

Theorem 76. *Source Coding Theorem [Shannon '48]*
Let $l_1, l_2, \cdots, l_n$ be the length of codewords of $x_1, x_2, \cdots, x_n \in \mathfrak{X}$ in the best encoding of X. Let

$$L := \sum_{i=1}^{n} p(x_i) \cdot l_i$$

be the expected code length. Then

$$H(X) \leq L < H(X) + 1$$

Joint Entropy

If X, Y are random variables with the joint distribution $p(x, y)$, then the joint entropy $H(X, Y)$ is defined as

$$H(X, Y) := \sum_{x \in X} \sum_{y \in Y} p(x, y) \log \frac{1}{p(x, y)}$$

Conditional Entropy

If X, Y are random variables with the joint distribution $p(x, y)$, then the conditional entropy $H(Y|X)$ is defined as

$$\begin{aligned} H(Y|X) :=& E_x[H(Y|X = x)] \\ =& \sum_{x \in \mathfrak{X}} p(x) H(Y|X = x) \\ =& \sum_{x \in \mathfrak{X}} p(x) \sum_{x \in \mathfrak{Y}} p(y|x) \log \frac{1}{p(y|x)} \end{aligned}$$

Chain Rule

Let X, X_1, X_2, $\cdots$ X_n, Y be random variables. Then the following Chain Rule can be applied:

$$H(X, Y) = H(X) + H(Y|X) \tag{6.1}$$

$$H(X_1, \cdots, X_n) = \sum_{i=1}^{n} H(X_i | X_1, \cdots, X_{i-1}) \tag{6.2}$$

Notice, that the second equation (over n variables) is a generalization of the first equation (over 2 variables).

Corollary 77. *Mutual Random Variables*

$$H(X, Y|Z) = H(X|Z) + H(Y|X, Z)$$

6.2 Mutual Information

If X and Y are two random variables, the Mutual Information $I(X, Y)$ measures us how much Y tells us about X:

$$I(X, Y) := H(X) - H(X|Y)$$

Some easy facts about mutual information:

$$I(X, Y) = H(X) + H(Y) - H(X, Y) \tag{6.3}$$
$$I(X, Y) = H(Y) - H(Y|X) \tag{6.4}$$
$$I(X, Y) = I(Y, X) \tag{6.5}$$
$$I(X, X) = H(X) - H(X|X) = H(X) \tag{6.6}$$

Remarks: The first fact (3) can be shown by using the Chain Rule. Fact (4) is just symmetric to fact (3).

Conditional Mutual Information

If X, Y and Z are three random variables, the Conditional Mutual Information $I(X, Y|Z)$ is defined as

$$I(X, Y|Z) := H(X|Z) - H(X|Y, Z)$$

Corollary 78. *Chain Rule for Conditional Mutual Information*

$$I(X_1, \cdots, X_n, Y) = \sum_{i=1}^{n} I(X_i, Y|X_1, \cdots, X_{i-1})$$

Proof.

$$\begin{aligned}
I(X_1, \cdots, X_n, Y) &= H(X_1, \cdots, X_n) - H(X_1, \cdots, X_n|Y) \\
&= \sum_{i=1}^{n} H(X_i|X_1, \cdots, X_{i-1}) - \sum_{i=1}^{n} H(X_i|Y, X_1, \cdots, X_{i-1}) \\
&= \sum_{i=1}^{n} (H(X_i|X_1, \cdots, X_{i-1}) - H(X_i|Y, X_1, \cdots, X_{i-1})) \\
&= \sum_{i=1}^{n} I(X_i, Y|X_1, \cdots, X_{i-1})
\end{aligned}$$

□

Definition 79. The Kullback-Lieber Divergence between distribution p and q is defined as:

$$D(p||q) := \sum_{x\in\mathcal{X}} p(x)\log\frac{p(x)}{q(x)} = E_{x\sim p}\left[\log\frac{p(x)}{q(x)}\right]$$

Kullback-Liebler Divergence is sometimes written as $D_{KL}(p||q)$ and is also known as KL-divergence, KL-distance and relative entropy.

Theorem 80. *Let $p(x,y)$ be a joint distribution on random variables X, Y. Let $p(x), p(y)$ be the marginal distribution of X and Y. Then*

$$D(p(x,y)||p(x)p(y)) = I(X;Y)\ .$$

We'll leave the proof of this as an exercise.

Definition 81. $f : \mathbb{R} \to \mathbb{R}$ is *convex* in the interval (a,b) if for all $x_1, x_2 \in (a,b)$ and all $0 \le \lambda \le 1$, we have

$$f(\lambda x_1 + (1-\lambda)x_2) \le \lambda f(x_1) + (1-\lambda)f(x_2)\ .$$

Theorem 82. ***(Jensen's Inequality)*** *If f is is* convex *and X is a random variable then*

$$E_X[f(X)] \ge f(E_X[X])\ .$$

Proof. (by induction on $|\mathcal{X}|$)

The base case is when $|\mathcal{X}| = 2$. Note that $E[f(x)] = p(x_1)f(x_1) + p(x_2)f(x_2)$. Setting $\lambda = p(x_1)$ and using the definition of convex functions, we get

$$E[f(x)] \ge f(p(x_1)x_1 + p(x_2)x_2) = f(E[x])\ .$$

For the induction step, suppose $|\mathcal{X}| = k$. Then, we have

$$E[f(X)] = \sum_i p(x_i)f(x_i) = p(x_k)f(x_k) + \sum_{i=1}^{k-1} p(x_i)f(x_i)\ . \tag{6.7}$$

Next, set $p'(x_i) = \frac{p(x_i)}{1-p(x_k)}$. Note that $p'(x)$ is a probability distribution on $\{x_1, \ldots, x_{k-1}\}$, so Equation (6.7) becomes

$$\begin{aligned}
E[f(X)] &= p(x_k)f(x_k) + (1-p(x_k))\sum_{i=1}^{k-1} p'(x_i)f(x_i) \\
&\ge p(x_k)f(x_k) + (1-p(x_k))f\left(\sum_{i=1}^{k-1} p'(x_i)x_i\right) \\
&\ge f\left(p(x_k)x_k + (1-p(x_k))\sum_{i=1}^{k-1} p'(x_i)x_i\right) \\
&= f\left(\sum_{i=1}^{k} p(x_i)x_i\right) \\
&= f(E[x])\ ,
\end{aligned}$$

where the first inequality follows from the induction hypothesis, and the second inequality is our base case. □

Theorem 83. *KL-divergence is* non-negative *$D(p||q) \ge 0$, with equality iff $p = q$.*

Proof. Let A:= $\{x : p(x) > 0\}$. By definition of KL-divergence, we get

$$\begin{aligned}
-D(p||q) &= -\sum_{x\in A} p(x)\log\frac{p(x)}{q(x)} \\
&= \sum_{x\in A} p(x)\log\frac{q(x)}{p(x)} \\
&= E_{x\sim p}\left[\log\frac{q(x)}{p(x)}\right] \\
&\leq \log E_{x\sim p}\left[\frac{q(x)}{p(x)}\right] \\
&= \log\left(\sum_{x\in A}\left(p(x)\frac{q(x)}{p(x)}\right)\right) \\
&= \log\left(\sum_{x\in A} q(x)\right) \\
&\leq \log(1) \\
&= 0.
\end{aligned}$$

Therefore, $D(p||q) \geq 0$. Note the inequality uses Jensen's inequality. It's easy to see that if $p = q$, then $p(x) = q(x)$ for all x, hence $E\left[q(x)/p(x)\right] = 1$, and $E[\log 1] = \log E[1] = 0$. □

Jensen's Inequality gives several other interesting lower bounds.

Corollary 84. *1. $I(X;Y) \geq 0$, with equality iff p(x,y) = p(x)p(y) i.e., if X, Y are independent.*

2. $I(X;Y|Z) \geq 0$.

Theorem 85. *$H(X) \leq \log|\mathcal{X}|$ with equality iff X is uniformly distributed.*

Proof. Let u be uniform on $\mathcal{X}$, so $u(x) = \frac{1}{|\mathcal{X}|}$ for all x. Then

$$\begin{aligned}
0 &\leq D(p||u) \\
&= \sum_x p(x)\log\frac{p(x)}{u(x)} \\
&= \sum_x p(x)(\log(p(x)) + \log|\mathcal{X}|) \\
&= \sum_x p(x)\log|\mathcal{X}| - \sum_x p(x)\log\frac{1}{p(x)}
\end{aligned}$$

It follows that $0 \leq \log|\mathcal{X}| - H(X)$, hence $H(X) \leq \log|\mathcal{X}|$. □

Theorem 86. ***(Conditioning reduces entropy.)***
$H(X|Y) \leq H(X)$ with equality when X, Y are independent.

Theorem 87. ***(Subadditivity of entropy.)***
$H(X_1, \ldots, X_n) \leq \sum_{i=1}^n H(X_i)$ with equality when $\{X_i\}$ are independent.

Theorem 88. ***(Log-sum inequality.)*** *For non-negative numbers $a_1, \ldots, a_n$ $b_1, \ldots, b_n$, we have*

$$\sum_{i=1}^n a_i \log\frac{a_i}{b_i} \geq \left(\sum a_i\right)\log\frac{\sum a_i}{\sum b_i}$$

with equality iff $\{\frac{a_i}{b_i}\}$ is constant.

Lecture 7

Information Theory and the INDEX problem

Scribe: Falk Altheide

7.1 Information Theory continued

Lemma 89 (Chain rule for mutual information)**.** *For random variables $\{X_i\}$ and Y,*

$$I\left(X_1,\dots,X_n;Y\right)=\sum_{i=1}^{n} I\left(X_i;Y|X_1,\dots,X_{i-1}\right)$$

Proof. The claim follows from the chain rule for conditional entropy, we saw in the last lecture.

$$\begin{aligned} I\left(X_1\dots X_n\right) &= H\left(X_1,\dots,X_n\right)-H\left(X_1,\dots,X_n|Y\right)\\ &= \sum_{i=1}^{n} H\left(X_i|X_1,\dots,X_{i-1}\right)-H\left(X_i|Y,X_1,\dots,X_{i-1}\right)\\ &= \sum_{i=1}^{n} I\left(X_i;Y|X_1,\dots,X_{i-1}\right) \end{aligned}$$

□

Lemma 90 (Superadditivity of Mutual Information)**.** *If random variables $\{X_i\}$ are independent given another random variable Z, then*

$$I\left(X_i,\dots,X_n;Y|\,Z\right)\geq\sum_{i=1}^{n} I\left(X_i;Y|Z\right)$$

Proof. If $\{X_i\}$ are independent conditioned on Z, then $H\left(X_1\dots X_n|Z\right)=\sum_{i=1}^{n} H\left(X_i|Z\right)$ and

$$\begin{aligned} I\left(X_1\dots X_n;Y|Z\right) &= H\left(X_1\dots X_n\,|Z\right)-H\left(X_1\dots X_n|Y,Z\right)\\ &\geq \sum_{i=1}^{n} H\left(X_i|Z\right)-\sum_{i=1}^{n} H\left(X_i|Y,Z\right)\\ &= \sum_{i=1}^{n} I\left(X_i;Y|Z\right). \end{aligned}$$

□

Example 91. Let Z be empty and $n = 2$. We consider two independent, uniform coins $X_1, X_2 \in_R \{0,1\}$ and $Y = X_1 \oplus X_2$. It is easy to see, that $H(X_1) = H(X_2) = 1$, because they are uniform and $H(X_1, X_2) = 2$, because they are independent.

