The million dollar question in computational complexity is "do all problems in NP have efficient algorithms?" The general consensus is "probably not", but a proof is still lacking after more than 50 years of research. In the first four lectures we gave examples of computational hardness in some restricted models of computation: decision trees, small depth circuits, restricted branching programs, and monotone circuits. If we could generalize these methods to prove that, for example, solving satsifiability of 3CNF formulas on n inputs requires circuits of size $2^{n^{1/100}}$, we would have an example of an NP problem that does not have small circuit families and therefore no efficient algorithms either.

There is an interesting explanation about why this seemingly promising approach has failed to yield consequential results. The reason is that proofs of computational hardness are often *learning algorithms* in disguise. The learning algorithm gets oracle access to the circuit and deduces certain properties that the function computed by the circuit must satisfy. Many proofs of computational hardness do in fact show that all functions computed by the circuit model in question are efficiently learnable (in a precise sense that we will define shortly).

On the other hand, it is widely believed that there exist efficiently computable functions that are not efficiently learnable. Such functions are called *pseudorandom functions* and can be constructed from sufficiently strong pseudorandom generators. So if there was a proof that 3SAT on n inputs requires circuits of size $2^{n^{1/100}}$, either this proof would look very different from, say, the proof that parity on n bits requires depth 3 circuits of size $2^{n^{1/3}}$, or else pseudorandom functions do not exist, everything is learnable efficiently, and there is no cryptography.

1 Smallness and constructivity

The circuit lower bounds we proved in lectures 1, 2, and 3 all followed the same basic pattern. First we specified a property that all functions in the circuit class must have. Then we argued that the hard function does not have this property.

For example, to show that PARITY on n inputs requires a decision tree of depth $d = n/2$, we used the following property of depth d decision trees: After fixing some d of the variables to suitably chosen values the function computed by the decision tree becomes a constant. $PARITY$ does not have this property.

To show that PARITY requires decision trees of size $s = 2^{\Omega(n)}$, we used the fact that after applying a 1/2-random restriction a size s decision tree reduces to a depth less than d with probability at least $s \cdot (3/4)^d$. Setting d to equal $n/2$ reduces the problem of lower bounding the size to the solved problem of lower bounding the depth.

Smallness The property "$f\colon \{0,1\}^n \to \{0,1\}$ does not become constant after fixing some half of its inputs to some fixed values", which we denote by $P(f)$, certainly applies to the parity function, but it also applies (with high probability) to a *random* function R. This can be easily checked by a union bound: There are $\binom{n}{2}$ choices of which half of the variables to fix and $2^{n/2}$ possible fixings. For any such fixing, the probability that R becomes a constant is $2 \cdot 2^{-2^{n/2}}$ (as all $2^{n/2}$ values must

be zero or one). Therefore

$$Pr[P(R) \text{ does not hold}] \leq \binom{n}{2} \cdot 2^{n/2} \cdot 2 \cdot 2^{-2^{n/2}} = 2^{-2^{\Omega(n)}}.$$

So our proof that parity requires size $2^{\Omega(n)}$ decision trees also proves that most functions on n inputs require size $2^{\Omega(n)}$ decision trees. Such properties are called *small* since the essential property of the circuit is only shared by a tiny fraction of all functions.

Definition 1. A property P of boolean functions $f\colon \{0,1\}^n \to \{0,1\}$ is *small* if the probability that a random function satisfies P is at most $1/3$.

The exact probability $1/3$ is not important. The proof that $PARITY$ on n bits requires depth d circuits of size $2^{\Omega(n^{1/(d-1)})}$ from Lecture 2 relied on a similar property: After restricting all but $n/(K \log s)^{d-1}$ inputs in a size s, depth d circuit, the circuit becomes a constant with constant probability. A calculation similar to the one we just did shows that the property "becomes constant after restricting some subset of $n - 2\log n$ inputs" is small. So the same proof shows that random functions require depth d circuits of size $2^{\Omega((n/\log n)^{1/(d-1)})}$.

