Week 2 Notes

I. Frewald's Algorithm

Inputs: Matrices $A, B, C \in \mathbb{Z}^{n \times n}$

Problem: Verify whether $AB = C$.

Straightforward approach is to compute product $P = A \cdot B$ ~~and~~ and check if $P = C$ entry-wise.

Can we do better?

Idea: Choose random $r \in \{0,1\}^n$ and check if $A(Br) = Cr$.

Note: Choosing $r = (r_1, \ldots, r_n)$ uniformly from $\{0,1\}^n$ is equivalent to choosing each r_i independently and uniformly from $\{0,1\}$.

Lemma: $\Pr_r [ABr \neq Cr] \geq \frac{1}{2}$.

Pf: Let $D = AB - C$ nonzero. W.l.o.g, suppose $D_{11} \neq 0$.

$$\Pr_r [(Dr)_1 = 0] = \Pr\left[\sum_{j=1}^{n} D_{1j} r_j = 0\right]$$

$$= \Pr\left[r_1 = -\frac{\sum_{j=2}^{n} D_{1j} r_j}{D_{11}}\right]$$

Now:

$$\Pr_{r_1,\dots,r_n}\left[r_1 = -\frac{1}{D_{11}} \sum_{j=2}^{n} D_{1j} r_j\right]$$

$$= \sum_{\substack{x_2,\dots,x_n\\ \in\{0,1\}}} \Pr_{r_1}\left[r_1 = -\frac{1}{D_{11}} \sum_{j=2}^{n} D_{1j} r_j \,\middle|\, \begin{matrix} r_2 = x_2, \\ \vdots \\ r_n = x_n \end{matrix}\right] \cdot \Pr_{r_2,\dots,r_n}\left[\begin{matrix} r_2 = x_2, \\ \vdots \\ r_n = x_n \end{matrix}\right]$$

$$= \sum_{x_2,\dots,x_n\in\{0,1\}} \Pr_{r_1}\left[r_1 = -\frac{1}{D_{11}} \sum_{j=2}^{n} D_{1j} x_j\right] \cdot \frac{1}{2^{n-1}}$$

$$\leq \sum_{x_2,\dots,x_n\in\{0,1\}} \frac{1}{2} \cdot \frac{1}{2^{n-1}} = \frac{1}{2} \cdot 2^{n-1} \cdot \frac{1}{2^{n-1}} = \frac{1}{2}$$

where the last inequality is because $\Pr_{r_1}[r_1 = z] \leq \frac{1}{2}$ for any z (the probability is exactly $\frac{1}{2}$ if $z \in \{0,1\}$ and 0 otherwise). $\boxtimes$

<u>Final algorithm</u>: For $i = 1,\dots,t$, choose $r^{(i)} \in \{0,1\}^n$ independently and reject if $A(Br^{(i)}) = Cr^{(i)}$ for any i. Otherwise, accept.

$$\Pr_{r^{(1)},\dots,r^{(t)}}\left[AB \neq C \text{ but algo accepts}\right] \leq \left(\frac{1}{2}\right)^t.$$

If we set $t = \lceil \log 1/\delta \rceil$, error probability $\leq \delta$.