Now, we look at the following facts

Fact 1: $H(X_1|Y) = H(X_2|Y) = 1 \Rightarrow H(X_1) - H(X_1|Y) = H(X_2) - H(X_2|Y) = 0$.

Fact 2: $H(X_1, X_2|Y) = 1 \Rightarrow H(X_1, X_2) - H(X_1, X_2|Y) = 1$.

Fact 3: $H(Y|X_1, X_2) = 0$.

Now, we can combine these facts and get

$$I(X_1, X_2; Y) = H(X_1 X_2) - H(X_1 X_2|Y) = 2 - 1 = 1 \ ,$$

$$I(X_1; Y) = H(X_1) - H(X_1|Y) = 1 - 1 = 0 \ ,$$

$$I(X_2; Y) = H(X_2) - H(X_2|Y) = 1 - 1 = 0 \ ,$$

Hence,

$$1 = I(X_1, X_2; Y) > I(X_1; Y) + I(X_2; Y) = 0 \ .$$

You can also view this in terms of Y:

$$\begin{aligned} I(X_1, X_2; Y) &= H(Y) - H(Y|X_1, X_2) \\ &= 1 - 0 = 1 \end{aligned}$$

Hence, this is an example for strict superadditivity.

Definition 92 (Binary entropy)**.** Let X be a random variable with two possible values (a biased coin), where $\mathrm{pr}[X = 1] = p$, with $0 \leq p \leq 1$. Then we define $H(p) = p \log \frac{1}{p} + (1-p) \log \frac{1}{1-p}$ as the *binary entropy* and $H(X) = H(p)$.

Example 93. (*Note:* By convention $0 \log 0 = 0$ and $0 \log \frac{1}{0} = 0$.)

1. $H(0) = 0$
2. $H\left(\frac{1}{2}\right) = 1$

Definition 94 (The biased coin game)**.** Let X be a biased coin with $\mathrm{pr}[X = 1] = p$. Alice wants to guess the value of X. What is the best strategy for her to choose?

It turns out, that Alice guesses 1, if $p > \frac{1}{2}$ and 0, if $p < \frac{1}{2}$. By this strategy, her Error becomes $\min\{p, 1-p\}$.

If the error should be at most $\epsilon < 1/2$, then $H(X) \leq H(\epsilon)$, since $H(p) is increasing on [0, 1/2]$.

7.2 The INDEX problem

Now we come to one of the fundamental problems in communication complexity.

Definition 95 (The INDEX problem)**.** Alice gets a String $x \in \{0,1\}^n$. Bob gets an index $i \in [n] \equiv \{1, \ldots n\}$. Now, the desired output is the value of x_i, the ith bit of x.

Note. *The trivial upper bound for this problem is $O(\log n)$ in the case that Bob sends i to Alice (plus the output x_i, which Alice sends).*

Say only Alice sends a message to Bob and he outputs the answer. Then we get $O(n)$ bits as the upper bound (plus Bob's output).

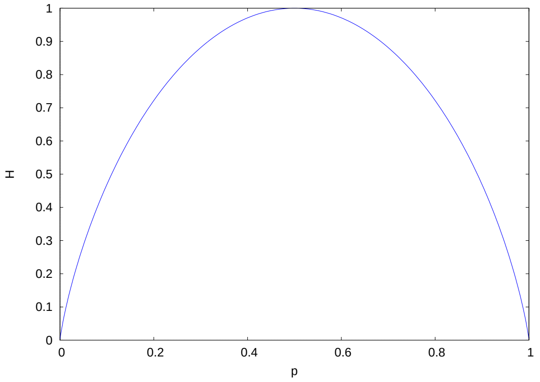


Figure 7.1: Binary entropy

Definition 96 (One-way protocol)**.** A *one-way protocol* is a protocol, where Alice sends a single message to Bob and Bob outputs the answer.

$R_\epsilon^{\rightarrow}(f)$ is the cost of the best one one-way, ϵ-error, randomised protocol for f.

Theorem 97 (Ablayev '96)**.** *$R_\epsilon^{pub,\rightarrow}(INDEX) \geq n\left(1 - H\left(\epsilon\right)\right)$*

Proof. At first, define a hard distribution: Let μ be uniform on $\{0,1\}^n \times [n]$.

By Yao's Lemma, it suffices to show that $D_\epsilon^\mu(\text{INDEX}) \geq n\left(1 - H\left(\epsilon\right)\right)$.

Let $c =$ cost of the best (μ, ϵ)-protocol for INDEX.

Alice sends a message $M = M\left(X\right)$ (note, that if X is a random variable, than $M\left(X\right)$ is a random variable, too).

Bob sees random variables I, M and needs to output X_I which is a single uniform bit random variable. Thus, this is an example of a biased coin game. After Alice's message, Bob needs to guess a biased coin X_I with entropy $H(X_I|I, M)$.

To finish the proof, we need the following facts

1. $H\left(X_I|I, M\right) \leq H\left(\epsilon\right)$
2. $c \geq n - H\left(X|I, M\right)$, which is Bob's impression of Alice's input
3. $H\left(X|I, M\right) \leq nH\left(X_I|I, M\right)$

Using these facts we see, that

$$\begin{aligned} c &\geq n - H\left(X|I, M\right) \\ &\geq n - nH\left(X_I|I, M\right) \\ &\geq n - nH\left(\epsilon\right) \\ &= n\left(1 - H\left(\epsilon\right)\right). \end{aligned}$$

□

It remains to prove Facts 2 and 3 above.

Proof. (Fact 2).

$$\begin{aligned} I\left(X; M|I\right) &= H\left(M|I\right) - H\left(M|X; I\right) \\ &\leq H\left(M\left|I\right|\right) \\ &\leq H\left(M\right) \\ &\leq \log\left(\#\text{ messages }\right) \\ &\leq c \text{ (there are at most } 2^c \text{ messages)} \end{aligned}$$

□

Proof. (Fact 3).

$$\begin{aligned} H\left(X_I|I, M\right) &= \frac{1}{n}\sum_{i=1}^{n} H\left(X_i|M\right) \\ &\geq \frac{1}{n}\sum_{i=1}^{n} H\left(X_i|M, M_1 \ldots M_{i-1}\right) \\ &\geq \frac{1}{n}\sum_{i=1}^{n} H\left(X_i|M; X_1 \ldots M_{i-1}, I\right) \\ &= \frac{1}{n} H\left(X|M, I\right) \end{aligned}$$

□

7.3 Circuit complexity

7.3.1 Boolean circuity

A boolean circuit is a directed, acyclic graph.

- nodes can be binary AND or OR gates or unary input of a variable z_i or the negation $\bar{z_i}$
- all circuits have fan-in 2 (note that a circuit with unbounded fan-in can be replaced by one, with fan-in at most 2, with a blow-up of $O(\log n)$ in depth and $O(n)$ in size.)
- the size of the circuit is the number of gates, the depth is the length of the largest path from input to output

Every circuit computes a function. We are interested in the smallest circuit for a given function.
The *XOR* function, for example, needs an exponential size circuit.

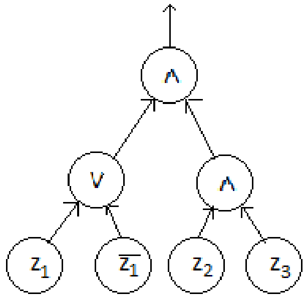
Lecture 8

Circuit Complexity

Scribe: Jakob Friis

Boolean Circuits. Recall our concept of Boolean circuits—they are directed acyclic graphs(DAGs), with

- n variables $Z_1, \dots, Z_n$
- **input nodes:** labelled with $Z_i or \bar{Z}_i$
- **gate nodes:** labelled with $\vee$ or $\wedge$



Definition 98. Circuit C is *monotone* if no input nodes is labeled with negations.

Definition 99. For $x, y \in \{0,1\}^n$ $x \leq y$ if $x_i \leq y_i$ for all $1 \leq i \leq n$. A function $f : \{0,1\}^n \to Z$ is *monotone* if $x \leq y \Rightarrow f(x) \leq f(y)$ for all x, y.

Complexity Measures. Let $depth(C)$ be the length of the longest input→output path in C. $size(C)$ is the number of gates in C. Define the depth and size complexities of a function f $d(f), s(f)$ to be the smallest $depth(C)/size(C)$ of a circuit C computing f. When f is a monotone function, we also define $d_m(f), s_m(f)$ to be the smallest depth/size of a monotone circuit computing f.

Definition 100. NC^i is the set of fuctions computeable by polynomial size, $O((\log n)^i)$ depth, fan-in 2 circuits. AC^i is the set of fuctions computeable by polynomial size, $O((\log n)^i)$ depth, with unbounded fan-in circuits.

Fact 101. For all i, $NC^i \leq AC^i \leq NC^{i+1}$.

The first inequality is a simple inclusion, since every fan-in two circuit is a circuit with unbounded fan-in. The second inequality is a homework exercise.

Showing an explicit function is outside NC^1 is a major open question in theoretical computer science. When we restrict the circuits to be monotone, more is understood.

8.1 Karchmer-Widgerson Games

For a function $f : \{0,1\}^n \to \{0,1\}$, the Karchmer-Wigderson game R_f is a communication problem, where Alice gets an input x such that $f(x) = 1$, Bob gets y such that $f(y) = 0$, and the players wish to output some i where $x_i \neq y_i$.

For monotone f, here is also a *monotone* version M_f of the Karchmer-Wigderson game. Alice gets $x \in f^{-1}(1)$, Bob gets $x \in f^{-1}(0)$, and the goal is to output i s.t. $x_i = 1$ and $y_i = 0$.

Let $g_{v_i} : \{0,1\}^n \to \{0,1\}$ be the function computed by gate v_i. Note that

- If v is an **AND** gate, then
$$g_v(x) = g_{v_1}(x) \wedge g_{v_2}(x)$$
For Alice $g_{v_1}(x) = g_{v_2}(x) = 1$. For Bob $g_{v_1}(x) = 0$ or $g_{v_2}(x) = 0$.
- If v is an **OR** gate, then
$$g_v(x) = g_{v_1}(x) \vee g_{v_2}(x)$$
For Alice $g_{v_1}(x) = 1$ or $g_{v_2}(x) = 1$. For Bob $g_{v_1}(x) = g_{v_2}(x) = 0$.

In this way, a circuit computing f gives the following protocol for the K-W game R_f. The players walk down the circuit towards an input node, maintaining the invariant that at the current gate v, $g_v(x) = 1 \neq g_v(y) = 0$.

To maintain this invariant, let Alice own all $\vee$-gates, and Bob own all $\wedge$-gates. When players reach an $\vee$-gate, Alice announces a child that outputs 1 on x. Similarly, when players reach an $\wedge$-gate, Bob announces a child that outputs 0 on y. In this way, Alice and Bob reach an input node where $x_i = 1$ and $y_i = 0$, thus $D(R_f) \leq d(f)$.

It's possible to show the converse also holds—given a communication protocol for R_f, it's possible to construct a depth-efficient circuit for f. Furthermore, the same simulations hold when using a monotone f and monotone circuits.

Theorem 102. *$D(R_f) = d(f)$. $D(M_f) = d_m(f)$.*

8.2 Relations

Karchmer-Wigderson games are an example of a class of communication problems called *relations*. Essentially, relations are like functions, except there are potentially multiple possible correct outpus.

