Theory of Computer Science to Msc Students, Spring 2007		
	Tirgul 6	
Lecturer: Shahar Dobzinski		*Scribe: Shahar Dobzinski*

1 The Problem

Consider the following problem: we are given a black box that represents a function $f : \{0,1\}^n \rightarrow \{0,1\}^n$. Each query $q \in \{0,1\}^n$ to the black box returns the value of $f(q)$. We are given that there exists some $\overrightarrow{s} \in \{0,1\}^n, s \neq \overrightarrow{0}$, such that for each $\overrightarrow{x} \neq \overrightarrow{y} \in \{0,1\}^n$, we have that $f(\overrightarrow{x}) = f(\overrightarrow{y})$ iff $\overrightarrow{x} = \overrightarrow{y} \oplus \overrightarrow{s}$. How many queries to the black box do we need in order to find $\overrightarrow{s}$?

It is an easy exercise to determine the exact number of queries that a deterministic algorithm must do. Suppose the algorithm makes t queries, that is, we know the values of f at $f(q_1), \ldots, f(q_t)$. If we have two different queries q_i, q_j such that $f(q_i) = f(q_j)$, then we clearly we have that $s = q_i \oplus q_j$. If for all two different queries q_i and q_j we have that $f(q_i) \neq f(q_j)$. Then clearly we have ruled out at most $\binom{n}{2}$ possible candidates for n. Since there are 2^{n-1} possible candidates for s in the first place, we get that any algorithm must make at least $O(\sqrt{2^n})$ queries in order to find s.

How many queries a *randomized* algorithm must make? Our goal in this class will be to develop the necessary machinery to answer this question. En route, we will learn a bit about game theory, linear programming, and randomization, and about the connection between them.

2 Yao's Principle and a Lower Bound

Theorem 1 (Yao's Principle) *The worst-case performance of the best randomized algorithm is equal to the average performance of the best deterministic algorithm on the worst distribution of the inputs.*

Yao's principle is a bit tricky to understand in a first reading, but is actually quite simple. We are interested in proving lower bounds on the performance of randomized algorithms. I.e., show that for *every* randomized algorithm A, there exists an input I such that the expected performance of A on I is at least t. Proving this directly is usually not an easy task. Yao suggests that instead, we will look at the worst distribution on the *inputs* that we can think of, and claim that every *deterministic* algorithm does not have, on average, a performance better than t on this set of inputs. By Yao's principle, this is a lower bound on the worst case performance of randomized algorithms.

2.1 An Application: A Lower Bound of $\Omega(\sqrt{2^n})$

Given Yao's principle, proving a lower bound for our problem is easy. First, notice that in order to completely define an instance for our problem we only have to specify a vector $\overrightarrow{s} \in \{0,1\}^n, s \neq \overrightarrow{0}$, and define the function f. Our distribution on the inputs will be construct a "random" f: select uniformly at random, a vector $\overrightarrow{s} \in \{0,1\}^n, s \neq \overrightarrow{0}$, select uniformly at random a set of 2^{n-1} values for f from $\{0,1\}^n$, and assign them uniformly at random to the elements in the domain of f (while making sure that $f(x) = f(y)$ iff $x = y \oplus s$). We will see that on average the best deterministic algorithm must do at least $\Omega(\sqrt{2^n})$ in order to find $\overrightarrow{s}$. By Yao's principle this means that the best randomized algorithm must do at least $\Omega(\sqrt{2^n})$ queries to find $\overrightarrow{s}$, which is what we wanted in the first place.

Let us show that that on average the best deterministic algorithm must do at least $\Omega(\sqrt{2^n})$ in order to find $\overrightarrow{s}$ on the defined distribution on the inputs. Suppose the algorithm is querying $x_1, ..., x_t$. Suppose that for each $i \neq j$, $f(x_i) \neq f(x_j)$. Observe that unless $t < \sqrt{2^n}$, we cannot know the value of $\overrightarrow{s}$. However, if for some $i \neq j$ we have that $f(x_i) = f(x_j)$, we can deduce the value of $\overrightarrow{s}$. What is the chance that for some $i \neq j$ we have that $f(x_i) = f(x_j)$? We will not do the exact calculation here, but it is quite easy to see that we need $\Omega(\sqrt{2^n})$ for the probability for a "collision" to be above $\frac{1}{2}$. The proof is similar to the proof of the birthday paradox[1].

Exercise: Show that this lower bound is tight. I.e., show that there exists a randomized algorithm that makes $O(\sqrt{2^n})$ queries and succeeds with probability which is at least $\frac{1}{2}$.

