

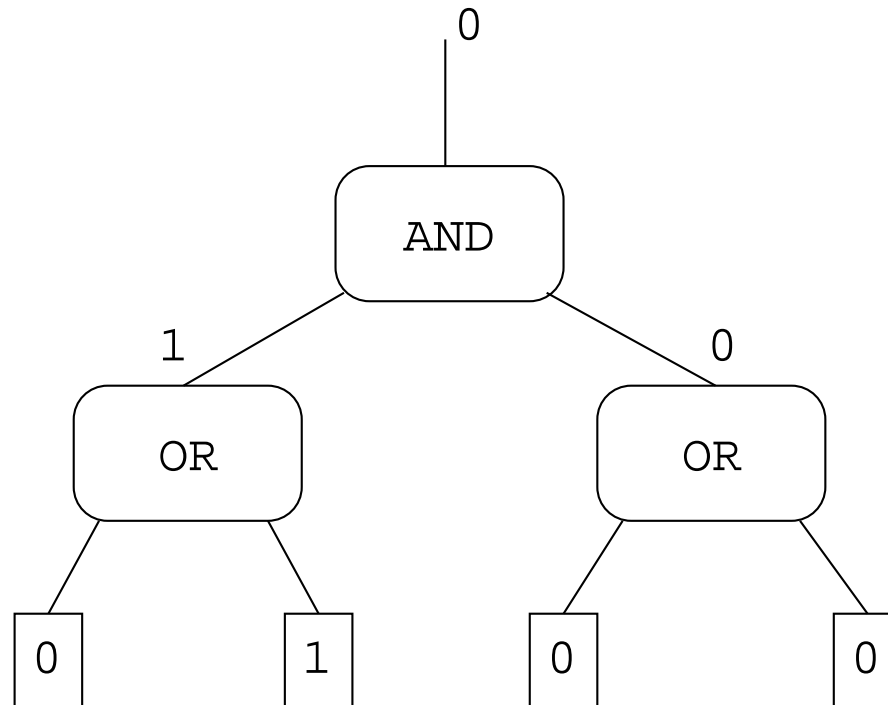
STUDY GROUP
RANDOMIZED ALGORITHMS

June 28, 2003

Topics to be Covered

- Recap of Game Tree Evaluation Problem
- Yao's Technique
- Lower Bound for Game Tree Evaluation

Recap of Game Tree Evaluation Problem



The expected cost of evaluating any instance of $T_{2,k}$ is at most 3^k , in other words, the expected running time of the randomized algorithm is $n^{\log_4 3} = n^{0.793}$. (note: $n = 4^k$ is the number of leaves in $T_{2,k}$)

Yao's Technique

- How good is your randomized algorithm?
 - *Deterministic Algorithm*: Your opponent can always choose an input which force you to evaluate all the 4^k leaves.
 - *Randomized Algorithm*: No matter how your opponent chooses the input, you can always guarantee the *expected* cost is evaluating no more than 3^k leaves.
 - Is there any *better* algorithms?

Yao's Technique is a general technique for proving the lower bound on the running time of randomized algorithms.

Apply the game theory to analyze the complexity:

Player	Role	Goal
row	“input provider”	max runtime of given algorithm
column	“algorithm designer”	min runtime of given input

Example of the payoff matrix:

	Alg_1	Alg_2	\dots	Alg_{100}
$Input_1$	20	20	\dots	0
$Input_2$	5	53	\dots	-24
\dots	\dots	\dots	\dots	\dots
$Input_{1000}$	95	-89	\dots	-30

von Neumann's Minimax Theorem:

$$\max_{\mathbf{p}} \min_{\mathbf{q}} \mathbf{p}^T \mathbf{M} \mathbf{q} = \min_{\mathbf{q}} \max_{\mathbf{p}} \mathbf{p}^T \mathbf{M} \mathbf{q} \quad (1)$$

Loomis' Theorem:

$$\max_{\mathbf{p}} \min_j \mathbf{p}^T \mathbf{M} \mathbf{e}_j = \min_{\mathbf{q}} \max_i \mathbf{e}_i^T \mathbf{M} \mathbf{q} \quad (2)$$

\mathcal{I} : set of input instances, \mathcal{A} : set of deterministic algorithms

$C(I, A)$: running time of algorithm $A \in \mathcal{A}$ on input $I \in \mathcal{I}$

$I_{\mathbf{p}}$: random input chosen according to prob distribution \mathbf{p} over \mathcal{I}

$A_{\mathbf{q}}$: random algorithm chosen according to \mathbf{q} over \mathcal{A}

$$\max_{\mathbf{p}} \min_{\mathbf{q}} \mathbf{E}[C(I_{\mathbf{p}}, A_{\mathbf{q}})] = \min_{\mathbf{q}} \max_{\mathbf{p}} \mathbf{E}[C(I_{\mathbf{p}}, A_{\mathbf{q}})]$$

$$\max_{\mathbf{p}} \min_{A \in \mathcal{A}} \mathbf{E}[C(I_{\mathbf{p}}, A)] = \min_{\mathbf{q}} \max_{I \in \mathcal{I}} \mathbf{E}[C(I, A_{\mathbf{q}})]$$

Yao's Minimax Principle

For all distributions \mathbf{p} over \mathcal{I} and \mathbf{q} over \mathcal{A} ,

$$\min_{A \in \mathcal{A}} \mathbf{E}[C(I_{\mathbf{p}}, A)] \leq \max_{I \in \mathcal{I}} \mathbf{E}[C(I, A_{\mathbf{q}})] \quad (3)$$

LHS: The expected running time of the *best deterministic* algorithm for an arbitrarily chosen input distribution \mathbf{p} .

