# 13 Circuit Complexity (April 24–May 6)

In this final series of lectures, we'll look at one of the most basic models of computation: Boolean logic circuits. Formally, a boolean circuit is a directed acyclic graph. Nodes with indegree zero are *input nodes*, labeled $x_1, x_2, \ldots, x_n$. A circuit has a unique node with outdegree zero, called the *output node*. Every other node is a *gate*. Each gate with indegree $d$ is labeled with a function of the form $f : \{0,1\}^d \to \{0,1\}$. Unless otherwise specified, we will use only the standard binary AND and Or gates, and unary NOT gates. (Later, we'll allow AND and Or with fan-in larger than 2, but not yet.)

Any boolean circuit with $n$ input nodes *realizes* some $n$-ary boolean function. The *circuit complexity* of a function $F : \{0,1\}^n \to \{0,1\}$, denoted $C(F)$, is the minimum number of gates in any realization of $F$.

**Motivation.** Obviously, building smaller circuits saves money, but a much more important motivation for studying this model is its connection to the P = NP question. Consider a Turing machine that computes a string recognition function $F : \{0,1\}^* \to \{0,1\}$. By restricting the input to strings of each possible length, we obtain a family of functions $F_n : \{0,1\}^n \to \{0,1\}$ for each $n$.

**Theorem 1.** *If $F \in$ P, then $C(F_n)$ is bounded by a polynomial in $n$. (The converse is not necessarily true!)*

Thus, to prove P $\neq$ NP, "all" we have to do is exhibit a problem in NP that does not have polynomial circuit complexity!

## 13.1 A Specific Function

Let $F_{k,n} : \{0,1\}^n \to \{0,1\}$ be the "at least $k$" function:

$$F_{n,k}(x_1, x_2, \ldots, x_n) = \left[\sum_{i=1}^{n} x_i \geq k\right].$$

What's the circuit complexity of this function? More concretely, what is the circuit complexity of the "at least two" function $C_{2,n}$? (This is the first nontrivial special case, since $C_{1,n}$ is equivalent to an $n$-ary Or, which clearly has complexity $n - 1$.)

A few base cases are obvious. Since $F_{2,1}$ is identically 0, its complexity is clearly also 0. Similarly, since $F_{2,2}$ can be realized by a single AND gate, its complexity is 1. Unfortunately, for general $n$, we don't have tight bounds.

**Theorem 2.** $C(F_{2,n}) \leq 3n - 4 + (n \bmod 2)$

**Proof:** Let $X = (x_1, x_2, \ldots, x_n)$ denote the vector of input bits, and for notational convenience, define $y = (x_1, \ldots, x_{n-2})$ and $z = (x_{n-1}, x_n)$. We have the following simple recurrence for $F_{2,n}(x)$:

$$F_{1,n}(x) = F_{1,n-2}(y) \vee F_{1,2}(z)$$
$$F_{2,n}(x) = F_{2,n-2}(y) \vee F_{2,2}(z) \vee (F_{1,n-2}(y) \wedge F_{1,2}(z))$$

This recurrence describes a circuit that computes both $F_{1,n}(x)$ and $F_{2,n}(x)$. If $C_n$ is the number of gates in this circuit, we have the recurrence

$$C_1 = 1, \qquad C_2 = 2, \qquad C_n = C_{n-2} + C_2 + 4,$$

whose solution is the required upper bound. □

Before we prove a lower bound, we quickly observe that without loss of generality, we can assume that the circuit has at most $n$ inverters, each attached to one of the input nodes. If any inverter appears after an AND or OR gate, we can push it ahead of the gate using De Morgan's laws:

$$\neg(x \wedge y) = (\neg x) \vee (\neg y) \qquad \neg(x \vee y) = (\neg x) \wedge (\neg y).$$

To prove a lower bound, it suffices to count the number of AND and OR gates.

**Theorem 3.** $C(F_{2,n}) \geq 2n - 3$

**Proof:** We prove the bound by induction on $n$; the base cases $n = 1$ and $n = 2$ are trivial.

Consider a gate $g$ in a minimal circuit for $F_{2,n}$ whose inputs are connected directly to (possibly inverted) input variables. Gate $g$ must take two *different* (possibly inverted) variables $x_i$ and $x_j$ as input; otherwise, the circuit would not be minimal.

I claim that one of $x_i$, $\overline{x_i}$, $x_j$, and $\overline{x_j}$ must occur as the input to at least one other gate. Intuitively, the circuit must 'know' whether the number of 1s between $x_i$ and $x_j$ is 0, 1, or 2, but one gate can't distinguish between the three possibilities. For example, if $g(0,1) = g(1,1)$ and neither $x_i$ nor $x_j$ is an input to any other gate, then setting $x_k = 0$ for all $k \neq i, j$ would lead to two inputs that the circuit can't distinguish, even though their outputs should be different. The remaining cases can be argued similarly.