Definition 103. A relation R is a subset $R \subseteq X \times Y \times Z$. In the communication version of computing a relation, Alice gets $x \in X$, Bob gets $y \in Y$, and they have to output z such that $(x, y, z) \in R$.

Note that K-W games and monotone K-W games can be expressed as
$$R_f = \{x, y, i) : x_i \neq y_i\}$$
$$M_f = \{x, y, i) : x_i = 1 \text{ and } y_i = 0\}$$

8.3 The Match Game

For the rest of this class, we'll present a strong polynomial lower bound for a monotone K-W game called *Match*, thereby getting polynomial depth lower bounds for monotone circuits computing it.

Definition 104. A *matching* is a set of disjoint edges in a graph.

Set $N := \binom{n}{2}$ where n is the number of vertices in a graph. Next, we define a function Match: $\{0,1\}^N \to \{0,1\}$. Think of the input as a N-bit string defining which edges are in the graph or not. E.g
$$x_{(i,j)} = \begin{cases} 1 & \text{if } (i,j) \in G \\ 0 & \text{otherwise} \end{cases}$$

Definition 105.
$$Match(G) = \begin{cases} 1 & \text{if G has matching} \geq \frac{n}{3} \text{ edges} \\ 0 & \text{if G has no such matching} \end{cases}$$

Theorem 106. *$d_m(Match) = \Omega(n) = \Omega(\sqrt{(N)})$.*

Proof. This proof has multiple subproblems.

Problem 1: K-W game M_{Match} : Alice gets graph G_A with *large matching* ($\geq \frac{n}{3}$) Bob gets graph G_B with no large matching. Output: Edge in G_A but not in G_B.

Fact 107. $D(Match) \leq d_m(Match)$

Problem 2: Relation M' $\boxed{n := 3m}$ Alice gets a graph which *is* a large matching. Bob gets a complete bipartite graph $S \times T$ with $|S| = m-1$, $|T| = n-(m-1)$. Output: edge in matching not in G_B.

Fact 108. $D(M') \leq D(Match)$.

Problem 3: Pair-Disjointness Relation $PD \subseteq X \times Y \times Z$, where

- $X = \{$list of m ordered pairs of elements from$\{1,\ldots,3m\}\}$
  example $P \in X$: $P =< (1,3),(4,9),(2,5) >$
- $Y = \{(m-1)$ element subsets $\subseteq \{1,\ldots,3m\}\}$
- $(P,S,i) \in PD$ if $P_i \cap S = \emptyset$ meaning that the ith pair doesn't intersect with S.

We also define a relation PD', which is like PD, exept with a promise that $|P_i \cap S \leq 1|$.

Fact 109. $D(M') \geq D(PD) \geq D(PD')$.

Problem 4: f: partial function version of PD Alice gets P Bob gets set T with m elements Promise: For all $|P_i \cap T \leq 1|$ $f(P,T) = \begin{cases} 1 & \text{if } |P_i \cap T| = 1 \text{ for all i} \\ 0 & \text{if } \exists \text{i s.t. } P_i \cap T = \emptyset \end{cases}$

Lemma 110. *$R^{Pub}_{\frac{1}{4}}(f) \leq 2(D(PD') + \log n)$*

Problem 5: Set disjointness on m coordinates

Lemma 111. *$R^{Pub}_{\frac{1}{4}}(DISJ) \leq R^{Pub}_{\frac{1}{4}}(f)$.*

Proof. DISJ protocol: Alice gets $x \in \{0,1\}^m$ Bob gets $y \in \{0,1\}^m$ *Alice* constructs $p = \{p_i\}$ as follows: $x_i \to (3i - x_i - 1, 3i)$ if $x_i = 1$ $p_i = (3i-2, 3i)$, if $x_i = 0$ $p_i = (3i-1, 3i)$ *Bob* maps $y_i \to (3i - y_i)$ if $y_i = 1 : (3i-1)$, if $y_i = 0 : (3i)$

Fact 112. $P_i \cap T = \emptyset$ iff $x_i = y_i = 1$

□

Pulling all the facts and lemmas together to make the proof of the theorem:

$$\begin{aligned} d_m(Match) &\geq D(Match) \geq D(M') \geq D(PD') \\ &\geq \frac{1}{2}\left(R^{Pub}_{\frac{1}{4}}(f) - \log n\right) \\ &= \Omega\left(R^{Pub}_{\frac{1}{4}}(DISJ) - \log n\right) \\ &= \Omega(m) = \Omega(n) = \Omega(\sqrt{N}) \end{aligned}$$

□

Lecture 9

Static Data Structure Problems

Scribe: NavidTalebanfard

A *static data structure* is one to which no update takes place. Thus our main concern in designing such data structures is the space needed to store them and the time that it takes for a query to be answered.

Example 113. (*Dictionary Problem*) We have a set $S \subseteq \mathbb{N} \times \mathbb{N}$ of pairs of keys and data that we would like to store, so that we can perform *look ups*, i.e., given x we want some y such that $(x, y) \in S$ if one exists, otherwise we want to get a report that there is no such y. We can use several methods or this problem using

1. Binary balanced search trees in $O(|S|)$ space and $\log(|S|)$ query time.
2. Sorted table give us the same bounds.
3. Arrays with keys as indices if we do not care about space. This gives a $O(\text{largest key})$ space and $O(1)$ query time.

A more precise definition of the dictionary problem goes as follows.

Definition 114. (*Cell Probe Model*). A set $S \subseteq \{0, 1, \ldots, 2^d - 1\} \times \{0, 1, \ldots, 2^d - 1\}$ with $|S| = n$ is to be stored. A (t, s, w) data structure is mechanism on a random access machine with word length w such that at most s memory of cells are used, and on every query at most t memory cells are scanned.

Using *perfect hashing* we can give a $O(n)$ space $O(1)$ time solution to this problem. Here we give a slightly worse solution with $O(n^2)$ space which could be modified into the linear space solution. We identify $\{0, 1, \ldots, 2^d - 1\}$ with the elements of $\mathrm{GF}(2^d)$. Assume that $n^2 = 2^k$ for some k. Then we partition $\mathrm{GF}(2^d)$ into n^2 parts $T_1, \ldots, T_{n^2}$ of equal size.

Claim 2. For every $S \subseteq \mathrm{GF}(2^d)$ there exist a and b such that the function f which maps x to the partition that contains $ax + b$ is injective on S.

Proof. Pick a and b uniformly at random from $\mathrm{GF}(2^d)$. Fix x and y such that $x \neq y$. We have

$$\begin{aligned}
\Pr[\exists i \text{ such that } ax + b, ay + b \in T_i] &\leq \sum_{i=1}^{n^2} \Pr[ax + b, ay + b \in T_i] \\
&= \sum_{i=1}^{n^2} \sum_{z,u \in T_i} \Pr\left[ax + b = z, ay + b = u\right] \\
&= \sum_{i=1}^{n^2} \sum_{z,u \in T_i} \Pr\left[a = \frac{z-u}{x-y}, b = z - \frac{z-u}{x-y}x\right] \\
&= n^2 \left(\frac{2^d}{n^2}\right)^2 \left(\frac{1}{2^d}\right)^2 \\
&= \frac{1}{n^2}.
\end{aligned}$$

Now by the union bound the probability that there exists a pair $x \neq y$ in S whose image falls in the same part is at most $\binom{n}{2}\frac{1}{n^2} \leq \frac{1}{2}$. Therefore there exists a and b such that f is injective on S. □

Now the solution follows easily. We just take an n^2 long array containing one piece of data for each key, and the query x is answered by looking at cell $T_{f(x)}$ in this array, which is computed by a hash function defined as above.

Using communication complexity we can prove lower bounds for several data structure problems. For the next problem we need a definition first.

Definition 115. Let $x, y \in \{0,1\}^d$. The *Hamming distance* between x and y denoted by $d(x,y)$ is defined as the number of coordinates on which x and y differ. A *Hamming ball* of *radius* r *around* x is $B_r(x) = \{y : d(x,y) \leq r\}$. The *density* of a set $S \subseteq \{0,1\}^d$ is define as $\mu(S) = \frac{|S|}{2^d}$.

Example 116. (*Hamming nearest neighbor*) Let $S \subseteq \{0,1\}^d$. Given x we want to know the closest element $y \in S$ to x, i.e., $\text{argmin}_{y \in S}\{d(x,y)\}$. Using communication complexity we show that $s = n^{O(1)}$ and $r = n^{o(1)}$ is not possible.

We look at decision version of this problem called λ-nearest neighbor. Alice is given a list S of n d-bit strings and Bob has $x \in \{0,1\}^d$. They want to know if there exists $y \in S$ such that $d(x,y) \leq \lambda$.

For a data strucure problem we consider the following communication game: Alice gets a query and Bob gets a set. Their objective is to find cooperatively the answer to the Alice's query with respect to the set that Bob holds. Note that if a data structure problem has a solution with parameters w, s and t then the communication could be done by Alice sending $t\lceil \log s \rceil$ bits and Bob sending tw bits—Bob can store his input in a data structure. Each message from Alice is a request for the contents of a cell. Each message from Bob returns these contents.

Let $f : X \times Y \to \{0,1\}$ be any function denoting a communication problem and let it be represented by a matrix M_X^Y. We say that M is (u,v)-*rich* if at least v columns contain at least u 0's.

Lemma 117. (Richness Technique). *Let M_X^Y be the matrix of some communication problem f that is (u,v)-rich. If there is a protocol in which Alice sends a bits and Bob sends b bits, then M contains an all-*0 *rectangle with dimensions $\frac{u}{2^a} \times \frac{v}{2^{a+b}}$.*

Proof. We prove the lemma by induction on $a+b$. For $a+b=0$ there is no communication and thus f is constant. Since M is (u,v)-rich we have $|X| \geq v$ and $|Y| \geq u$ and $f(x,y) = 0$ for all x and y, proving the claim.

For the induction step assume that Alice sends the first bit. Let X_0 and X_1 be the inputs on which she sends 0 and 1 respectively. Let f_1 and f_2 be the restrictions of f on $X_0 \times Y$ and $X_1 \times Y$, respectively. It is easy to see that either f_0 or f_1 is $(\frac{u}{2}, \frac{v}{2})$-rich. Without loss of generality assume that it is f_0. Note that by assumption f_0 has a protocol in which Alice sends $a-1$ bits and Bob sends b bits. Thus applying the induction hypothesis we have a 0-rectangle with dimensions $\frac{u/2}{2^{a-1}} \times \frac{v/2}{2^{a-1+b}}$. But these are the exact dimensions that we wanted and thus we are done.

Assuming that Bob sends the first bit we can define Y_0, Y_1, f_0 and f_1 analogously. But note that this time either f_0 or f_1 is $(u, \frac{v}{2})$-rich. We can now apply the induction hypothesis and get the desired rectangle. □

We prove the lower bound for the λ-nearest neighbor problem by showing that its matrix is rich and yet it does not have large 0-rectangles. This is done using the following combinatorial facts about the Hamming balls.

Fact 118. There exists $c = c(n,d) \simeq \frac{1}{\sqrt{2/\ln 2}}$ such that setting $\lambda = \frac{d}{2} - c\sqrt{d \log n}$ implies $\frac{1}{n} \leq \mu(B_\lambda) \leq \frac{2}{n}$.