RHS: The expected running time of an arbitrarily chosen *randomized* algorithm for the *worst-case* input.

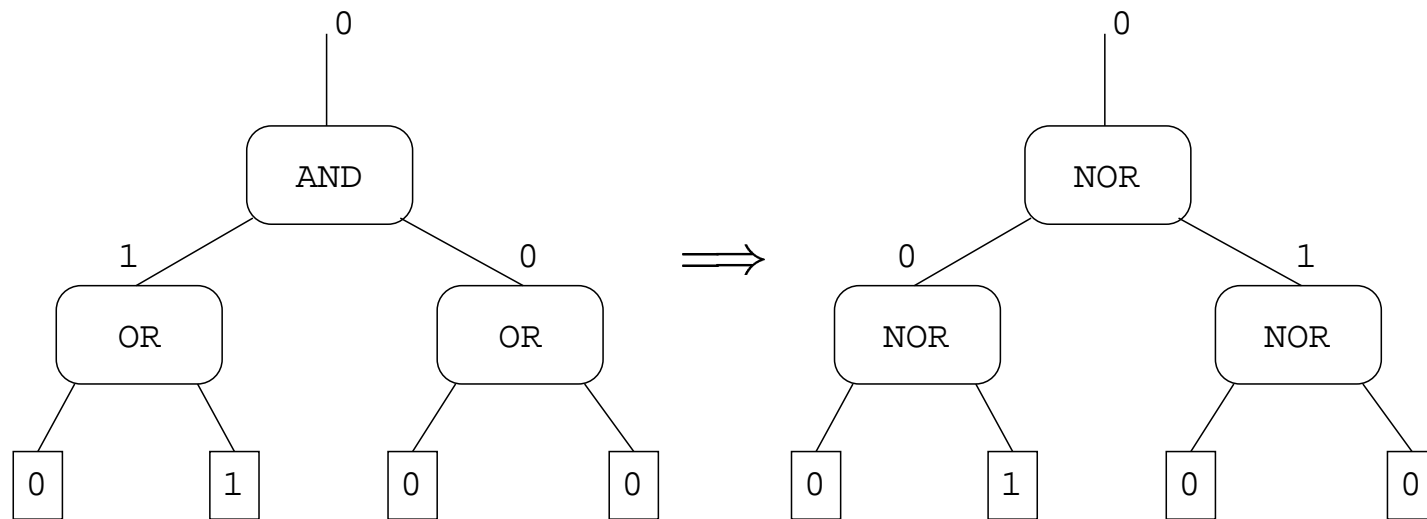
Lower Bound for Game Tree Evaluation

1. Pick a distribution p of inputs over \mathcal{I}
2. Find the *best* deterministic algorithm A for this distribution
3. Calculate the runtime of A

For simplicity, we first transform the AND-OR tree to a NOR tree. By DeMorgan's Law, we then have:

$$(x_1 \vee x_2) \wedge (x_3 \vee x_4) = \neg(\neg(x_1 \vee x_2) \vee \neg(x_3 \vee x_4)) \quad (4)$$

Apply this to the AND-OR tree recursively, we can transform it into a tree with all the internal nodes being NOR gates.



1. *How to pick the distribution p ?*

As to ease the analyze, we choose a uniform distribution for each leaf node to have a probability of $p = (3 - \sqrt{5})/2$ to be the value 1.

2. *How to find the best deterministic algorithm A ?*

DFS (depth-first search) is optimal! The intuition is that when we evaluate the tree in depth-first fashion, we can prune it as early as possible. Tarsi [393] gives a formal proof.

3. *How to calculate its expected ruining time?*

Let $W(h)$ denote the expected number of leaves the algorithm need to inspect so as to determining the value of a node at distance h from the leaves.

$$\begin{aligned} W(h) &= W(h-1) + (1-p)W(h-1) \\ &= (2-p)W(h-1) \end{aligned}$$

Therefore

$$\begin{aligned}
 W(h) &= (2 - p)^h && \text{(since } W(0) = 1\text{)} \\
 &= (2 - p)^{2k} \\
 &= (2 - p)^{2\log_4 n} && \text{(since } n = 4^k\text{)} \\
 &= \left(2 - \frac{3 - \sqrt{5}}{2}\right)^{2\log_4 n} \\
 &= n^{0.694}
 \end{aligned}$$

But $n^{0.694} < n^{0.793}$!

Actually it is because of the probability distribution \mathbf{p} we chooses. Saks and Wigderson [362] gives a better lower bound and shows $n^{0.793}$ is optimal.

Conclusions

- Yao's technique applies game theoretic approach to complexity analysis of randomized algorithms.
- It is the only known general technique to prove the lower bound on the running time of randomized algorithms.
- The choice of input distribution is subtle.