Suppose $x_i$ or $\overline{x_i}$ occurs as an input to some other gate $g$. If we fix $x_i = 0$, we can eliminate at least two gates $g$ and $g'$ from the circuit. The simplified circuit computes the $(n-1)$-ary function $F_{2,n-1}(x_1, \ldots, \widehat{x_i}, \ldots, x_n)$.

By the inductive hypothesis, the resulting circuit has at least $2(n-1) - 3 - 2n - 5$ gates, so the original circuit must have at least $2n - 3$ gates, as claimed. □

## 13.2 Most Functions Are Hard, But We Don't Have Any Bad Examples

To the everlasting shame of theoretical computer scientists everywhere, there is no known *explicit example* of a family of functions $F_n : \{0,1\}^n \to \{0,1\}$ such that $C(F_n) - 3n = \Omega(n)$. The best lower bound known (as of late 1992) has the form $3n + o(n)$. However, we can prove by a simple counting argument that *most* boolean functions have *exponential* circuit complexity.

**Theorem 4 (Shannon).** *As $n \to \infty$, the fraction of $n$-ary functions with circuit complexity less than $2^n/3n$ tends to 0.*

**Proof:** There are exactly $2^{2^n}$ $n$-ary boolean functions.

The number of circuits with $t$ gates, assuming inverters have been pushed back to the inputs, can be upper-bounded as follows. We can number the gates in any circuit from 1 to $t$. Each gate has one of two types (AND or OR). Each of the inputs to a gate is either a constant 0 or 1, an input $x_i$, an inverted input $\overline{x_i}$, or the output of another gate; thus, there are at most $2 + 2n + t - 1$ possible gate inputs. It follows that the number of circuits with $t$ gates is at most $2^t(t + 2n + 1)^{2t}$. If $t = 2^n/3n$, then

$$\frac{2^t(t + 2n + 1)^{2t}}{2^{2^n}} = o(1).$$ □

## 13.3 Shannon's Bound is Tight

It is not too hard to show that Shannon's lower bound is tight up to constant factors. Before we prove the matching upper bound, let's prove something easier: any $n$-ary function can be realized

using $O(2^n)$ gates. We define a *binary-to-positional converter* $B_n : \{0,1\}^n \to \{0,1\}^{2^n}$ as a circuit that takes $n$ inputs and has $2^n$ outputs, exactly one of which is equal to 1. Specifically, if the input bits are $x_0, x_1, \ldots, x_{n-1}$, then the output bit $y_x$ equals 1 if and only if

$$x = \sum_{i=1}^{n-1} x_i 2^i.$$

It is an easy exercise (really!) to realize $B_n$ with $O(2^n)$ gates. We can then realize any function $F : \{0,1\}^n \to \{0,1\}$ by ORing together the outputs of $B_n$ corresponding to 1s in the truth table of $F$; this requires at most $2^n - 1$ additional OR gates.

**Theorem 5.** *There is a constant $c$ such that every $n$-ary boolean function has circuit complexity at most $2^n/cn$.*

**Proof:** Fix an $n$-ary function $F : \{0,1\}^n \to \{0,1\}$. Let $x = (x_1, x_2, \ldots, x_n)$ denote the input vector, and for some integer $k$ to be specified later, define $y = (x_1, \ldots, x_k)$ and $z = (x_{k+1}, \ldots, x_n)$.

We begin by constructing a $2^k \times 2^{n-k}$ truth table for $F$, where each row is specified by a possible value of $y$, and each column by a possible value of $z$. Suppose that there are only $t$ different column vectors in this matrix. The inequality $t \le 2^{n-k}$ is obvious; less obvious but still trivial is the inequality $t \le 2^t$. Let $F_i(z) = 1$ if column $z$ has the $i$th pattern and 0 otherwise. Similarly, let $G_i(y)$ be the function specified by the $i$th column vector. Then we can write

$$F(x) = F(y,z) = \bigvee_{i=1}^{t} G_i(y) \wedge F_i(z)$$

Suppose for the moment that all the 1s in this table are restricted to $s$ of the $2^k$ rows. This immediately implies that $t \le 2^s$. Under this assumption, we can build a circuit for $F$ as follows. The inputs are connected to two binary-to-positional converters $B_k$ and $B_{n-k}$, which requires $2^k + 2^{n-k}$ gates. For each $i$ between 1 and $t$, we add two trees of ORs connected by a single AND to compute $F_i(y) \wedge G_i(z)$; for each $i$, this requires $2^k + 2^{n-k} + 1$ gates. Finally, we connect all these with another tree of $t \le 2^s$ OR gates. The total number of gates used in this special case is at most $2(2^k + 2^{n-k}) + 2^s + 1$.

