Lower Bounds in Theoretical Computer Science, Fall 2013	Instructors: Xi Chen

Lecture 1: Introduction

September 3, 2013 — Scribe: Fernando Krell

1 First Lower Bound: Sorting through comparisons

$\texttt{Sort}_n$: Given a permutation $x = x_1..x_n$ of $[n]$ output a permutation π such that

$$x_{\pi(1)} < x_{\pi(2)} < \cdots < x_{\pi(n)}$$

We will analyse the number of comparison needed to sort the elements using a decision tree. Each node of the tree represent a comparison, and leafs represent the permutation computed.

Let t be a decision tree and x representing an instance. Lets define the following functions:

- $\texttt{cost}(t, x)$: # of comparison made along the path that traverse t using input x.
- $\texttt{cost}(t) = \texttt{max}_x\{\texttt{cost(t,x)}\}$.
- $D(P)$ minimum $\texttt{cost(t)}$ overall trees t that solves problem P.

Question: What is $D(\texttt{Sort}_n)$?

Theorem 1. $D(\texttt{Sort}_n) \geq \Omega(n \log n)$

Proof. We will use a *counting argument.*

If t solves $\texttt{Sort}_n$ for every possible input then t has at least $n!$ leaves because every possible permutation must appear in a leaf.

Since the tree is binary, it must have depth at least $\log n! = \Theta(n \log n)$. Thus, we conclude that $D(\texttt{Sort}_n) = \Omega(n \log n)$.

□

We have used a deterministic decision tree. We can also use randomized decision trees, where nodes may represent comparisons or probabilitic decision (e.g. go to left subtree with probability 1/3).

If a probabilistic tree is correct, then it induce a probability distribution over deterministic trees. The mapping is as follows: we build a tree by just flipping the coins before hand and remove paths in the randomized tree as coins decide.

Let p be distribution over deterministic trees. We define $\texttt{cost}(p, x) = \mathbb{E}_{t \in p}[cost(t, x)]$.

Theorem 2. *For all distributions p* $\texttt{cost}(p) := \texttt{max}_x\{\texttt{cost}(p, x)\} = \Omega(n \log n)$

Proof. Step 1: Assume that $\texttt{cost}(p) = C$. Then $\forall x\ \mathbb{E}_{t \in p}[\texttt{cost}(t, x)] \leq C$. Using markov's inequality we obtain that $\Pr[\texttt{cost}(t, x) \geq 2] \leq \frac{\mathbb{E}_t[\texttt{cost}(t,x)]}{2C} \leq 1/2$.

Claim: There is t such that for at least half of inputs x, $\texttt{cost}(t, x) < 2C$.

Lets define the indicatior random variable $Y_x = 1$ if and only if $\texttt{cost}(t, x) < 2C$. Then,

$$\begin{aligned}
\mathbb{E}_t[\sum_x Y_x] &= \sum_x \mathbb{E}_t[Y_x] \\
&= \sum_x \Pr[Y_x = 1] \\
&= \sum_x \Pr[\texttt{cost}(t,x) < 2C] \\
&\geq \frac{n!}{2}
\end{aligned}$$

This implies (by counting argument) that there is a tree that solve half the instances with $< 2C$ comparisons. Therefore we can conclude that $2C \geq \log n!/2$, and thus, $C = \Omega(n \log n)$. □

2 Decision Trees

Let $f : \{0,1\}^n \to \{0,1\}$ be a function. Given access to black box bit-access to input x, how many access to the black box are needed to compute a function f on input x?

Example 1: $\texttt{OR}_n(x_1, ..., x_n) = \bigvee x_i$. $D(\texttt{OR}_n) = n$.

Example 2: $\texttt{GC}_n$: Graph Connectivity of an m-vertex graph G ($n = \binom{m}{2}$). $D(\texttt{GC}_{n=\binom{m}{2}}) = n$.

Conjecture 1. *Any non-constant monotone function f has $D(f) = n$*

However, It is known that $D(f) = \Omega(n)$.

Definition 1. *A funtion f is* evasive *if $D(f) = n$.*

Example 3: $k - \texttt{Block-CNF}(x_1, x_2, x_3, ..., x_{k^2}) = (x_1 \vee .... \vee x_k) \wedge ... \wedge (x_{k(k-1)+1} \vee ... \vee x_{k^2})$. $D(f) = n = k^2$

Example 4: $f(x_1, ..., x_k, y_1, ..., y_{2^k}) = y_{x_1 x_2 ... x_k}$. $D(f) = k + 1$

Definition 2 (Certificates)**.** *Given $x \in \{0,1\}^n$ with $f(x) = b \in \{0,1\}$ a b-*certificate *for x is $S \subseteq [n]$ such that for all $x' \in \{0,1\}^n$ with $x'|_S = x|_S$ $f(x') = f(x)$.*

- *$c_x(f)$: smallest k such that x has a $f(x)$-certificate S of size $|S| \leq k$*
- *$C_b(f) = \texttt{max}_{x:f(x)=b}\{c_x(f)\}$*
- *$C(f) = \texttt{max}(C_0(f), C_1(f))$*

Examples: $C_1(OR_n) = 1$, $C_0(OR_n) = n$, $C_0(k\texttt{-Block-CNF}) = k$ (need to show one entire zero block), $C_1(k\texttt{-Block-CNF}) = k$ (need a 1 representative for each block).

Theorem 3. *$D(f) \geq C(f)$*

Proof sketch: Every path in the best tree for f is a certificate for some input.

We will see later that $D(f) \leq C(f)^2$.

Definition 3 (Sensitivity)**.** *Given f, x we say that $i \in [n]$ is sensitive for x if $f(x) \neq f(x^{(i)})$ (flip i-th bit of x)*

- *$S_x(f) = \#\ i$ that are sensitive for x.*

- $S(f) = \mathtt{max}_x\{S_x(f)\}$

Theorem 4. $D(f) \geq S(f)$

Proof sketch: Let t be the best the tree for f. For every x let $T_x \subseteq [n]$ the path on t taken on input x. We first note that for every x $|T_x| \leq D(f)$ by definition of $D(f)$. Also we note that the set of sensitive values for x is a subset of T, since values not in the path cannot change the value of the function. Then $S_x \leq |T_x|$, and thus $S_x \leq D(f)$ for every x.

Definition 4 (Block Sensitivity)**.** *Given f, x we say that $B \subset [n]$ is sensitive for x if $f(x) \neq f(x^{(B)})$ (flip all bits indicated by B)*

- *$bs_x(f)$: maximum k such that there are disjoint $B_1, ..., B_k$ where B_i is sensitive for x*
- *$bs(f) = \mathtt{max}_x\{bs_x(f)\}$*

Theorem 5. $D(f) \geq C(f) \geq bs(f) \geq S(f)$

Proof Sketch: We will prove that $C(f) \geq bs(f)$. Fix an input string x. Each sensitive block for x must contain a member of a certificate for x. Otherwise, flipping the bits of a sensitive block B not containing a certificate member will change the value of the function. However, both strings x and $x^{(B)}$ have same certificate, reaching the contradiction.