Randomized Algorithms, Lecture 2

Jacob Holm (`jaho@di.ku.dk`)

April 29th 2019

Good morning.

Today's Lecture

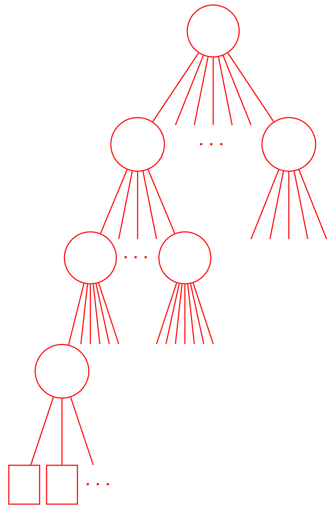
Game-Theoretic Techniques

- Game Tree Evaluation (GTE)
- Yao's Minimax Principle
- GTE Lower Bound
- Randomness and Non-Uniformity

Game Tree Evaluation is an example of a problem where we have a Las Vegas style randomized algorithm that in expectation does better than the best deterministic algorithm.
Yao's Minimax Principle gives us a way to prove lower bounds for randomized algorithms.
We use this to prove a lower bound for randomized GTE.
Finally we talk a bit about complexity theory.

Game Tree, Definition

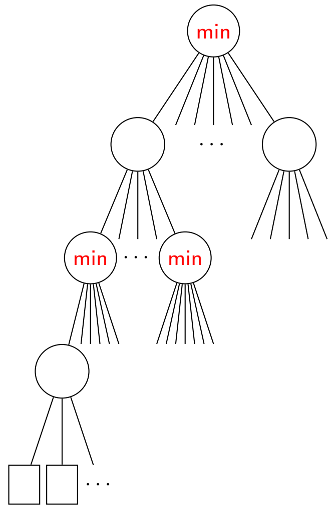
- Rooted tree
- Even depth = min
- Odd depth = max
- Leaves $\in \mathbb{R}$



A game tree corresponds to a finite turn-based two-player game between a *maximizer* and a *minimizer*.
Each node corresponds to a possible *state* of the game, and the children to available choices for the player whose turn it is in that state.
Each leaf describes a final state of the game, and its *value* determines who wins (and how much).
In principle, given such a game tree we can compute an optimal strategy for each player.
In practice, e.g. for Chess or Go, the game trees are often too large, so other ideas are needed there.
Even for simple games we do not know the values of all leaves, but instead have some way to compute them as needed.

Game Tree, Definition

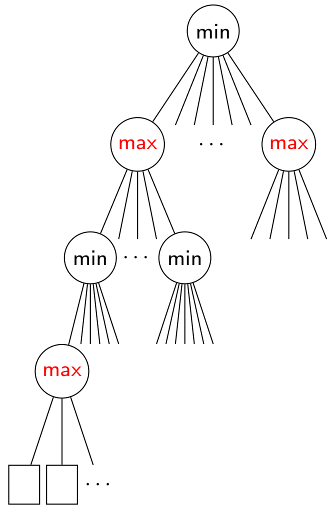
- Rooted tree
- Even depth $=$ min
- Odd depth $=$ max
- Leaves $\in \mathbb{R}$



A game tree corresponds to a finite turn-based two-player game between a *maximizer* and a *minimizer*.
Each node corresponds to a possible *state* of the game, and the children to available choices for the player whose turn it is in that state.
Each leaf describes a final state of the game, and its *value* determines who wins (and how much).
In principle, given such a game tree we can compute an optimal strategy for each player.
In practice, e.g. for Chess or Go, the game trees are often too large, so other ideas are needed there.
Even for simple games we do not know the values of all leaves, but instead have some way to compute them as needed.

Game Tree, Definition

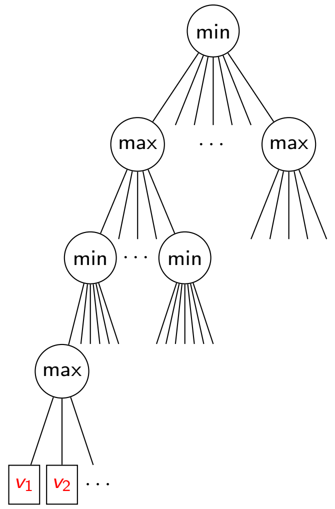
- Rooted tree
- Even depth = min
- Odd depth = max
- Leaves $\in \mathbb{R}$



A game tree corresponds to a finite turn-based two-player game between a *maximizer* and a *minimizer*.
Each node corresponds to a possible *state* of the game, and the children to available choices for the player whose turn it is in that state.
Each leaf describes a final state of the game, and its *value* determines who wins (and how much).
In principle, given such a game tree we can compute an optimal strategy for each player.
In practice, e.g. for Chess or Go, the game trees are often too large, so other ideas are needed there.
Even for simple games we do not know the values of all leaves, but instead have some way to compute them as needed.