Lecture 10

Data Structure Lower Bounds

Scribe: Søren Frederiksen

10.1 Recap

10.1.1 Hamming nearest Neighbor

Problem 119 (Hamming nearest Neighbor)**.** Represent $S \subset \{0,1\}^d$ where $|S| \leq n$ as a s-word datastructure with w-bits words. For our case let $w = d$, so queries "What is the closest point to x in S i Hamming distance" can be answered by inspecting at most t cells of the data structure.

Theorem 120. *No scheme can satisfy $s = n^{O(1)}$ and $t = n^{o(1)}$ for all word sizes.*

We can make a communication version of the problem which is clearly easier to solve, so a lower bound for that gives us a lower bound for the Hamming nearest Neighbor.

Problem 121 (λNN)**.** Alice gets a $x \in \{0,1\}^d$. (So $|X| = 2^d$) and Bob gets a list of size n of elements from $\{0,1\}^d$. (So $|Y| = 2^{nd}$). They have to output 1 if $\exists y \in S : \text{dist}(x, y) \leq \lambda$, else 0.

To prove the lower bounds for λNN we will need another result namely the Richness lemma.

Theorem 122 (Richness lemma)**.** *Let $M_{x\in X}^{y\in Y}$ be a communication matrix where every column contains an α-fraction 0's. If there exists a protocol for M such that Alice sends a bits and Bob b bits, then M contains an all-0 rectangle of dimensions $\frac{\alpha|X|}{2^a} \times \frac{|Y|}{2^{a+b}}$.*

Now let $B_r(x)$ denote the Hamming ball centered at x with radius r. We then have the following facts for $\omega(\log n) \leq d \leq n^{o(1)}$ and n sufficiently large:

Fact 123. $\exists c = c(n, d) \approx \frac{1}{\sqrt{\frac{2}{\ln 2}}}$ such that $\lambda = \frac{d}{2} - c\sqrt{d \log n}$ gives us that $\frac{1}{n} \leq \mu(B_\lambda) \leq \frac{2}{n}$.

Fact 124. $\mu(B_{0.4d}) \leq 2^{-0.01d}$

Fact 125. Let x_1, x_2 points in the Hamming space. If $\text{dist}(x_1, x_2) \geq 0.4d$ then $\frac{\mu(B_\lambda(x_1) \cap B_\lambda(x_2))}{\mu(B_\lambda(x_1))} \leq \frac{1}{n^{0.1}}$

The first two follows from respecively Sterlings formula and the Chernoff bound. The last one we will not prove, but it shows some interesting things about the Hamming space, even if we move away from a point by just a small margin, the balls around the point and the new point will have a very small intersection.

10.2 Proof for bounds of λNN

Proof of Theorem 154. We are going to apply the Richness lemma, so we want to show that are are alot of 0's in the communication matrix. First thing we notice is that each row contains at least an $\frac{1}{10}$ fraction of

0's. This is true because each row is given by $x \in \{0,1\}^d$ the input of Alice, then the row will filled with 1's and 0's according to

$$\begin{aligned}&\text{Pr[random list of size n does not contain any elements } \lambda\text{-close to x]}\\ &= (1 - \text{Pr[random element is } \lambda\text{-close to x]})^n = ((1-\frac{2}{n})^{n/2})^2 \equiv \left(\frac{1}{e}\right)^2 > \frac{1}{10}\end{aligned}$$

These equalities come from fact 123 and that $\lim_{n\to\infty}(1-2/n)^{n/2} = \frac{1}{e}$. Since we iterate all random lists, we will get this fraction for the number of 0's. Since we have this number of 0's in the rows we can get a bound of at least $\frac{1}{20}$ columns have at least $\frac{1}{20}$-fraction 0's in them. We get this by contradiction, since assume it's not true. But when we try to pack the 0's into $\frac{1}{20}$ columns filled with 0's, and the rest with less than $\frac{1}{20}$-fraction 0's we get.

$$(\frac{1}{20}2^{dn})2^d + (\frac{19}{20}2^{dn})\frac{1}{20}2^d = 2^d 2^{dn}(\frac{1}{20} + \frac{19}{20}\frac{1}{20}) = 2^d 2^{dn}\frac{39}{400} < \frac{1}{10}2^d 2^{dn}$$

We now just focus on$\frac{1}{20}$-fraction of columns that contain the threshold number of 0's, and we now want to show that there are no large 0-rectangles in our communication matrix. For this we will use a lemma:

Lemma 126. *Let $I \subset \{0,1\}^d$ such that $|I| = (n^{0.1}2^{-0.01d})2^d$ then $\mu(\bigcup_{x\in I} B_\lambda(x)) \geq \frac{1}{2n^{0.9}}$*

Proof of lemma. Greedily chose a subset $C \subset I$ such that all the points in C is $\geq 0.4d$ apart. By fact 124 we only remove $2^{-0.01d}2^d$ points at each time, so $|C| = n^{0.1}$. From fact 125 we now get that:

$$\mu(\bigcup_{x\in C} B_\lambda(x)) \geq \frac{1}{n} + \frac{1}{n}\left(1 - \frac{1}{n^{0.1}}\right) + \frac{1}{n}\left(1 - \frac{2}{n^{0.1}}\right) + \ldots \geq \frac{1}{2}\frac{1}{n}n^{0.1}$$

The first inequality coming from having $\frac{1}{n}$ to be the measure of the first point in C then from fact 125 the intersection between the first and second point is small, so the measure of the secondpoint is $\frac{1}{n}\left(1 - \frac{1}{n^{0.1}}\right)$ and so on. The second inequality comes from the fact that the sum is dominated by the right triangle with legs of length $\frac{1}{n}$ and $n^{0.1}$, giving the area $\frac{1}{2}\frac{1}{n}n^{0.1}$. Since this is the measure of a subset of I then the measure of I is even bigger. □

How big can a 0-rectangle now be? To answer this we look at how many lists that does not intersect a set of density $\frac{1}{2n^{0.9}}$. So let $Q = \bigcup_{x\in I} B_\lambda(x)$ obtained from the previous lemma, and look at the probability:

$$\text{Pr[random list of size } n \text{ does not intersect } Q] = \left(1 - \frac{1}{2n^{0.9}}\right)^n = \left(\frac{1}{e}\right)^{\frac{1}{2}n^{0.1}} < 2^{-\frac{1}{2}n^{0.1}}$$

So the maximum amount of columns is at most a $2^{-\frac{1}{2}n^{0.1}}$-fraction of the total. Which then gives us that there a no all-0 rectangles of dimensions $(n^{0.1}2^{-0.01d})2^d \times 2^{-\frac{1}{2}n^{0.1}}2^{nd}$. Using the richness lemma on this, we get after alot of calculations that

$$t > \min\left(\frac{0.01d - 0.1\log n}{\log s} - 6, \frac{0.5n^{0.1}}{\log s + w} - 5\right)$$

Now for the final part of the proof, we see at for example $d = 1000t$ that the first one becomes obviously the largest so $t > \frac{0.5n^{0.1}}{\log s+w} - 5$ which does not satisfy $t = n^{o(1)}$ □

10.3 Predessesor Problem

Problem 127 (Predessesor)**.** Store $S \subset \{0, 1, \ldots, 2^d - 1\}$, where $|S| \leq n$ as a s-word datastructure with w-bits words ($w = d$), so queries "What is $\max\{S \cap \{y | y \leq x\}\}$" can be answered for a given x by inspecting at most t cells of the data structure.

Beame and Fich showed that for $s = n^{O(1)}$, we get $t = O\left(\sqrt{\frac{\log n}{\log\log n}}\right)$, $t = \Omega\left(\sqrt{\frac{\log n}{\log\log n}}\right)$.

Exercise 128. Put numbers into the coming proof to obtain the above bounds.

Observation 129. If a datastructure problem has a protocol with parameters s, t, w then the corrosponding communication problem has a protocol with t rounds of communication where Alice sends $\log s$ bits and Bob sends w bits.

We now define the communication version of the problem

Problem 130 (Comm. Predessesor)**.** Alice gets $x \in \{0, \ldots, 2^{d-1}\}$ Bot gets $S \subset \{0, \ldots, 2^{d-1}\}$, $|S| \leq n$, each element is colored red/blue, and the goal is to find the color of $\max\{S \cap \{y | y \leq x\}\}$

We will use the following lemma which we will not prove, which has an equivalent version for Bob:

Lemma 131 (Round elimination)**.** *Let f be any communication problem. Let f^k_{Alice} be the problem where Alice gets $x_1, \ldots, x_k \in X$ as input and sends the first message, all her messages being of length a. Bob gets $y \in Y$ and $i \in [k]$, his messeges being of length b. The goal is to output $f(x_i, y)$.*

If there is a protocol for f^k_{Alice} with error probability δ, then there is a protocol for f where Bob sends the first message, but with the total number of messages one less, which has error probability $\delta + O\left(\sqrt{\frac{a}{k}}\right)$.

Note: This lemma holds even if Bob is additionally given $x_1, \ldots, x_{i-1}$.

We will now show two reductions: $g^k_{Alice} \rightsquigarrow g$: Alice gets $x_1, \ldots, x_k \in \{0, \ldots, 2^d - 1\}$, she computes a kd-bit string $\hat{x} := x_1 || \cdots || x_k$[1] Bob gets $S \subset \{0, \ldots, 2^d - 1\}$, $i \in [k]$, $x_1, \ldots, x_{i-1}$, and he constructs a set of kd-bit strings $S' = \{z | z = x_1 || \cdots || x_{i-1} || y || 0^{d(k-i)}$ for some $y \in S\}$. Then, Alice and Bob solve g^k_{Alice} by running a protocol for g where Alice speaks first. Note that in the protocol for g^k_{Alice}, inputs are d bit strings and sets of d bit strings. In g, the inputs are kd bit strings and sets of kd bit strings.

$g^k_{Bob} \rightsquigarrow g$: Alice gets $x \in \{0, \ldots, 2^d - 1\}$, $i \in [k]$, she computes a $(d + \log k)$-bit string $i || x$. Bob gets $S_1, \ldots, S_k \subset \{0, \ldots, 2^d - 1\}$, with $|S_j| = n$, he computes $S = \bigcup_j \{z | z = j || y$ for some $y \in S_j\}$. Then, they solve g^k_{Bob} by running a protocol for g on $(d + \log k)$-bit strings with Bob speaking first.

By chaining together the round elimination lemma with the two reductions described above, and by setting the parameters right we can get something non-trivial with 0 communication. This means that there originally was no protocol for f^k_{Alice} with these parameters, so there can be no protocol for the original problem with these parameters since that would be able to solve f^k_{Alice} or f^k_{Bob}.

[1] Notice that we use || for concatenation

Lecture 11

Data Structure Lower Bounds

Scribe: Jana Kunert, Gozde Kocabas Last time we saw a data structure lower bound technique (using communication complexity) which showed that if the space is polynomial in n, then the query time must be large. Those bounds did not distinguish between space n and n^2: in fact, the technique was not sensitive to polynomial factors in the space. However, some problems (such as the one we consider below – rectangle stabbing on the grid) have a trivial data structure if the space is allowed to be n^2. We show a strengthened version of the technique from the last lecture, which *does* distinguish between polynomial factors in the space. This strengthened version is due to Mihai Patrascu, and is explored in his paper "Unifying the Landscape of Cell-Probe Lower Bounds " (although the proof we show is a little different). In particular, for rectangle stabbing this technique shows that if the space is $n \cdot polylog(n)$ then the query time must be $\Omega(\log n / \log\log n)$. (Throughout this lecture, we assume that the word size is $w = \log n$ bits. The same proof in fact also holds for $w = polylog(n)$, and can be generalized for any word size.