The proof that the inner product modulo 2 function on $2n$ bits requires read-k-times randomized branching programs of width $w = \Omega(2^{n/2k})$ relied on the following property: For every function $f\colon \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ computable by a width w, read-k-times randomized branching program there exist set X and Y such that $|X| \cdot |Y| \geq 2^{2n}/2w^{2k}$ such that f is constant on a $2/3$ fraction of entries in $X \times Y$ (assuming error $1/3$). We then argued that the inner product function does not have this property as long as $|X| \cdot |Y|$ is at least $\Omega(2^n)$.

It is again not difficult to argue that the property "is almost constant on $X \times Y$ for every X, Y such that $|X| \cdot |Y| \geq 100n$" is small. By large deviation bounds, the probability that a random function is constant on a $2/3$ fraction of $X \times Y$ for any such fixed X and Y is at most 2^{-3n}. By a union bound the probability that there exist such a pair X, Y is at most $2^{2n} \cdot 2^{-3n} \leq 1/3$.

To summarize, all of the lower bound proofs we saw with the exception of the bound on monotone circuits for CLIQUE from Lecture 4 followed the same pattern: We described some property of functions, showed that it holds for all functions computed by small circuits, and then argued that it does not hold for the hard function in question. It also happens that the property does not hold for most functions, so the proof also shows that random functions are hard for the model in question. This is not surprising as we know by a counting argument that most functions should in fact be hard for any "reasonable" circuit model. It can be proved that any property based on a "formal complexity measure" — a concept we won't define — is small as long as it is not satisfied by all functions.

Constructivity Constructivity postulates that the relevant lower bound property is efficiently computable:

Definition 2. A property P of boolean functions $f\colon \{0,1\}^n \to \{0,1\}$ is *c-constructive* if, P can be computed in time 2^{cn} given the ability to evaluate f at inputs of its choice.

We think of c as a constant independent of n. Since f itself is an object of size 2^n, "polynomial in 2^n" means "polynomial in the size of f."

It is not at all clear that properties behind circuit lower bound proofs must be constructive; let us look at the above examples. The property "f becomes constant after restricting k of its variables" is constructive: There are $\binom{n}{k}$ ways to choose the restricted variables and 2^k values that they can

be restricted to, in which case all that needs to be verified is that the function is constant on the remaining $n-k$ inputs; for this we need to examine 2^{n-k} outputs of f. The running time of the brute force algorithm for this property is $\binom{n}{k}2^k 2^{n-k}$, which is at most quadratic in 2^n.

On the other hand, it is not clear if the property "f is constant on a $2/3$ fraction of entries of $X \times Y$ for some X, Y such that $|X| \cdot |Y| \geq K$" that we used to prove our branching program width lower bound is constructive. A brute-force algorithm for this property would need to go over all sufficiently large subsets X and Y of $\{0,1\}^n$. Each set ranges over $2^{\Omega(n)}$ possibilities, so the number of sets to verify is $2^{2^{\Omega(n)}}$.

Let's take a step back and remember how we proved that the inner product function has this property. We derived it from the stronger statement that

$$\left|\sum_{x,y\in\{0,1\}^n} p(x) \cdot (-1)^{\langle x,y\rangle} \cdot q(y)\right| \leq \sqrt{\frac{2^n}{|X||Y|}}$$

where p and q are the probability mass functions of the uniform distributions on the sets X and Y. The proof we gave in Lecture 3 in fact works not only for probability mass functions, but for any two functions p and q of ℓ_2 norm $1/\sqrt{|X|}$ and $1/\sqrt{|Y|}$, respectively. In fact, our argument from Theorem 14 in Lecture 3 more generally proves that

$$\left|\sum_{x,y\in\{0,1\}^n} p(x) \cdot (-1)^{f(x,y)} \cdot q(y)\right| \leq 2^{n/2} \cdot \sqrt{\sum_{x\in\{0,1\}^n} p(x)^2} \cdot \sqrt{\sum_{x\in\{0,1\}^n} q(x)^2}$$

for all functions $p, q\colon \{0,1\}^n \to \mathbb{R}$. This says that the largest singular value of the $2^n \times 2^n$ matrix $F(x,y) = (-1)^{f(x,y)}$ is at most $2^{n/2}$. Singular values of a matrix are computable in time polynomial in the size of the matrix, so stated in this way the property becomes constructive.