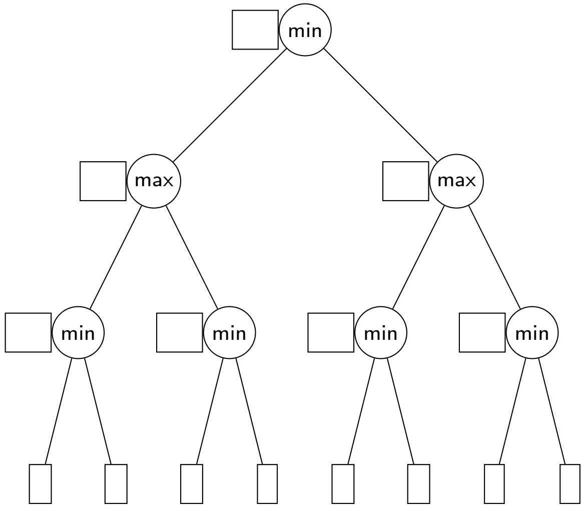
Game Tree, Definition

- Rooted tree
- Even depth $=$ min
- Odd depth $=$ max
- Leaves $\in \mathbb{R}$



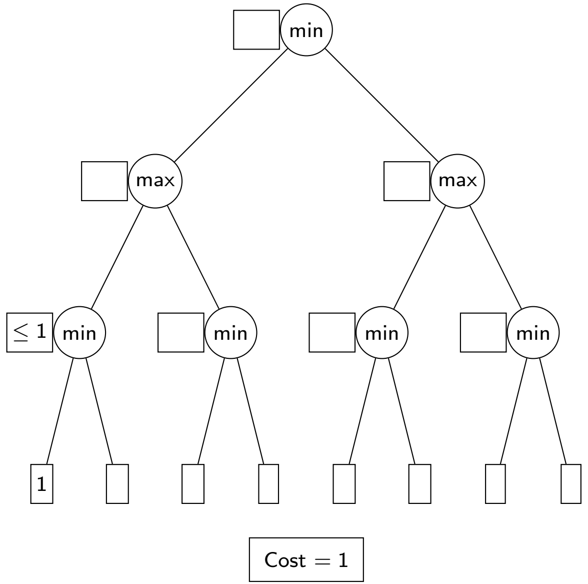
A game tree corresponds to a finite turn-based two-player game between a *maximizer* and a *minimizer*.
Each node corresponds to a possible *state* of the game, and the children to available choices for the player whose turn it is in that state.
Each leaf describes a final state of the game, and its *value* determines who wins (and how much).
In principle, given such a game tree we can compute an optimal strategy for each player.
In practice, e.g. for Chess or Go, the game trees are often too large, so other ideas are needed there.
Even for simple games we do not know the values of all leaves, but instead have some way to compute them as needed.