11.1 Recap

Example 132. DS (DATA STRUCTURE)
Input: some information y, a query x
Output: $f(x, y)$.

We saw last time that if there exists a data structure that can answer the query x on the data set y in time t using s space, then there also exists a communication protocol over t rounds where Alice sends $\log s$ bits in each round and Bob sends w bits in each round. If we look closer at the resulting lower bound, we can find its main drawback:

$$\begin{aligned} C_{A\to B}: \quad & t\log s \geq \text{communication lower bound} \\ & \Leftrightarrow t \geq \frac{\text{communication lower bound}}{\log s} \end{aligned}$$

The time t is not sensitive to polynomial factors in space—if $s = n \cdot polylog(n)$ or $s = n^{1000}$, the resulting bound on t remains the same (up to constant factors).

11.2 k-to-1 reductions

To achieve a better lower bound, Patrascu introduced a k-to-1 reduction: Bob still gets a set of data y, but Alice now gets k queries $x_1, x_2, \cdots, x_k$ instead of only one query. Now we want to compute $f(x_1, y), f(x_2, y), \cdots, f(x_k, y)$.

Claim 3. If there exists a data structure with query time t, space s and wordsize w, then there exists a t-round protocol where Alice sends $\log \binom{s}{k}$ bits in each round and Bob sends wk bits in each round.

Proof. Alice will simulate the behavior of the querier in the data structure, while Bob will simulate the data structure itself. In each round of communication, Alice needs to make makes k cell probes. She can make this request to Bob using just $\log\binom{s}{k}$ bits of communication, simply be sending the set of all cells that she wants to probe. Bob replies with the contents of these cells, using wk bits. □

For $k = 1$ we still get the same bound as we had before:
Alice sends $\log\binom{s}{k} = \log\binom{s}{1} = \log s$ bits.
Bob sends $wk = w \cdot 1 = w$ bits.
But the closer k gets to s, the smaller $\log\binom{s}{k}$ gets compared to $k \log s$. The strength of the argument will come precisely from this difference between $\log\binom{s}{k}$ and $k \log s$.

We will extensively use the fact that $\log\binom{s}{k} \leq \log\left(\frac{es}{k}\right)^k = k \cdot \log(es/k)$. In our application below, we will be interested in $s = n \cdot \mathit{polylog}(n)$ while $k = n/\mathit{polylog}(n)$. For such s and k, $\log\binom{s}{k} \leq O(k \log\log n)$, while $k \log s$ is $\Theta(k \cdot \log n)$. The ratio of $\frac{\log n}{\log\log n}$ between the two expressions is the source of our good lower bound.

11.3 Lopsided Set Disjointness

Lopsided Set Disjointness if the following communication problem. It is parametrized by two parameters: N and B.

Example 133. LSD (Lopsided Set Disjointness)
Input: two sets S, T, where $S, T \subseteq [NB] = \{1, \cdots, NB\}$, $|S| \leq N$
Output: $S \cap T = \emptyset$?

Theorem 134 (Patrascu)**.** *If a randomized protocol solves* LSD*, then either*

$$C_{A\to B} \geq 0.1 N \log B$$

or

$$C_{B\to A} \geq NB^{0.1}$$

This holds even for protocols with two-sided error.

In fact, the constant 0.1 should be replaced by another, unspecified, constant. But for convenience we will just use 0.1. This lower bound is already proved in the paper of Miltersen, Nisan, Safra and Wigderson for the case of one-sided error. But Patrascu was the first to prove this for the case of two-sided error.

In the obvious protocol, the following communication is needed:

$$\begin{aligned}
C_{A\to B} &= \log\binom{NB}{\leq N} = \log\left(\sum_{i=0}^{N}\binom{NB}{i}\right) \leq \log\left(2\binom{NB}{N}\right) = O\left(\log\left(\binom{NB}{N}\right)\right) \\
&\log\left(\binom{NB}{N}\right) \leq N\log(eB) \approx N \log B \\
C_{B\to A} &= \log(2^{NB}) = NB
\end{aligned}$$

We now show a protocol that provides a tradeoff between the communication from Bob to Alice and the communication from Alice to Bob.

Partition the set $[NB]$ into N chunks of size B each. Assume for simplicity that each block contains exactly one of Alice's elements (this assumption can be removed with some further work, which we will not describe). Now, for each block, Alice sends the last d digits of her element in this block, and Bob replies by sending all of his elements in the block whose last d digits agree with the digits that Alice sent. We get

$$C_{B\to A} \leq \frac{B}{2^d} \cdot N$$

$$C_{A\to B} = dN$$

We now give some intuition behind the lower bound. First, consider the case where $N = 1$. In this case the lopsided set disjointness problem becomes simply the INDEX problem, and the lower bound indeed matches what we know about the INDEX problem. Now, for larger N, we can think about the same special case that we considered above, where there are N blocks of B elements each, and Alice has exactly one element from each block. We can see that the lopsided set disjointness problem is in fact a sort of direct sum on N copies the index problem: one for each block. The "combining function" in this direct sum if the OR function: we simply need to know whether for at least one of the blocks, the element of Alice is among the elements of Bob. Since we know that a direct sum of N copies of a problem is often (but not always) as hard as N times the complexity of the single problem, then we should not be surprised by the lower bound described in Theorem 134. The proof of this theorem is surprisingly complicated, and uses information theory.

11.3.1 Lopsided Set Disjoiness with Restricted Inputs

Example 135. LSD with restricted inputs
Input: two sets S, T, where $S, T \subseteq [NB]$, $|S| \leq N$ and S is in some family $\mathfrak{F} \subseteq \binom{[NB]}{\leq N}$
Output: $S \cap T = \emptyset$?

Theorem 136. *Assuming* $|\mathfrak{F}| \geq \binom{[NB]}{N}^{0.99}$ *(and N is large enough), then either*

$$C_{A\to B} \geq 0.05 N \log B$$

or

$$C_{B\to A} \geq N B^{0.05}$$

Proof. Let S, T be some inputs to non-restricted LSD.
Before the protocol starts, Alice and Bob agree on r naming schemes. Each scheme

$$\pi_i : [NB] \rightarrowtail [NB]$$

is a bijection. These schemes are chosen randomly.
Now when Alice gets S, she tries all naming schemes π_i until she finds one such that $\pi_i(S) \in \mathfrak{F}$ (where $\pi_i(S) = \{\pi_i(x) : x \in S\}$). She sends Bob the index i. Bob also applies π_i to his set B and then they run the restricted LSD on $\pi_i(S)$ and $\pi_i(B)$. □

Claim 4. Taking $r = \frac{\binom{NB}{N}}{|\mathfrak{F}|} \log \binom{NB}{N}$ random naming schemes suffices to get a protocol.

Proof. Fix some set S. Now look at the probability for a scheme $\pi_i(S)$ being in $\mathfrak{F}$:

$$Pr[\pi_i(S) \in \mathfrak{F}] = \frac{|\mathfrak{F}|}{\binom{NB}{N}}$$

$$Pr[\pi_i(S) \notin \mathfrak{F}] = 1 - \frac{|\mathfrak{F}|}{\binom{NB}{N}}$$

$$Pr[\forall i : \pi_i(S) \notin \mathfrak{F}] = \left(1 - \frac{|\mathfrak{F}|}{\binom{NB}{N}}\right)^r$$

Let $\lambda := \log \binom{NB}{N}$.
When $r = \frac{\binom{NB}{N}}{|\mathfrak{F}|}\lambda$, then

$$\left(1 - \frac{|\mathfrak{F}|}{\binom{NB}{N}}\right)^r \approx \frac{1}{e^{\lambda}}$$

We use the union bound when S ranges over all possible N-element sets, and get

$$Pr[\forall S \exists i \text{ s.t. } \pi_i(S) \in \mathfrak{F}] \geq 0.9$$

Exercise: Finish the proof. (It is useful to observe that Alice and Bob can choose the set of mappings deterministically rather than randomly, because of the probabilistic method). □

11.4 Data Structure Lower Bounds from Lopsided Set Disjointness

The data structure problem that we consider is the following.

Example 137. Rectangle stabbing on the grid.
Input: a set T containing $n \log n$ rectangles on a $n \times n$ grid
Query: a point x
Output: Is there a rectangle in S, that contains x?

Known solutions:

space	time	idea
$O(n)$	$O(n)$	
$O(n \log^2 n)$	$O(\log n)$	interval / range trees
$O(n^2)$	$O(1)$	precompute all queries

Theorem 138. *Any data structure that uses space $s \leq n \cdot polylog(n)$ must use time $t \geq \Omega\left(\frac{\log n}{\log \log n}\right)$ in the cell probe model.*

Proof. We first define the hard instances. Our instance will be "almost good" and at the end of the proof we will change it to be entirely good.

The hard instance is constructed as follows. We choose $\log n$ different tiling of the $n \times n$ grid. Each tiling partitions the grid into n non-intersecting rectangles of of size $2^i \times 2^{\log n - i}$. There are $\log n$ different integral values of i, and each one of them produces one of the tilings. Clearly, each point in the grid is included in at most $\log n$ rectangles. In the hard instance, the rectangles of the data set would be a subset of these rectangles. Intuitively, in this construction the rectangles are "independent" enough or "convoluted enough" that in order to find out for a certain point whether one of the $\log n$ rectangles that contain it are actually in the data-set T, we'd need to make, on average, one probe for each of those rectangles, giving a lower bound of $\Omega(\log n)$.

Now, let us look at the k-to-1 reduction, transforming the data structure problem into a communication problem. Alice gets k queries and Bob has a set of rectangles. Given an LSD instance S, T we will create a DS instance: Bob will get a set of rectangles and Alice will get k queries. The correspondence between elements of the universe $[NB]$ and the rectangles in all tilings is $NB = n \log n$: the rectangles correspond to the elements of the universe in a canonical one-to-one fashion, which is known to all parties in advance. Bob creates the data-set by taking his set T, and putting in the structure exactly those rectangles that correspond to the elements of T. Alice, on her side, takes her set S and tries to write it as a union of k disjoint "stars". A "star" is a set of all rectangles that contains a particular point: thus there are n^2 possible stars. If Alice's set is small, then arranging it into rectangles is difficult; but since Alice's set is quite large, then we will show later that she can rearrange it into a union of k stars quite often. Define the family $\mathfrak{F}$ to contain all the S's where Alice succeeds is writing her set as a union of stars. If Alice and Bob can answer all k queries, then they can decide whether the sets S and T intersect, because $S \cap T \neq \emptyset$ iff one of the queries returned "yes".

We set the parameters as follows.

$$k = \frac{N}{\log n}, \quad NB = n \log n, \quad B = \log^{100} n$$

Now, assume there is a data structure for rectangle stabbing with space s and time t. From the k-to-1 reduction result, we thus know that there is a communication protocol that solves k queries, where the communication from Alice to Bob is at most $tk\log(s/k)$, and the communication from Bob to Alice is at most kwt. (For simplicity, we hide multiplicative constants.) On the other hand, we know that this protocol also solves LSD with restricted inputs. Assume for now that the family $\mathfrak{F}$ is indeed large enough that we can apply Theorem 136. Then, we get that either

$$0.1N\log B \leq tk\log(s/k)$$

or

$$NB^{0.1} \leq tkw$$

Consider the second inequality. If it holds, then we get $t \geq NB^{0.1}/kw = B^{0.1}$, which will finish the proof. This inequality is always the less interesting one: even for larger word size, e.g. $w = polylog(n)$, we could fiddle with the parameters to make B large enough so that this inequality would give what we want. It is only a matter of setting the value of B appropriately.