II. AND/OR tree

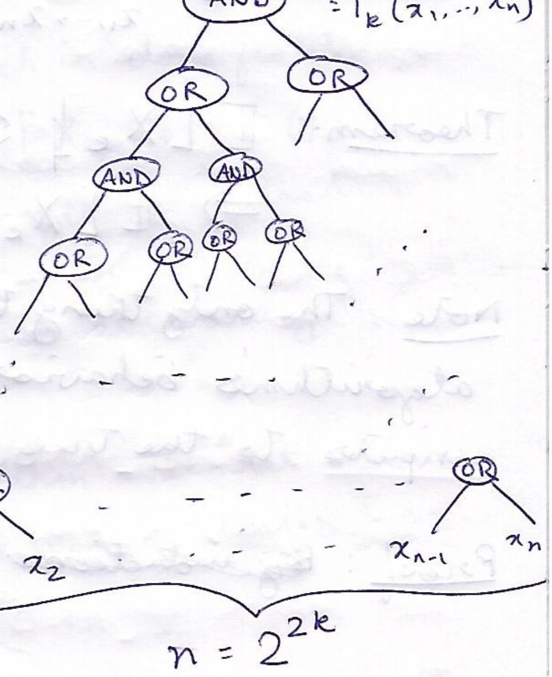
$T_k : \{0,1\}^{2^k} \to \{0,1\}$

$n = 2^{2k}$

<u>Fact</u>: Any deterministic algorithm needs $\Omega(n)$ time to compute T_k.

(See optional exercises!)

$T_k(x)$
$= T_k(x_1, \dots, x_n)$



<u>Randomized algorithm</u>:

- Start at the root
- If AND node,
  - Randomly choose between left and right
  - Evaluate chosen subtree recursively
  - If 0, return 0
  - If 1, evaluate other subtree recursively & return that value.
- If OR node,
  - Randomly choose between left and right
  - Evaluate chosen subtree recursively.
  - If 1, return 1.
  - If 0, evaluate other subtree recursively & return that value.

Define: $X_k = \max_{x_1,\dots,x_n} \left[\#\text{ of steps to evaluate } T_k(x) \right]$

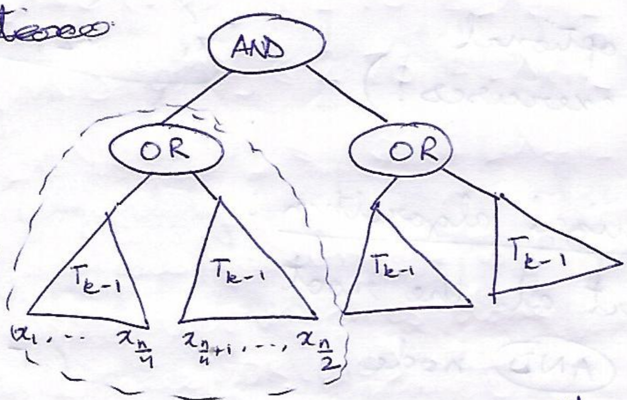
<u>Theorem</u>: $\mathbb{E}[X_k] \leq 3 \cdot \mathbb{E}[X_{k-1}] + O(1)$

$$\Rightarrow \mathbb{E}[X_k] = O(3^k) = O(n^{0.793})$$

<u>Note</u>: The only thing that's random here is the algorithm's behavior! Theorem holds for all inputs to the tree.

<u>Proof</u>:

Let's first look at this OR gate:



Time to compute the OR gate depends on the values of its subtree. Let's define:

$$OR(a,b) = \max_{\substack{x_1,\dots,x_{\frac{n}{4}} : T_{k+1}(x_1,\dots,x_{\frac{n}{4}}) = a \\ x_{\frac{n}{4}+1},\dots,x_{\frac{n}{2}} : T_{k-1}(x_{\frac{n}{4}+1},\dots,x_{\frac{n}{2}}) = b}} \left[\#\text{ of steps to evaluate } OR(T_{k-1}(x_1,\dots,x_{\frac{n}{4}}), T_{k-1}(x_{\frac{n}{4}+1},\dots,x_{\frac{n}{2}})) \right]$$

<u>Lemma</u>: (i) $\mathbb{E}[OR(0,0)] \leq 2 \cdot \mathbb{E}[X_{k-1}] + O(1)$

(ii) $\mathbb{E}[OR(1,0)] \leq 1.5\ \mathbb{E}[X_{k-1}] + O(1)$,

$\mathbb{E}[OR(0,1)] \leq 1.5\ \mathbb{E}[X_{k-1}] + O(1)$

(iii) $\mathbb{E}[OR(1,1)] \leq \mathbb{E}[X_{k-1}] + O(1)$

Pf :

(i) When both subtrees evaluate to 0, both must be evaluated no matter which is done first.