3 Yao's Principle via von-Neumann's Minimax Theorem

Let us now prove Yao's principle. We will do this by using game theory arguments. In a *2-players game* we have two players, Alice and Bob. A *strategy* in the game is simply an action that each player can play. Alice has n *strategies*, and Bob has m strategies. When Alice plays the strategy i and bob plays j, the payoff of Alice is a_{ij}, and Bob's payoff is $-a_{ij}$. Let A be the set of Alice's strategies, and B be the set of Bob's strategies. Here we assume that both A and B are finite.

Suppose Alice knows that Bob is going to play j. Alice's *best response* for j is her strategy i such that $i = \arg\max_k a_{kj}$. Similarly, define Bob's best response for strategy i played by Alice. An *equilibrium in pure strategies* is a pair of strategies (i, j) such that i is Alice's best response for j, and j is Bob's best response for i.

Most games do not exhibit equilibrium in pure strategies. However, we might consider the case where the plays play *mixed* strategies, that is, a probability distribution over their (pure) strategies. If Alice plays the mixed strategy $\overrightarrow{x}$, and Bob plays $\overrightarrow{y}$, then Alice's payoff is $\Sigma_{i\in A, j\in B} x_i y_j a_{ij}$, and Bob's payoff is $-\Sigma_{i\in A, j\in B} x_i y_j a_{ij}$. We define the best response and equilibrium in mixed strategies in a similar manner to before. The following theorem tells

[1] What is the probability that in a room with n people we will have two with the same birthday? If $n > \sqrt{365}$ then the probability is at least $\frac{1}{2}$.

us that an equilibrium in mixed strategies always exists:

Theorem 2 (von Neumann Minimax Theorem) *Any two players zero-sum games has an equilibrium in mixed strategies* $(\overrightarrow{x}, \overrightarrow{y})$. *Moreover, the payoffs are equal:*

$$\max_x \min_y \sum_{i,j} x_i y_j a_{ij} = \min_y \max_x \sum_{i,j} x_i y_j a_{ij}$$

We will soon prove the theorem, but first here is how to derive Yao's principle. Fix an input size n. Define the following game: let Alice be the algorithms player: her strategies are all possible deterministic algorithms (observe that there is only a finite number of them). Let Bob be the inputs player: his strategies are all possible inputs (again, only a finite number). Let a_{ij}, the payoff of Alice from "playing" the algorithm i while bob is "playing" the input j be the performance of i on j. By the minimax theorem, there is an equilibrium point $(\overrightarrow{x}, \overrightarrow{y})$ such that the payoff of both players is t. Suppose that Bob is playing $\overrightarrow{y}$. Observe that for each pure strategy i we have that $\Sigma_j y_j a_{ij} \leq t$ (if there is a strategy the provides a better payoff then $\overrightarrow{x}$ is not a best response for $\overrightarrow{y}$). In particular, each strategy played with non-zero probability in x obtains a payoff of exactly t. In other words, the best deterministic algorithm has an average performance of t on the worst probability distribution on the inputs. On the other hand, we can view $\overrightarrow{x}$ as a randomized algorithm. Similarly, for every randomized algorithm there is an input on which the randomized algorithm obtains a payoff of t. This is what Yao's principle says.

Proof (of the minimax theorem) Alice faces the following problem:

Maximize: c

Subject to:

- $\forall j$: $\Sigma_i x_i a_{ij} \geq c$
- $\Sigma_i x_i = 1$
- $\forall i : x_i \geq 0$

Write this problem as:

Minimize: $\Sigma_i x_i$

Subject to:

- $\forall j$: $\Sigma_i x_i a_{ij} \geq 1$
- $\forall i : x_i \geq 0$

We claim that the optimum of the first LP is the inverse of the optimum of the second LP. To see this, take a feasible solution $\overrightarrow{x}$ to the first LP with a value c and observe that $\frac{\overrightarrow{x}}{c}$ is a feasible solution to the second LP with a value of $\frac{1}{c}$. Similarly, take a feasible solution $\overrightarrow{x}$ to the second LP with a value $\frac{1}{c}$ and observe that $c \cdot \overrightarrow{x}$ is a feasible solution to the first LP with a value of c.

Similarly, Bob faces the following problem:

Minimize: d

Subject to:

- $\forall i$: $\Sigma_j y_j a_{ij} \leq d$
- $\Sigma_j y_j = 1$
- $\forall j : y_j \geq 0$

And as before the optimum of the following LP has a value that is inverse to the value of the previous LP:

Maximize: $\Sigma_j y_j$

Subject to:

- $\forall i$: $\Sigma_j y_j a_{ij} \leq 1$
- $\forall j : y_j \geq 0$

Now we observe that the second and the fourth LP's are the dual of each other. By the strong duality theorem, they both have the same optimum. The theorem follows. ■