So, from now on consider the case that the first inequality holds.

$$0.1N\log B \leq tk\log(s/k)$$

Notice that by our setting of parameters $s = n\cdot polylog(n)$ and $k = n/\ polylog(n)$, so $\log(s/k) = O(\log\log n)$, and also that $\log B = \Theta(\log\log n)$. By substituting the parameters and simplifying and hiding multiplicative constants, we get simply that

$$t \geq N/k$$

Which gives $t \geq \log n$, as we wanted.

All of the above holds as long as $\mathfrak{F}$ is large enough. Unfortunately, with our setting of parameters and choice of hard instance, $\mathfrak{F}$ is not quite large enough. To make it large enough, we need to choose the hard instance slightly differently: rather than taking $\log n$ tilings, we take $\log_B n$ tilings. Each tiling consists of Bn rectangles of sizes $B^i \times B^{\log_B n-i-1}$, and there is a total of $NB = Bn\log_B n$ rectangles overall. Each star now consists of $\log_B n$ rectangles, and we thus set $k = \frac{N}{\log_B n}$.

The calculations above now work in a similar way, and we get

$$t \geq N/k = \log_B n = \Omega(\log n/\log\log n)\ ,$$

as desired.

We still need to prove that the family $\mathfrak{F}$ is large. We will not give the details of this proof here, but we just remark that in fact it can be shown that any set S of N rectangles that contains exactly $N/\log_B n$ rectangles from each tiling, it holds that S can be written as a union of disjoint stars. This is proved in somewhat different language in Patrascu's paper. There are a lot of S's with this property, and thus $\mathfrak{F}$ is large.

□

Lower bounds for many other data structures problems were proved using similar reductions from lopsided set disjointness, both in Patrascu's paper and in other papers. All of these proofs show how restricted lopsided set disjointness (on family $\mathfrak{F}$) can be solved using a data structure task, and then they show that the family $\mathfrak{F}$ is large. Both parts are highly dependent on the specifics of the data structure problem.

Lecture 12

Data Structure Lower Bounds

Scribe: Søren Frederiksen, Jesper Sindahl Nielsen

In the past few lectures, we've looked at space/time tradeoffs for *static* data structures—i.e., we receive all of the data in advance, and want to store it in a way that efficiently supports queries. Today, we'll discuss *dynamic* data structures. In this kind of problem, the data structure must support both queries and *updates* to the structure. We'll see several problems whose query/update times are expected to be very high, as well as a very recent technique by Mihai Pătraşcu that might prove *polynomial* lower bounds for several dynamic data structures.

First, we'll see a dynamic data structure that does have efficient solutions.

Problem 139 (Dynamic partial sums)**.** Store an array $A[1,\ldots,n]$ of w-bit integers $x \in \{0,\ldots,2^w-1\}$ and support operations:

- update(k,x): Sets the k'th element to be x ($A[k]=x$).
- query(k): Returns $\sum_{i=1}^{k} A[i]$

There are two easy solutions to Dynamic Partial Sums:

1. Just store the array. The update time is $O(1)$, and the query time is $O(n)$
2. Store the prefix sums $B[k] := \sum_{i=1}^{k} A[i]$. The update time is $O(n)$, and the query time is $O(1)$.

Next, a few less trivial solutions:

3. Split the array into $\sqrt{n}$ blocks each of size $\sqrt{n}$. Now for each block, store the elements in the block as well as the sum of the elements. Updates can then be performed in $O(1)$ by changing the number and the sum of the block by the difference from the previous number. Queries can be performed in $O(\sqrt{n})$ by adding the stored sum of each block up until the block which contains k, then sum the elements of that block up to k.
4. Use a binary tree with the leaves being the elements of the array and the internal nodes containing the sum of the elements below it. Updates can be performed in $O(\log n)$ by going down in the tree towards the element when and adjusting the values along the path. Queries can be done similar in $O(\log n)$ by going done the tree towards the element, and whenever we take the right child, we add up the sum of the left child.

Notice, that the $\sqrt{n}$ solution can also be used to switch the time bounds in the same manner as the $O(n)$ solution.

12.1 Dynamic Data Structures

We will from here on use the following notation.

N : Size of input (compared to the last section this would be $n \cdot w$).

t_u : The update time.

t_q : The query time.

Note that when discussing the solutions to Dynamic Partial Sums, we didn't consider the space, only the update and query time. For most dynamic data structures problems, space is not the primary concern. Instead, we'd like tradeoffs between t_u and t_q. The goal will be to minimize $\max\{t_u, t_q\}$.

Fact 140. Dynamic partial sums requires at least $\tilde{\Omega}(\log N)$, where $\tilde{\Omega}$ means we ignore any $\text{poly}(\log\log N)$ factors.

This is still the best lower bound known for many dynamic data structures even though they are conjectured to be $\text{poly}(n)$, until a few months ago! As a news flash Kasper Green Larsen, a Ph.D. student here at Aarhus, recently proved a lower bound for dynamic range counting of $\tilde{\Omega}(\log^2 N)$.

Again, for many dynamic data structure problems, we expect $\max\{t_u, t_q\}$ to by polynomial in N, so we won't cover these lower bounds in class. Instead, we discuss a new approach that has come forth is due to Mihai Pătraşcu, using a problem he calls the multiphase problem.

12.2 Multiphase Problem

Problem 141 (Multiphase)**.** The problem consists of the following phases where the algorithm performs the given tasks in the given times. The goal is then to minimize τ:

I We receive k subsets $S_1, \dots, S_k \subset [n]$. We have $O(nk\tau)$ time to preprocess these.

II We receive another set $T \subset [n]$, now we have $O(n\tau)$ processing time.

III We receives an index $i \in [k]$. We must decide in time $O(\tau)$ whether $S_i \cap T = \emptyset$.

This problem reduces to several problems in dynamic data structures, so that a lower bound for this problem will imply a similar lower bound for those problems. Here are some examples:

1. Dynamic Reachability Maintain directed graph under insertion and deletion of edges such that queries "Does a path $u \rightsquigarrow v$ exist?" can be answered.
2. Dynamic Shortest path Maintain undirected graph under insertion and deletion of edges such that queries "What is the length of the shortest path $u \rightsquigarrow v$?" can be answered.
3. Subgraph Connectivity Maintain undirected graph under updates that can turn on/off nodes such that queries "Does a path $u \rightsquigarrow v$ using only nodes that are 'on' exist?" can be answered.

Conjecture 142. There exist constants $\gamma > 1, \delta > 0$ such that if $k = \Theta(n^\gamma)$ then any solution to Multiphase requires $\tau = \omega(n^\delta)$.

12.2.1 Multiphase to subgraph connectivity reduction

In this subsection, we show how to solve the Multiphase problem using a data structure for the Subgraph Connectivity problem. The idea is to create a layered graph.

- Vertices. Create a vertex in the graph:
  - for each S_i,

  - for each $j \in [n]$,
  - and a "sink" v.

- Edges
  - For each i and each $j \in S_j$, create an edge (S_i, j).
  - For each $j \in [n]$, create an edge (j, v).

Lets see how this works through the three phases.

I In Phase I,

  - Construct all the edges.
  - Preprocess G.
  - In the beginning, all vertices are "off" except vertex v.

II In the second phase, turn vertex j on for all $j \in T$.

III In the final phase, turn S_i on, and query (S_i, v).

Notice that there is a path from S_i to v if and only if $S_i \cap T \neq \emptyset$.

The input size, that is the size of the graph, is $O(n \cdot k)$. If we assume the Multiphase conjecture we get $N = O(nk) = \Theta(n^{1+\gamma})$. Next, let us analyse the time for each phase

- Phase I: Preprocessing in $O(n \cdot k \cdot \tau) = O(N)$
- Phase II: $O(n \cdot \tau)$ time gives $O(\tau)$ pr update.
- Phase III: One update and one query and we have $O(\tau)$ time.

If we can prove that $\tau = \Omega(n^{\delta})$ then we get

$$\max\{t_u, t_q\} \sim \Omega(n^{\delta}) = \Omega(N^{\epsilon}) \text{ for } \epsilon = \frac{\delta}{1+\delta}$$

One approach to proving this is through number-on-the-forehead communication complexity.

12.3 Number-on-the-forehead communication complexity (NOF)

In a typical NOF communication problem, there are k players and k inputs. Player sees all the input except for input x_i. You can imagine input x_i attached to player i's forehead. The goal is then to cooperatively compute the function $f(x_1, x_2, \ldots, x_k)$. Now we will see how this translates to the Multiphase problem.

12.3.1 The CM (communication multiphase) problem

The CM problem is a communication game with three players, Alice, Bob and Carol, who get the following inputs (on their foreheads):

- Alice gets an index $i \in [k]$.
- Bob gets $S_1, S_2, \ldots, S_k$.
- Carol gets $T \subseteq [n]$.

Again, these inputs are written on the players' foreheads, so each player sees the inputs they *don't* have on their foreheads. The goal is to compute

$$CM(i, S_1, \ldots, S_k, T) = \begin{cases} 1 & if S_i \cap T \neq \emptyset \\ 0 & otherwise. \end{cases}$$

The protocol will be as follows

1. Alice sends a single message to Bob of size $n \cdot M$ bits (privately).

2. Bob and Carol speak back and forth for M bits of communication in total.

3. Bob and/or Carol output answer $CM(i, S_1, S_2, \ldots, S_k, T)$

Conjecture 143. CM. There exists constant $\gamma > 1$, $\delta > 0$ s.t. if $k = \Theta(n^\delta)$ then any CM protocol requires $M = \Omega(n^\delta)$.

Since we are working with NOF communication it is interesting to see who can do which phases:
Alice sees $S_1, \ldots, S_k$ and T - can do first and second phase.
Bob sees i and T - can do second and third phase if someone tells him about the first phase.
Carol sees i and $S_1, \ldots, S_k$ - can do first and third phase if she can communicate with Bob.
Now we describe how a data structure solving Multiphase translates to the NOF protocol for CM. This is similar to when we worked with the cell probe model—the protocol will simply emulate the data structure, imagine every player as having the memory at different phases.

1. Alice computes phases I and II by herself.

2. Alice then sends all the cells (and their contents) changed in phase two to Bob.

3. Bob and Carol now have enough information together to do phase III. Bob will try to simulate phase III, and whenever the algorithm queries a cell that was only used in phase I, Bob asks Carol for the contents of that cell.

Phase three has at most $O(\tau)$ queries and each cell has w bits and a request is w bits long, which all in all gives $O(\tau \cdot w) = O(M)$ total communication. If $M = \Omega(n^\delta)$ and $w = O(\log n)$, then there exists some constant δ' such that $\tau = \Omega(n^{\delta'})$. In this way, the conjectured communication lower bound for CM implies a polynomial lower bound on $\max\{t_u, t_q\}$ for the Multiphase problem, which gives polynomial lower bounds for the other data structure problems.