So, $\mathbb{E}[OR(a,b)] \leq 2 \cdot \mathbb{E}[X_{k-1}] + O(1)$.

Here, we are implicitly using the fact that the algorithm's choice at the top OR node doesn't affect its choices in evaluating the subtrees.

(ii) If $a=1$, $b=0$, the algorithm evaluates one subtree if left is chosen but evaluates two subtrees if right is chosen.

$$
\begin{aligned}
&\mathbb{E}[OR(a,b)] \\
&= \frac{1}{2}\,\mathbb{E}[OR(a,b) \mid \text{left first}] + \frac{1}{2}\,\mathbb{E}[OR(a,b) \mid \text{right first}] \\
&\leq \frac{1}{2}\left(\mathbb{E}[X_{k-1}]\right) + \frac{1}{2}\cdot\left(2\,\mathbb{E}[X_{k-1}]\right) + O(1) \\
&= 1.5\ \mathbb{E}[X_{k-1}] + O(1).
\end{aligned}
$$

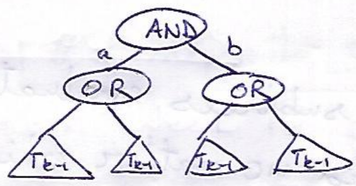
Same for $a=0$, $b=1$.

(iii) If both subtrees evaluate to 1, only one will be evaluated no matter which is done first.

$\mathbb{E}[OR(1,1)] \leq \mathbb{E}[X_{k-1}] + O(1)$.

$\boxtimes$

Now, look at the top AND gate:



Again, cost of evaluating AND depends on its subtrees

Let $$AND(a,b) = \max_{\substack{x_1,\dots,x_{\frac{n}{2}} : OR(T_{k-1}(\cdot\cdot), T_{k-1}(\cdot\cdot)) = a \\ x_{\frac{n}{2}+1},\dots,x_n : OR(T_{k-1}(\cdot\cdot), T_{k-1}(\cdot\cdot)) = b}} \left[\begin{array}{l} \text{\# of steps} \\ \text{to evaluate } T_k(x) \end{array} \right]$$

<u>Lemma</u>: (i) $E[AND(0,0)] \leq 2 \cdot E[X_{k-1}] + O(1)$

(ii) $E[AND(1,0)] \leq 2.75\, E[X_{k-1}] + O(1)$,

$E[AND(0,1)] \leq 2.75\, E[X_{k-1}] + O(1)$

(iii) $E[AND(1,1)] \leq 3\, E[X_{k-1}] + O(1)$.

<u>Pf</u>: (i) When both subtrees ~~are~~ evaluate to 0, only one is evaluated, no matter which is done first. Each subtree takes OR(0,0) steps in the worst case. So:

$$\cancel{E[X_k]}\; E[AND(0,0)] \leq E[OR(0,0)] + O(1)$$
$$\leq 2\, E[X_{k-1}] + O(1)$$

from part (i) of previous lemma.

(ii) Suppose $a=1$, $b=0$. The algorithm evaluates both subtrees if left is done first but only evaluates one if right is done first

$$\mathbb{E}[\mathrm{AND}(1,0)]$$

$$= \frac{1}{2} \cdot \mathbb{E}[\mathrm{AND}(1,0) \mid \text{right first}] + \frac{1}{2} \mathbb{E}[\mathrm{AND}(1,0) \mid \text{left first}]$$

$$\leq \frac{1}{2} \cdot \mathbb{E}[\mathrm{OR}(0,0)] + \frac{1}{2} \Big(\max\{\mathbb{E}[\mathrm{OR}(1,1)], \mathbb{E}[\mathrm{OR}(1,0)], \mathbb{E}[\mathrm{OR}(0,1)]\} + \mathbb{E}[\mathrm{OR}(0,0)] \Big) + O(1)$$

$$\leq \frac{1}{2} \cdot 2\, \mathbb{E}[X_{k-1}] + \frac{1}{2} \left(1.5\, \mathbb{E}[X_{k-1}] + 2\, \mathbb{E}[X_{k-1}] \right) + O(1)$$

$$= 2.75\, \mathbb{E}[X_{k-1}] + O(1)$$

The first inequality is because an OR gate can be 1 for three possible inputs : (1, 1), (1, 0), (0, 1).

