Today we will briefly talk about non-uniform complexity classes and Yao’s XOR lemma.

1 Non-uniform complexity classes

Definition 1 Let $C$ be a class of languages (e.g. $P$, $NP$) and let $a(n)$ be a length function (e.g. $\log n$). Define $C/a$ to be the class $C/a = \{L | \exists L' \in C \text{ and } "advice" \alpha_1, \alpha_2, ... \in \{0,1\}^\ast, |\alpha_n| \leq a(n) \forall n \text{ s.t. } x \in L \iff (x, |x|, \alpha_n) \in L'\}$.

Note that the advice string is the same for all inputs of a given length.

For example, $P/poly = \bigcup_c P/n^c$ is the set of languages computable via Boolean circuits of polynomial size (the polynomial advice corresponds to the polynomial description of the circuit).

Uniform vs. non-uniform computational model. We use non-uniform complexity classes when talking about non-uniform models of computation. In the uniform model, we have a uniform Turing Machine which does the same algorithm regardless of the size of the input, whereas in the non-uniform model we have a different algorithm for each input size.

We also showed in homework that randomness does not help in the non-uniform model, since we can hardcode random strings as advice. In fact, we showed that $P/poly = RP/poly$.

Can we hope to make statements like $P/1 = P$? Not really, since even $P/1$ contains undecidable languages. For example, consider the language $L = \{x \mid M_x \text{ halts on the empty string } \epsilon\}$. Then $L \in P/1$ trivially since the advice bit $\alpha_n$ could tell you the answer. (Never mind that we don’t know how to find $\alpha_n$, the fact that it exists is enough.) Nevertheless, these complexity classes are still interesting.

2 Yao’s XOR Lemma

Throughout this section, we consider functions $f : \{\pm 1\}^n \rightarrow \{\pm 1\}$ and when we say $\oplus$ (the XOR operation), we really mean multiplication in $\{\pm 1\}$.

Recall the following definitions of hard and hardcore from last time.

Definition 2 $f$ is $\delta$-hard on a distribution $D$ for size $g$ if for any Boolean circuit $C$ with at most $g$ gates,

$$\Pr_{x \sim D(\{\pm 1\}^n)}[C(x) = f(x)] \leq 1 - \delta.$$ 

Definition 3 Let $S \subseteq \{\pm 1\}^n$. Then $f$ is $\epsilon$-hardcore on $S$ for size $g$ if for every Boolean circuit $C$ with size at most $g$,

$$\Pr_{x \sim U_S}[C(x) = f(x)] \leq \frac{1}{2} + \frac{\epsilon}{2}.$$ 

Recall the following theorem (actually combination of two theorems from last time).

Theorem 4 If $f$ is $\delta$-hard for size $g$ on the uniform distribution and $0 < \epsilon < 1$, then there exists a $2\epsilon$-hardcore set $S$ for $f$ for size $g' = \frac{1}{4}\epsilon^2\delta^2g$ with $|S| \geq 2^n$. 

1
So, if we start with a function \( f \) which is a little hard to predict, we can obtain a small set \( S \) on which \( f \) is very hard to predict. From this, we can get a function \( f' \) which is hard to predict on the whole domain (actually, the Cartesian product of \( k \) copies of the domain). In summary,

\[
\delta - \text{hard} \rightarrow \delta'(\epsilon, \delta) - \text{hardcore measure} \rightarrow 2\delta' - \text{hardcore set} \rightarrow 2\delta' + 2(1 - \delta)^k - \text{hard on domain to the } k
\]

The function \( f' \) will simply be the XOR of \( k \) copies of \( f \). We will obtain this result immediately with Yao’s XOR lemma, whose precise statement is as follows. For notational convenience, define \( f^\oplus_k(x_1, \ldots, x_k) = f(x_1) \oplus \cdots \oplus f(x_k) \).

**Theorem 5** If \( f \) is \( \epsilon \)-hardcore for a set \( H \) of size at least \( \delta 2^n \) for size \( g \), then \( f^\oplus_k \) is \( \epsilon + 2(1 - \delta)^k \)-hardcore on \( \{\pm 1\}^{nk} \) for size \( g - 1 \).

**Proof** Assume not. Then there exists a circuit \( C \) of size at most \( g - 1 \) such that

\[
\Pr_{x_1, \ldots, x_n} [C(x_1, \ldots, x_n) = f^\oplus_k(x_1, \ldots, x_n)] \geq \frac{1}{2} + \frac{\epsilon}{2} + (1 - \delta)^k.
\]

Our plan will be to show that for any \( H \) such that \( |H| \geq \delta 2^n \), we will get a circuit \( C' \) with at most \( g \) gates which guesses \( f \) with probability greater than \( \frac{1}{2} + \frac{\epsilon}{2} \).

**Constructing \( C' \)**: Let \( A_m \) denote the event that exactly \( m \) of \( x_1, \ldots, x_k \) are in \( H \). Then \( \Pr_{x_1, \ldots, x_k} [A_0] \leq (1 - \delta)^k \), so \( \Pr_{x_1, \ldots, x_k} [\bar{A}_0] \geq 1 - (1 - \delta)^k \). Therefore,

\[
\Pr_{x_1, \ldots, x_k} [C(x_1, \ldots, x_k) = f^\oplus_k(x_1, \ldots, x_k) | \bar{A}_0] \geq \frac{1}{2} + \frac{\epsilon}{2}.
\]

By averaging, there exists \( m \in \{1, \ldots, k\} \) such that

\[
\Pr_{x_1, \ldots, x_k} [C(x_1, \ldots, x_k) = f^\oplus_k(x_1, \ldots, x_k) | A_m] \geq \frac{1}{2} + \frac{\epsilon}{2}.
\]

Now we give the circuit for \( f \) on input \( x \in H \). To randomly generate a random element of \( A_m \), one can do the following:

1. Pick \( x_1, \ldots, x_{m-1} \in_R H \)
2. Pick \( y_{m+1}, \ldots, y_k \in_R \overline{H} \)
3. Permute \( x_1, \ldots, x_{m-1}, x, y_{m+1}, \ldots, y_k \) via random permutation \( \pi \).

Denoting \( \mathbf{x} = (x_1, \ldots, x_{m-1}) \) and \( \mathbf{y} = (y_{m+1}, \ldots, y_k) \), we have

\[
\Pr_{x, y, \pi} [C(\pi(\mathbf{x}, \mathbf{y})) = f^\oplus_k(\pi(\mathbf{x}, \mathbf{y}))] \geq \frac{1}{2} + \frac{\epsilon}{2}.
\]

By averaging again, there exists \( \mathbf{x}, \mathbf{y}, \pi \) such that

\[
\Pr_x [C(\pi(\mathbf{x}, \mathbf{y})) = f^\oplus_k(\pi(\mathbf{x}, \mathbf{y}))] \geq \frac{1}{2} + \frac{\epsilon}{2}.
\]

Now, let \( b = C(\pi(\mathbf{x}, \mathbf{y})) \oplus f(x_1) \oplus \cdots \oplus f(x_{m-1}) \oplus f(y_{m+1}) \oplus \cdots \oplus f(y_k) \). Define the circuit \( C' \) which computes \( C(\pi(\mathbf{x}, \mathbf{y})) \oplus b \). Then

\[
\Pr_x [C'(x) = f(x)] \geq \frac{1}{2} + \frac{\epsilon}{2}
\]

and moreover \( C' \) has size at most \( g \), since \( C(\pi(\mathbf{x}, \mathbf{y})) \) can be computed without adding any more gates to \( C \) and XORing with \( b \) takes at most one more gate. ■