To summarize, the property $P(f) =$ "$F(x,y) = (-1)^{f(x,y)}$ has a singular value larger than $2^{\alpha n}$" is sufficient to prove that F requires read-k-times branching programs of width $\Omega(2^{\alpha n/k})$. The property P is constructive. It turns out that it is also large (although we won't prove it): The largest singular value of a random $2^n \times 2^n$ ± 1 matrix is $O(2^{n/2})$ with high probability.

2 Pseudorandom functions and natural proofs

A distribution over functions $\{F\colon \{0,1\}^n \to \{0,1\}\}$ is called (s,ε)-pseudorandom if for every circuit D of size at most s,

$$\Pr_F[D^F(1^n) = 1] - \Pr[D^R(1^n) = 1] \leq \varepsilon$$

where R is a uniformly random function from $\{0,1\}^n$ to $\{0,1\}$.

The requirement is that no small distinguisher D, which is allowed to query the function at inputs of its choice, can tell apart F from a random function with advantage better than ε. From a learning-theoretic perspective, pseudorandom functions are functions that are hard to learn from their input-output behaviour: Even after observing F at any number of inputs, it is hard for the distinguisher D to predict the value at some before unseen input with advantage better than ε.

Pseudorandom functions F that are computable by circuits of size polynomial in n are believed to exist for every n. We will next show how such functions can be constructed from sufficiently strong pseudorandom generators.

On the other hand, if polynomial-size circuits satisfy some property P that is both large and constructive, then they cannot compute pseudorandom functions:

Theorem 3. *If all functions $f\colon \{0,1\}^n \to \{0,1\}$ in some class $\mathcal{C}$ satisfy some property that is small and c-constructive then $\mathcal{C}$ cannot compute $(2^{cn}, 1/3)$-pseudorandom functions.*

Proof. Let D be the property in question. By smallness $\Pr[D^R(1^n) = 1] \leq 1/3$. Since all properties in $\mathcal{C}$ have the property, for any distribution on F in $\mathcal{C}$, $\Pr[D^F(1^n) = 1] = 1$. So

$$\Pr_F[D^F(1^n) = 1] - \Pr[D^R(1^n) = 1] \geq \frac{2}{3}.$$

Since the property is computable in time 2^{cn}, $\mathcal{C}$ cannot compute $(2^{cn}, 1/3)$-pseudorandom functions. □

We now show that polynomial-size circuits *can* compute $(2^{cn}, 1/3)$-pseudorandom functions for every c assuming sufficiently strong pseudorandom generators exist. To do this, it will be helpful to think of the distribution F as a family of functions $F_K\colon \{0,1\}^n \to \{0,1\}$, where K is some random "key" that indicates which function in the family is sampled.

We will in fact construct pseudorandom functions from $\{0,1\}^n$ to $\{0,1\}^k$ for some larger k. The most significant bit of such a function is a pseudorandom function from $\{0,1\}^n$ to $\{0,1\}$ with the same parameters.

3 Construction of pseudorandom functions

We will now show how to construct a pseudorandom function from a pseudorandom generator. Let's start with a pseudorandom function that takes *one* bit of input. In other words, we want a family of functions $F_K\colon \{0,1\} \to \{0,1\}^k$ whose output is indistinguishable from the output of a random function.