Same holds for $\mathbb{E}[\mathrm{AND}(0,1)]$.

(iii) If both subtrees evaluate to 1, both need to be evaluated no matter which is done first. So:

$$\mathbb{E}[\mathrm{AND}(1,1)]$$

$$\leq 2 \cdot \max\{\mathbb{E}[\mathrm{OR}(1,1)], \mathbb{E}[\mathrm{OR}(1,0)], \mathbb{E}[\mathrm{OR}(0,1)]\} + O(1)$$

$$\leq 2 \cdot 1.5\, \mathbb{E}[X_{k-1}] + O(1) = 3 \cdot \mathbb{E}[X_{k-1}] + O(1).$$

$\boxtimes$

Finally, since X_k bounds the largest possible cost of evaluating the root:

$$\mathbb{E}[X_k] \leq \max\{\mathbb{E}[\mathrm{AND}(0,0)], \mathbb{E}[\mathrm{AND}(1,0)], \mathbb{E}[\mathrm{AND}(0,1)], \mathbb{E}[\mathrm{AND}(1,1)]\}$$

$$\leq 3 \cdot \mathbb{E}[X_{k-1}] + O(1).$$

$\boxtimes$

III. Randomized Quicksort

Recall Rand QS from Week 1: pivot is chosen uniformly from subarray at each step.

Suppose input array is $a_1, \ldots, a_n$ (distinct).

Rename the sorted version of this array as $b_1, \ldots, b_n$.

Also, define $b_{ij} = \{b_i, b_{i+1}, \ldots, b_{j-1}, b_j\}$ for $1 \leq i < j \leq n$.

We prove the following:

<u>Theorem</u>: If X is the total number of comparisons made by Rand QS, $\mathbb{E}[X] = O(n \log n)$.

<u>Proof</u>: First, note that any pair of elements is compared at most once. Because elements are only compared to the pivot and subsequently, that pivot is not used in any recursive call.

Define the indicator random variable

$$X_{ij} = \mathbb{1}[b_i \text{ and } b_j \text{ are compared by Rand QS}].$$

Since b_i and b_j are compared at most once,

$$X = \sum_{i=1}^{n} \sum_{j=i+1}^{n} X_{ij}.$$

$$\Rightarrow \mathbb{E}[X] = \sum_{i=1}^{n} \sum_{j=i+1}^{n} \mathbb{E}[X_{ij}]$$

When are b_i and b_j compared?

If some element not in b_{ij} is chosen as pivot, both b_i and b_j fall on the same side of the partition.

Let x be the first element in b_{ij} chosen as pivot.

- If $b_i < x < b_j$, then $X_{ij} = 0$ because b_i and b_j fall on different sides of the partition and are never compared.
- If $x = b_i$ or $x = b_j$, then $X_{ij} = 1$.

So,
$$\mathbb{E}[\cancel{X}_{ij}] = \Pr\left[\begin{array}{c}\text{first element chosen as pivot} \\ \text{in } b_{ij} \text{ is } b_i \text{ or } b_j\end{array}\right] = \frac{2}{j-i+1}.$$

The last line is because each element in b_{ij} has equal chance of being selected as the first pivot.

$$\therefore \quad \mathbb{E}[X] = \sum_{i=1}^{n} \sum_{j=i+1}^{n} \frac{2}{j-i+1}$$
$$= \sum_{i=1}^{n} \sum_{k=2}^{n-i+1} \frac{2}{k}$$
$$\leq \sum_{i=1}^{n} \sum_{k=1}^{n} \frac{2}{k} \leq 2 \sum_{i=1}^{n} 2 \ln n = O(n \log_2 n).$$

where we used that $H_n := \sum_{k=1}^{n} \frac{1}{k} = \ln n + \Theta(1)$.

$\boxtimes$