Now back to the general case. We can write any $n$-ary function $F$ as a disjunction of $2^k/s$ different $n$-ary functions, each of which has a truth table with at most $s$ non-zero rows. To construct a circuit for $F$, we build a circuit for each of its $s$-row compoments, sharing a single pair of binary-to-positional converters. The total gate count for this construction is at most

$$2^k + 2^{n-k} + \frac{2^k}{s}(2^k + 2^{n-k} + 2^s + 1) = 2^k + 2^{n-k} + \frac{2^n + 2^k(2^s + 2^k + 1)}{s}.$$

Finally, we are free to choose the parameters $k$ and $s$ to minimize this gate count. If we take

$$k = 2\lg n \quad \text{and} \quad s = n - 2\lg n,$$

then the total gate count is at most

$$\begin{aligned} n^2 + \frac{2^n}{n^2} + \frac{2^n + n^2(2^n/n^2 + n^2 + 1)}{n - 2\lg n} &= n^2 + \frac{2^n}{n^2} + \frac{2^{n+1} + n^4 + n^2}{n - 2\lg n} \\ &= O\left(\frac{2^n}{n}\right) \end{aligned}$$

□

## 13.4 Circuit Depth and Communication Complexity

In addition to the number of gates, another useful measure of circuit complexity is the *depth*, defined as the length of the longest path from an input node to the output node. Let $d(F)$ denote the minimum depth of any circuit that realizes any fucntion $F$.

There is a nice connection between circuit depth and the communication complexity of the following game.

Let $A$ and $B$ be fixed, non-empty, disjoint sets of $n$-bit strings. Alice is given a string $x \in A$, and Bob is given a string $y \in B$. The goal off the game is to find an index $i$ such that $x_i \neq y_j$. Such an index must exist, since $A \cap B = \varnothing$. Let $C(A, B)$ denote the minimum number of bits that must be transmitted to compute this index in the worst case.

**Theorem 6.** *$d(f) = C(F^{-1}(0), F^{-1}(1))$ for any nonconstant function $F$.*

**Proof (Upper Bound):** We prove the following more general claim by induction on $d(F)$: For any function $F$ and any sets $A \subseteq F^{-1}(0)$ and $B \subseteq F^{-1}(1)$, we have $C(A, B) \leq d(f)$. Specifically, given a circuit for $F$, we inductively construct a protocol whose cost is the depth of the circuit.

In the base case $d(F) = 0$, we have $F \equiv x_i$ or $F = \overline{x_i}$ for some index $i$. In this case, we do not need to transmit any bits, since $F(x) \neq F(y)$ implies that $x_i \neq y_i$, so $C(A, B)$ is also zero.

Otherwise, consider a minimum-depth circuit that computes $F$, without loss of generality without inverters except at the inputs. Suppose the last gate is an AND. (The OR case is similar.) The inputs to this gate are described by two functions $F_0$ and $F_1$. Alice has a string $x$ such that $F(x) = 0$, which implies that $F_0(x) = 0$ or $F_1(x) = 0$. Bob holds a string $y$ such that $F(y) = 1$, so $F_0(y) = F_1(y) = 1$. The protocol begins by Alice sending an bit $b$ such that $F_b(x) = 0$, and continues by recursively evaluating $F_b$, which by construction has depth at most $d(F) - 1$. □

**Proof (Lower Bound):** Given any disjoint nonempty sets $A$ and $B$, there is a function $F$ such that $F(x) = 0$ for all $x \in A$ and $F(y) = 0$ for all $y \in B$. We claim that there is at least one such function where $d(F) \leq C(A, B)$. Given a protocol for $A$ and $B$, we construct the function $F$ and a low-depth circuit that realizes it, as follows.

The construction proceeds by induction on $C(A, B)$. In the base case $C(A, B) = 0$, we must have $F(x) = x_i$ or $F(x) = \overline{x_i}$ for some index $i$, which implies that $d(F) = 0$.

Otherwise, consider a minimum-length protocol for the sets $A$ and $B$. Without loss of generality, suppose Alice transmits first. Her transmission splits $A$ into two subsets $A_0$ and $A_1$; by construction, $C(A_0, B) \leq C(A, B) - 1$ and $C(A_1, B) \leq C(A, B) - 1$. By the inductive hypothesis, there are two functions $F_0$ and $F_1$ such that

$$A_b \subseteq F_b^{-1}(0), \qquad B \subseteq F_b^{-1}(1), \qquad d(F_b) \leq C(A_b, B)$$

for any $b \in \{0, 1\}$. If we set $F = F_0 \wedge F_1$, we have

$$\begin{aligned} d(F) &\leq 1 + \max\{d(F_0), d(F_1)\} \leq 1 + C(A, B) - 1 = C(A, B) \\ A &= A_0 \cup A_1 \subseteq F_0^{-1}(0) \cup F_1^{-1}(0) \subseteq F^{-1}(0) \cup F^{-1}(0) = F^{-1}(0) \\ B &\subseteq F_0^{-1}(1) \cap F_1^{-1}(1) \subseteq F^{-1}(1) \end{aligned}$$

□