Lecture 13

Streaming Algorithms

Scribe: Falk Altheide The material in class today is mostly based on the paper "The space complexity of approximating the frequency moments" by Alon, Matias and Szegedy ("AMS"), first presented at STOC 1996. This paper won the Gödel Prize in 2005 "for their foundational contribution to streaming algorithms" and has 918 citations according to Google scholar.

The idea behind streaming algorithms is, that you get an amount of data to process, that is to big to fit into memory at once. For example in the context of the internet (note that the AMS paper came out shortly after the internet became really popular), we are confronted with a huge amount of data.

13.1 The Basic Model

The Input is a *stream* $A = \langle a_1, \ldots, a_m \rangle$ of m *tokens* $a_i \in [n]$. The goal is to compute some function $f(A)$ on the stream using as little space as possible. Ideally this would be a logarithmic amount $O(\log m + \log n)$, but polylogarithmic $O(\text{polylog}(mn))$ is good, too.

Examples of fuction to compute are the median of the stream's tokens or the number of different items. Important functions to compute are also the following

Definition 144 (Frequency Moments)**.** For $i \in [n]$ let $m_i := |\{j : a_j = i\}|$. The kth *Frequency Moment* is

$$F_k = \sum_{i=1}^{n} m_i^k.$$

Example 145. $F_1 = \sum_{i=1}^{n} m_i = m$. F_1 is just the number of elements in the stream.

Example 146. $F_0 = \sum_{i=1}^{n} m_i^0$ is the number of Distinct Elements. (Note that $0^0 = 0$)

Coming back to the network traffic example, one could use frequency moments to analyse the incoming packages on a web server and detect various kinds of attacks: If source adresses are analysed and F_0, the numer of distinct IP adresses, suddenly goes down to just one or just a few, this hints to a (D)DoS attack. If you monitor the destination port of incoming packages and F_0 goes up from a very small number (for example only HTTP on port 80) to a very big one, it alerts you that someone is doing a port scan.

13.2 Reduction

Now we want to apply our knowledge about communication complexity to streaming algorithms. But how to turn streaming algorithm problems into CC problems?

Idea: Split the stream into two parts, one is Alice's input, one is Bob's.

Alice's part				Bob's part		
a_1	a_2	a_3	$\ldots$	$\ldots$	a_{m-1}	a_m

Alice and Bob can turn their input x, y into streams of elements. Then Alice simulates a streaming algorithm on the first half and sends the contents of her memory to Bob. Afterwards Bob simulates a streaming algorithm on the second half. This describes a one-pass streaming algorithm and leads to a one-way communication. A k-pass streaming algorithm (where it is allowed to process the stream k times) would lead to a $(2k-1)$-round protocol.

13.3 Approximation/Randomization needed

For the following, we assume that $m = \Theta(n)$.

One of the big messages to take from the AMS paper is that often both *approximation* and *randomization* are required for stream computation.

Theorem 147. *For any non-negative $k \neq 1$ any deterministic algorithm that outputs $\widehat{F_k}$ such that $\left|\widehat{F_k} - F_k\right| \leq 0.1 F_k$ requires linear space.*

Theorem 148. *For all non-negative $k \neq 1$ any randomized algorithm that outputs F_k exactly requires $\Omega(n)$ space.*

We'll prove Theorem 147 for the Distinct Elements problem.

Proof of Theorem 147. We reduce from the Equality problem (EQ). The proof will rely on the following fact.

Fact 149 (from coding theory). There exists a set $S \subseteq \{0,1\}^n$ such that

1. $|S| = 2^{\Omega(n)}$
2. $|x| = \frac{n}{4}$ for all $x \in S$ $\qquad$ ($|x|$ = Hamming weight = number of i such that $x_i = 1$)
3. $\forall x \neq y \in S : |x \cap y| \leq \frac{n}{8}$

Now, we create an EQ protocol only on strings in S. Call this version of equality EQ_S. It's easy to show that $D(\mathrm{EQ}_S) \geq \log|S| = \Omega(n)$, via the same technique that gives the lower bound for the standard EQ problem. Next, we'll reduce $EQ_S \to F_0$.

Alice gets x, puts i on stream for each $x_i = 1$

Bob gets y, puts i on stream for each $y_i = 1$

Example 150. $x = 00011000$, $y = 00001010$. Then $A = \boxed{4 \,|\, 5 \,\|\, 5 \,|\, 7}$.

If $x = y$ then $F_0(A) = \frac{n}{4}$, but $\widehat{F_0} \leq \frac{n}{4} + 0.1\frac{n}{4}$.

If $x = y$ then $F_0(A) \geq \frac{n}{4} + \frac{n}{8} = \frac{3n}{8}$, but $\widehat{F_0} \geq \frac{3n}{8} - 0.1\frac{3n}{8}$.

Conclusion: Alice and Bob use F_0 algorithm to create a protocol for EQ. The cost of the protocol is the space used for F_0. But since $D(EQ) = \Omega(n)$ we can conclude that space is $\Omega(n)$.

```
Alice creates her half of the stream
Bob creates his half of the stream
Alice and Bob simulate $F_0$ algorithm
if $\widehat{F_0}(A) < 0.3n$ then
  Bob outputs $x = y$
end if
```

```
if $\widehat{F_0}\left(A\right) > 0.3n$ then
    Bob outputs $x \neq y$
end if
```

□

13.4 Estimators for F_2

Definition 151. An (ε, δ)-*estimator* for F_2 is a randomised streaming algorithm that, with a probability $\geq 1 - \delta$, returns $\widehat{F_2}$ such that $\left|\widehat{F_2} - F_2\right| \leq \varepsilon F_2$.

Algorithm 2 A basic F_2 estimator

```
Let $H : [n] \to \{-1, +1\}$ be a uniformly random function (which means $\Pr\left[H\left(i\right) = +1\right] = \frac{1}{2}\forall i$).
$X \leftarrow 0$
for all $j = 1 \ldots m$ do
    $X \leftarrow X + H\left(a_j\right)$
end for
return $X^2$
```

Recall that $F_2\left(A\right) = \sum_{i=1}^{n} m_i^2$, with $m_i = \left|\{j : a_j = i\}\right|$ and note that

$$
\begin{aligned}
X^2 &= \left(\sum_{j=1}^{m} H\left(a_j\right)\right)^2 \\
&= \left(\sum_{i=1}^{n} m_i H\left(i\right)\right)^2 \\
&= \sum_{i=1}^{n} \left(m_i H\left(i\right)\right)^2 + \sum_{i \neq j} m_i m_j H\left(i\right) H\left(j\right) .
\end{aligned}
$$

Then

$$
\begin{aligned}
E\left[X^2\right] &= \sum_{i=1}^{n} m^2 E\left[H\left(i\right)^2\right] + \sum_{i \neq j} m_i m_j E\left[H\left(i\right)\right] E\left[H\left(j\right)\right] \\
&= \sum_{i=1}^{n} m_i^2 \qquad \text{(since } E\left[H\left(i\right)^2\right] = 1 \text{ and } E\left[H\left(i\right)\right] = E\left[H\left(j\right)\right] = 0\text{)} \\
&= F_2 \ .
\end{aligned}
$$

So we know that $E\left[X^2\right] = F_2$. To ensure that X^2 is a good F_2 estimator, we need to control the variance. To do that, we use Chebyshev's Inequality:

$$
\Pr\left[\left|X - E\left[X\right]\right| \geq \alpha\right] \leq \frac{Var\left[X\right]}{\alpha^2}
$$

In our case we need $\alpha := \varepsilon F_2$ so we get

$$
\Pr\left[\left|X - E\left[X\right] \geq \varepsilon F_2\right|\right] \leq \frac{Var\left[X\right]}{\varepsilon^2 F_2^2}
$$

.

The basic estimator has $Var\left[X^2\right] \leq 2F_2^2$. If we directly used Chebyshev's Inequality, then the right hand side above would become $\frac{2}{\varepsilon^2}$, which is too high to say anything meaningful. Instead, we do the following: Let $X_1, \ldots, X_s$ be s independent and identically distributed basic estimators. $\left(X_i \sim X^2\right)$

Let $Y := \frac{1}{s}\sum X_i$. Then $E\left[Y\right] = E\left[X^2\right] = F_2$ and $Var\left[Y\right] = \frac{Var\left[X^2\right]}{s}$ (exercise).
If we set $s := \frac{16}{\varepsilon^2} = O\left(\frac{1}{\varepsilon^2}\right)$, then $\Pr\left[\left|Y - E\left[Y\right]\right| \geq \varepsilon F_2\right] \leq \frac{1}{8}$.

In this case the amound of space needed is $s\log\left(n\right) = O\left(\frac{\log(n)}{\varepsilon^2}\right)$ (plus space to store randomness, which, because it turns out to be sufficient to have 4-wise independence, adds $O\left(\log n\right)$ space).

13.5 Gap-Hamming-Distance

If $0 \leq k \leq 2$, $k \neq 1$, one can approximate F_k in $O\left(\frac{\log n}{\varepsilon^2}\right)$ space. If $k > 2$ then F_k required $\tilde{\Theta}\left(n^{1-\frac{2}{k}}\right)$ space

In practice $\frac{1}{\varepsilon^2}$ dominates for $k \leq 2$. This can become a problem. If you want to have a precision of 0.99 ($\Rightarrow \varepsilon = 0.01$), this would result in a factor of 10000 for space.

It turns out that this $1/\varepsilon^2$ factor is necessary. The reduction comes from the Gap-Hamming-Distance problem.

Definition 152. Gap Hamming Distance (GHD)
Alice gets $x \in \{0,1\}^n$
Bob gets $y \in \{0,1\}^n$
Then $\mathrm{GHD}\left(x,y\right) := \begin{cases} 1 & \text{if } \Delta\left(x,y\right) \geq \frac{n}{2} + \sqrt{n} \\ 0 & \text{if } \Delta\left(x,y\right) \leq \frac{n}{2} - \sqrt{n} \\ * & \text{otherwise (any answer ok)} \end{cases}$

where Δ is the *Hamming Distance* $\Delta\left(x,y\right) = \left|x \oplus y\right|$ =number of i such that $x_i \neq y_i$

13.5.1 GHD History

- 2003: defined by Indyk and Woodruff
- 2004: Woodruff proved $R^{\rightarrow}\left(\mathrm{GHD}\right) = \Omega\left(n\right) \Rightarrow$ streaming requires $\Omega\left(\frac{1}{\varepsilon}\right)$ space
- 2009: Brody and Chakrabarti: k-round protocol requires $\frac{1}{2^{O\left(k^2\right)}}$ communication
- 2010: Brody, Chakrabarti, Regev, Vidick and de Wolf: At least $\Omega\left(\frac{n}{k^2\log k}\right)$ needed
- 2011: STOC, Chakrabarti and Regev showed $\Omega\left(n\right)$ for all k

Next class, we'll look at the reduction of GHD to a streaming problem, and we'll go over the BCRVW lower bound.

Lecture 14

Lower Bounds for Property Testing, Conclusion

Scribe: Søren Frederiksen, Jesper A. S. Nielsen

14.1 Streaming algorithms

We first recall some of the definitions from last time.