In this case the solution is really simple: The pseudorandom function is fully described by the pair of values $(F_K(0), F_K(1))$, and so it is sufficient that this pair be indistinguishable from a truly random pair. But we can interpret the pair $(F_K(0), F_K(1))$ as a pseudorandom string of length $2k$, which suggest the following construction: Take a pseudorandom generator $G\colon \{0,1\}^k \to \{0,1\}^{2k}$ and let $F_K(0) = G_0(K)$, $F_K(1) = G_1(K)$, where G_0 and G_1 denote the first m and last m bits of the output of G respectively.

How about a pseudorandom function on two bits $F_K\colon \{0,1\}^2 \to \{0,1\}^k$? We can do the same trick again: take a pseudorandom generator with $4k$ bits of output, which we divide into four blocks $F_K(00), F_K(01), F_K(10), F_K(11)$. But in fact it suffices to use a pseudorandom generator $G\colon \{0,1\}^k \to \{0,1\}^{2k}$ and set:

$$F_K(00) = G_0(G_0(K)) \quad F_K(01) = G_1(G_0(K)) \quad F_K(10) = G_0(G_1(K)) \quad F_K 11 = G_1(G_1(K)).$$

We can view this as a two-level construction. The first input of F_K determines if we take the left or the right part of the output of G. Next, we use this part as a seed and choose the left or the right part as output depending on the value of the second input.

How do we argue that F_K is pseudorandom? We can do it in two stages: First, we replace the inner application of G by a truly random string S_0S_1 of length $2k$ and argue that

$$\begin{array}{rcccc} H = & (G_0(S_0), & G_1(S_0), & G_0(S_1), & G_1(S_1)) \\ \text{is indistinguishable from} & (G_0(G_0(K)), & G_1(G_0(K)), & G_0(G_1(K)), & G_1(G_1(K)). \end{array}$$

But now we have a distribution H that is of the form $(G(S_0), G(S_1))$, where S_0 and S_1 are independent seeds, so by a hybrid argument its output will be indistinguishable from random. (It is a good exercise to complete the missing steps in this proof.)

This suggests the following general construction of a pseudorandom function $F_K \colon \{0,1\}^n \to \{0,1\}^k$ from a pseudorandom generator $G \colon \{0,1\}^k \to \{0,1\}^{2k}$:

$$\boxed{F_K(x_1 x_2 \dots x_n) = G_{x_n}(G_{x_{n-1}}(\dots G_{x_1}(K)\dots)).}$$

where G_0 and G_1 are the first k and last k bits of the output of G, respectively.

Theorem 4. *If G is a (s,ε)-pseudorandom generator, then $\{F_K\}$ is an $(\Omega(s/tn), sn\varepsilon)$ pseudorandom function family, where t is the circuit size of G.*

If $s = 2^{\Omega(k)}$, t is polynomial in k, and $\varepsilon = 1/3sn$ and k is a sufficiently large polynomial in n we obtain a pseudorandom function with the desired parameters.

To prove this theorem, it will be convenient to use an alternative characterization of pseudorandom generator: Here, we give the distinguisher oracle access to the output of G.

Lemma 5. *If $G : \{0,1\}^k \to \{0,1\}^{2k}$ is a pseudorandom against size s and bias ε, then for every circuit A of size s:*

$$\left|\Pr[A^{G(R_k)} = 1] - \Pr[A^{R_{2k}} = 1]\right| \le s\varepsilon,$$

where R_n is an oracle that returns a random string of length n on every invocation.

Proof. Suppose there for some $A^?$ of size at most s (and therefore makes at most s oracle queries)

$$\left|\Pr[A^{G(R_k)} = 1] - \Pr[A^{R_{2k}} = 1]\right| > s\varepsilon.$$

We apply a hybrid argument. Consider the hybrid oracle H_i that answers its first i queries as $G(R_k)$ and the other $s-i$ queries as R_{2k}. Then there must exist some i such that

$$\left|\Pr[A^{H_{i-1}} = 1] - \Pr[A^{H_i} = 1]\right| > \varepsilon.$$

Since the oracle answers are independent, the following circuit B is a distinguisher for G:

> B: On input z, simulate A by answering its first $i-1$ queries as $G(X_j)$ for a random string $X_j, 1 \le j \le i-1$, its ith query by z, and its last $s-i$ queries as Y_j for a random string $Y_j, i+1 \le j \le s$.