- **Hamming weight** $|x| = |\{i | x_i = 1\}|$.
- **Hamming distance** $\Delta(x,y) = |\{i | x_i \neq y_i\}|$.
- **Gap-Hamming-Distance**: $\mathrm{GHD}(x,y) = \begin{cases} 0 & \text{if } \Delta(x,y) \geq n/2 + \sqrt{n} \\ 1 & \text{if } \Delta(x,y) \leq n/2 - \sqrt{n} \\ * & \text{otherwise} \end{cases}$

We now present the following theorems

Theorem 153. *Any 1-pass (ϵ, δ) streaming algorithm for F_0 needs $\Omega(\frac{1}{\epsilon^2})$ space.*

Theorem 154. *$R_\delta(GHD) = \Omega(n)$.*

We will now prove that theorem 154 implies theorem 153. By making a reduction from GHD to F_0.

Proof. We create a stream with $m := 2n$ elements from $[N] := [2n]$ where we let Alice's ith element be $a_i \leftarrow 2i - x_i$ and Bob's ith element be $b_i \leftarrow 2i - y_i$. Then the stream $\langle a_1, ..., a_n, b_1, ..., b_n \rangle$ will have $n + \Delta(x,y)$ distinct elements.

For our GHD protocol set $\epsilon = \frac{1}{2\sqrt{n}}$ (note that $\frac{1}{\epsilon^2} = \Omega(n)$). Now run the (ϵ,δ)-stream algorithm for F_0 on the stream $\langle a_1, ..., a_n, b_1, ..., b_n \rangle$ and output $\mathrm{GHD}(x,y) = 1$ if $\hat{F}_0 \geq \frac{3n}{2}$.

From this we see that if $\mathrm{GHD}(x,y) = 1$ then $\Delta(x,y) \geq \frac{n}{2} + \sqrt{n} \Rightarrow F_0 \geq \frac{3n}{2} + \sqrt{n} \Rightarrow \hat{F}_0 \geq F_0(1-\epsilon) = \frac{3n}{2} + \sqrt{n} - \frac{1}{2\sqrt{n}}(\frac{3n}{2} + \sqrt{n}) \geq \frac{3n}{2}$. □

For the proof of theorem 154 we will instead prove another theorem that generalizes it.

Theorem 155. *For all $\epsilon < 1/50$, any k-round (μ, ϵ)-distributional protocol for GHD where μ is the uniform distribution on $\{(x,y) : GHD(x,y) \neq *\}$, necessarily has a message of length $\Omega(\frac{n}{k^4 \log^2 k})$.*

The proof of this theorem uses the round elimination lemma in a new form which we will prove afterwards

Lemma 156 (Round elimination). *$\forall k' \leq k$ if there exists a k'-round (μ, ϵ)-protocol for GHD, then there exists a $(k'-1)$-round (μ, ϵ')-protocol for GHD with $\epsilon' \leq \epsilon + \frac{1}{16k}$.*

Using this the proof of theorem 155 becomes very simple

Proof of Theorem 155. Fix k, assume all the messages are of length $C \approx \frac{n}{k^4 \log^2 k}$ where C ignores some constant factors. Now if there exists a k-round (μ, ϵ)-protocol, then there exists a $(k-1)$-round $(\mu, \epsilon + \frac{1}{16k})$-protocol. By repeating this argument k times we reach a 0-round $(\mu, \epsilon + \frac{1}{16})$-protocol, which gives us a contradiction, since any 0-round protocol must make mistakes on half the inputs (when inputs are distributed uniformly). □

Now we just need to prove the round elimination lemma.

Proof of REL. Let P be a k'-round (μ, ϵ)-protocol with the first message being C bits long, Then there exists a message $m \in \{0,1\}^C$ and a set $S \subseteq \{0,1\}^n$ such that $|S| \geq \frac{2^n}{2^C} = 2^{n-C}$ so Alice sends m on every $x \in S$.

Now fix this message m and set S. Now we construct the $(k'-1)$-round protocol Q as follows: Alice picks z such that $z = \text{argmin}_{z \in S} \Delta(x, z)$ and outputs $P(z, y)$. Notice that in this protocol we do not need to send the first message (m).

The hope of this protocol is that $\text{GHD}(x, y) = \text{GHD}(z, y)$ when $\Delta(x, z)$ is small, and that since $|S|$ is big then $\Delta(x, z)$ is small. From this we see that the possibility of error in the protocol Q comes from

(1) If the original protocol makes an error $P(z, y) \neq \text{GHD}(z, y)$.

(2) The distance is not small $\Delta(x, z) \geq t := \frac{n}{k^2 \log k}$.

(3) The distance is small $\Delta(x, z) \leq t$ but it does not match $\text{GHD}(x, y) \neq \text{GHD}(z, y)$.

The error probability of (1) is obviously ϵ. The one for (2) is $o(1)$ and for (3) it is $O(\frac{1}{k})$, these are more complicated and will be split into separate proofs. □

Proof of (2). For the proof of (2) we use the idea of isoperimetric inequalities. The idea is that of having two measures on an object, fix one of them and maximize or minimize the other by that constraint. The original isoperimetric inequality was for shapes on the plane and asked how much area can be enclosed in a shape with fixed perimeter P? The classic inequality states that the area is maximized when our shape is a circle.

Define $S_t = \{x | \Delta(x, z) \leq t \text{ for some } z \in S\}$. The version of isoperimetric inequality we will use is that we among all $S \subseteq \{0,1\}^n$ with fixed $|S|$ what is the minimum $|S_t|$.

Harpers inequality gives us that this happens when S is a Hamming ball of some radius r. But this then implies that S_t is also a Hamming ball, but of radius $r + t$. Recall that $|S| \geq 2^{n-C}$. Then $r \gtrsim \frac{n}{2} - \sqrt{C \cdot n} \approx \frac{n}{2} - \frac{t}{2}$ if we chose the constants of C correctly. From this we also see that S_t becomes the hamming ball with radius $\frac{n}{2} + \frac{t}{2}$. From this we get that

$$\Pr[x \notin S_t] \leq \Pr[x \in B_{n/2 - t/2}] \leq \frac{2^{n-C}}{2^n} = o(1)$$

The first equality comes from the fact that the complement of a hamming ball of radius $r + t$ is again a hamming ball now with radius $n - (r + t)$ which is less than or equal to $\frac{n}{2} - \sqrt{Cn}$.

□

Proof of (3). This relates to the drunkard's walk problem. Imagine a drunk walking home from a bar and with probability 1/2 he takes one step left and with probability 1/2 he takes one step right, and this continues. We'd like to know how far from the bar the drunk finds himself after t steps. It turns out that with high probability, his position is $\Theta(\sqrt{t})$ away, either to the left to the right. Each case is equally likely. The important part is that the drunk is $\Theta(\sqrt{t})$ steps away from where he started.

The problem is essentially the same as having a t-bit string, w, where 1 means walk left and 0 means walk right. Then we get that the difference between the number of ones and zeros is $\Theta(\sqrt{t})$. Which is the same as saying that either $|w| = t/2 + \sqrt{t}$ or $|x| = t/2 - \sqrt{t}$.

Now back to the proof, assume that $\Delta(x,z) \leq t = O(\frac{n}{k^2 \log k})$ and assume wlog. that $GHD(x,y) = 0 \Rightarrow \Delta(x,y) \leq \frac{n}{2} - \sqrt{n}$.

Question: What is $Pr[GHD(z,y) = 1]$?

Fact: This probability is the same as if y had been fixed and z was random. (Homework 14)

Fact: $\Delta(x,y) = |x \oplus y|$.

Now we take a random z close to x. This gives us that $\Delta(z,y)$ will correspond to flipping t bits in $x \oplus y$ The probability is maximized in extremal cases. That is when $\Delta(x,z) = t$, $\Delta(x,y) = n/2 - \sqrt{n}$ which should be somewhat intuitively clear.

Now we want to look at $z \oplus y$ and find its Hamming weight. We know that $z \oplus y = (x \oplus y) \oplus (x \oplus z)$ and this is where the intuition from the drunkard's walk comes in. Essentially this corresponds to having a string $(x \oplus y)$ which we flip t random bits in, so now the question becomes how far away do we get from $x \oplus y$? Well w.h.p. we walk $\Theta(\sqrt{t})$ from it which means that w.h.p. we get

$$|z \oplus y| \leq \frac{n}{2} - \sqrt{n} + \Theta\left(\sqrt{t}\right) \leq \frac{n}{2} - \sqrt{n} + \Theta\left(\sqrt{\frac{n}{k^2 \log k}}\right)$$

The short answer for error (3) is a Chernoff-Hoeffding bound, which is a generalization of Chernoff bounds that we saw in the probability crash course.

□

14.2 Property Testing

Property testers are algorithms that solves a sort of decision problem on an object or tells that it is not possible for the object to attain the property by only doing a small amount of work. The algorithm essentially wants to decide if the object has the property or is far from having the property, for an appropriate definition of "far".

Example 157. Given a graph $G = (V, E)$ where $|V| = 500.000.000$ and the property *triangle freeness*: no three vertices are all connected.

The model is that we can query the graph for membership with regards to edges, that is we can query G is $(i,j) \in E$? And the graph then replies. We want to prove lower bounds on the query complexity and for the *triangle freeness* problem we have a query complexity of $\mathcal{O}(\phi(\frac{1}{\epsilon}))$, for some function ϕ which depends on $1/\epsilon$, *but not the size of the graph.*

Why do people care about these things? Social networking sites like to analyze cliques in graphs which triangle freeness is an instance of. Also advertisement companies might like to do some data mining where these things can be useful.

A new technique for proving property testing lower bounds is due to Blais, Brody and Matulef and can be found in the paper Property Testing Lower Bound via Communication Complexity (appeared in CCC 2011). The objects they deal with are functions on the form: $f : \{0,1\}^n \to \{0,1\}$.

Example 158. K-linearity.

f is linear if $\exists S \subseteq [n]$, $S = \{i_1, i_2, \ldots, i_{|S|}\}$ such that $f(x) = x_{i_1} \oplus x_{i_2} \oplus \cdots \oplus x_{i_{|S|}}$, $\forall x \in \{0,1\}^n$. We say that f is k-linear if $|S| = k$ and we say that f is ϵ-far from k-linear if:

$$\forall k\text{-linear functions } g, Pr[f(x) \neq g(x)] \geq \epsilon$$

Before applying the new technique the lower bound was $\Omega(\sqrt{k})$ and the new lower bound will give $\Omega(k)$.

Proof. We will reduce from $(\frac{k}{2})$-disjointness. Alice gets $A \subseteq [n]$ with $|A| = k/2$. Bob gets $B \subseteq [n]$ with $|B| = k/2$. We have a promise that the intersection intersects at exactly one point or not at all: $|A \cap B| \leq 1$

Alice: Turns A into $Lin_A(x) := \bigoplus_{i \in A} x_i$

Bob: constructs $Lin_B(x) := \bigoplus_{i \in B} x_i$

Now they must decide if $Lin_{A \Delta B} = Lin_A(x) \oplus Lin_B(x)$ is k-linear or not. Because of the following facts the reduction is done.

- $A \cap B = \emptyset \Rightarrow Lin_{A \Delta B}$ is k-linear
- $|A \cap B| = 1 \Rightarrow Lin_{A \Delta B}$ is $(k-2)$-linear.
- $(k-2)$ linear functions are 1/2-far away from k-linear functions.

□

If we had $[n] = [8]$, $A = \{3, 4, 7\}$ and $B = \{1, 2, 4\}$ then we get $Lin_A(x) = x_3 \oplus x_4 \oplus x_7$, $Lin_B(x) = x_1 \oplus x_2 \oplus x_4$ and we get $Lin_{A \Delta B} = x_1 \oplus x_2 \oplus x_3 \oplus x_7$.