Then $B(G(X))$ is identically distributed with $A^{H_{i-1}}$, while $B(Y)$ is identically distributed with A^{H_i}, and so

$$\left|\Pr[B(G(X)) = 1] - \Pr[B(Y) = 1]\right| \ge \varepsilon.$$

By fixing the optimal choices of X_j and Y_j and hardwiring them into B, we can get a circuit B of size s that performs the distinguishing. □

We can now prove Theorem 4.

Proof of Theorem 4. Suppose that for some A of size $s' = \Omega(s/tn)$,

$$\left|\Pr\left[A^{F_K} = 1\right] - \Pr\left[A^{R} = 1\right]\right| \ge sn\varepsilon.$$

Consider the following family of hybrid functions $H_0, \dots, H_n$:

$H_i(x) = G_{x_n}(\cdots G_{x_{i+1}}(R(x_i \ldots x_1))\cdots)$, where $R\colon \{0,1\}^i \to \{0,1\}^k$ is a random function.

Notice that H_0 is exactly the distribution F_K, while H_n is a random function from $\{0,1\}^n$ to $\{0,1\}^k$.

By the hybrid argument, there must exist an index i such that

$$\left|\Pr[A^{H_{i-1}} = 1] - \Pr[A^{H_i} = 1]\right| > s\varepsilon.$$

By Lemma 5, to show that G is not pseudorandom it is sufficient to construct a circuit $B^?$ of size s so that

$$\left|\Pr[B^{G(R_k)} = 1] - \Pr[B^{R_{2k}} = 1]\right| > s\varepsilon$$

To do this, notice that the functions H_{i-1} and H_i differ only in what happens at level i. In H_{i-1}, the inputs chosen at this level look like the outputs of G, while in H_i they look random. Intuitively, if we can distinguish between H_{i-1} and H_i, we should be able to distinguish random and pseudorandom strings of length $2k$.

The distinguisher B will do the following:

B^O: Simulate the circuit A. When A makes its jth query x,
  If this is the first query of A with prefix $x_1 \ldots x_{i-1}$,
    Query the oracle O to get a string $z_0 z_1 \in \{0,1\}^{2k}$.
    Answer A's query by $G_{x_n}(\cdots G_{x_{i+1}}(z_{x_i})\cdots)$
    and memorize the pair $(x_1 \ldots x_{i-1}, z_0 z_1)$.
  Otherwise,
    Find the previously memorized pair $(x_1 \ldots x_{i-1}, z_0 z_1)$.
    Answer A's query by $G_{x_n}(\cdots G_{x_{i+1}}(z_{x_i})\cdots)$.
Return the output of A.

As this simulation goes along, B^O dynamically builds a random function $F\colon \{0,1\}^n \to \{0,1\}^k$. By construction, if O is the oracle $G(R_k)$, then F is distributed like H_{i-1}, and if O is the oracle R_{2k}, then F is distributed like H_i. It follows that

$$|\Pr[B^{G(R_k)} = 1] - \Pr[B^{R_{2k}} = 1]| = |\Pr[A^{H_{i-1}} = 1] - \Pr[A^{H_i} = 1]| > s\varepsilon.$$

The size of B is at most $O(tn)$ times the size of A (every time A calls its oracle, B performs at most n evaluations of G, each of which takes circuit size t), which is at most s by our choice of parameters. By Lemma 5, G is not (s,ε)-pseudorandom. □

References

Natural proofs were introduced and studied by Razborov and Rudich. Their paper contains an extensive study of all circuit lower bounds proved before 1995 and shows they all rely on constructive and small properties (with the exception of the monotone circuit lower bounds such as the ones from our Lecture 4). Theorem 4 is due to Goldreich, Goldwasser, and Micali.