Game Tree Evaluation, Example

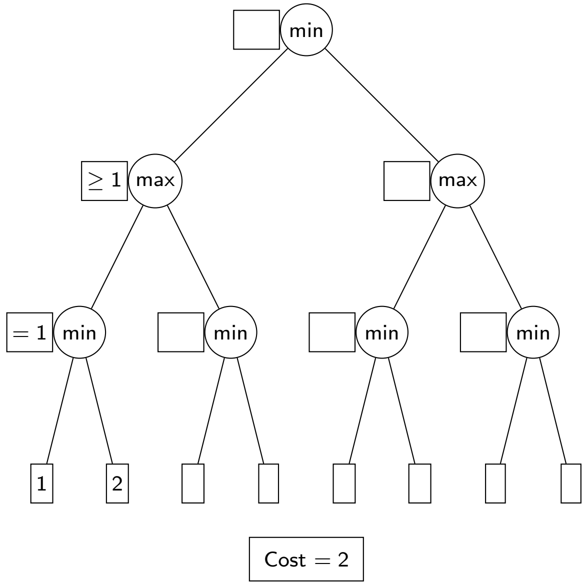


Cost = 0

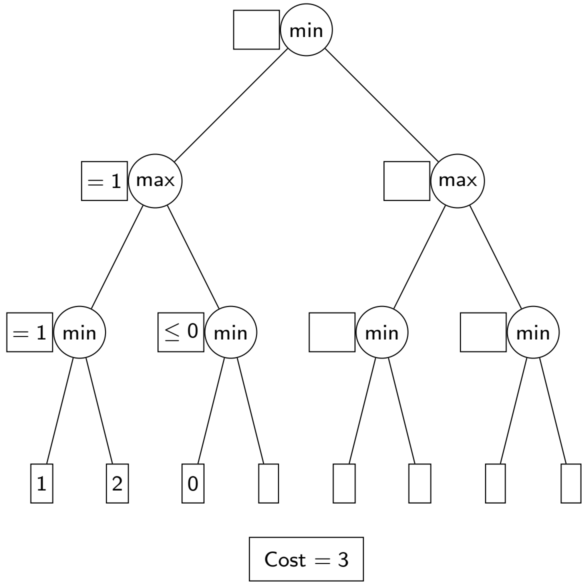
Game Tree Evaluation, Example



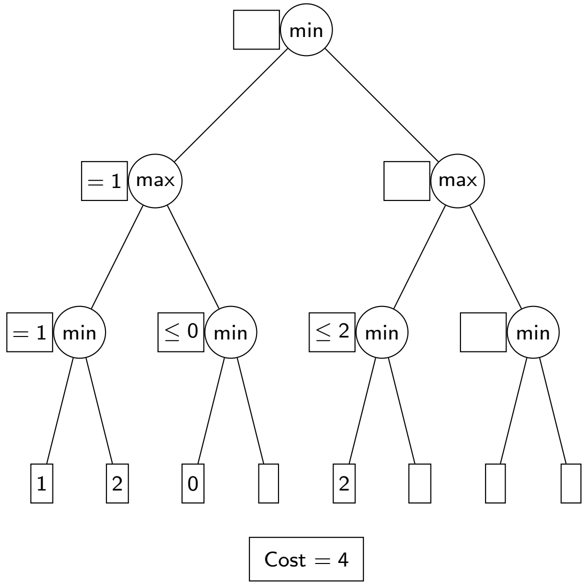
Game Tree Evaluation, Example



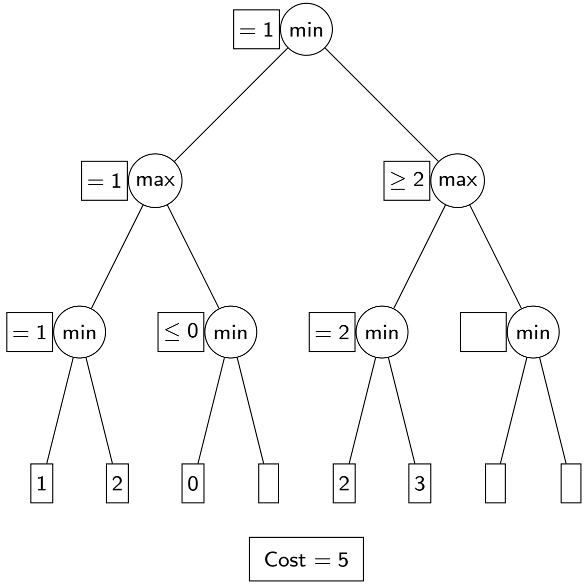
Game Tree Evaluation, Example



Game Tree Evaluation, Example



Game Tree Evaluation, Example



Boolean Game Tree Evaluation

An *instance* of the *boolean game tree evaluation* problem is a game tree $T_{d,k}$, where

- Every internal node has same degree d.
- Every leaf has same depth $2k$.
- Leaf values are restricted to $\{0, 1\}$, so min $\equiv \wedge$ and max $\equiv \vee$.

The goal is to evaluate $T_{d,k}$ while reading the values of as few leaves as possible.

GTE, Deterministic Lower Bound

Problem (Part of Assignment 1)

Show that for any deterministic evaluation algorithm, there is an instance of $T_{d,k}$ that forces the algorithm to read the values on all d^{2k} leaves.

GTE, Deterministic Lower Bound

Problem (Part of Assignment 1)

Show that for any deterministic evaluation algorithm, there is an instance of $T_{d,k}$ that forces the algorithm to read the values on all d^{2k} leaves.

In other words, the worst-case cost of any deterministic algorithm is d^{2k}.

RandBoolGTE

```
1: function $\textsc{RandBoolGTE}(v)$
2:     if $v$ is a leaf then
3:         Return value of $v$.
4:     else
5:         $stop \leftarrow \begin{cases} 0 & \text{if } v \text{ is a "}\wedge\text{" node} \\ 1 & \text{if } v \text{ is a "}\vee\text{" node} \end{cases}$
6:         for each child $c$ in unif. rand. order do
7:             if $\textsc{RandBoolGTE}(c) = stop$ then
8:                 return $stop$
9:         return $1 - stop$
```

RandBoolGTE, Analysis

Theorem

The expected number of leaves read by RandBoolGTE *for any instance of* $T_{2,k}$ *is at most* 3^k.

(Deterministic lower bound is $d^{2k} = 2^{2k} = 4^k$)

RandBoolGTE, Analysis

Theorem

The expected number of leaves read by RandBoolGTE *for any instance of* $T_{2,k}$ *is at most* 3^k*.*

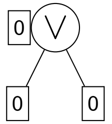
(Deterministic lower bound is $d^{2k} = 2^{2k} = 4^k$)

RandBoolGTE, Proof

Consider a $\vee$ node v that is evaluated by the algorithm. Let Y_v be a random variable that counts the number of v's children that are evaluated.

RandBoolGTE, Proof

Consider a $\vee$ node v that is evaluated by the algorithm. Let Y_v be a random variable that counts the number of v's children that are evaluated.

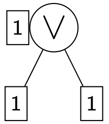


If the value at both children of v is 0, then $Y_v = 2$.

We write $\mathbb{E}[Y_v \mid 00] = 2$.

RandBoolGTE, Proof

Consider a $\vee$ node v that is evaluated by the algorithm. Let Y_v be a random variable that counts the number of v's children that are evaluated.

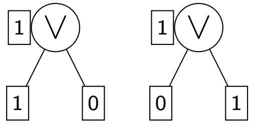


If the value at both children of v is 1, then $Y_v = 1$.

We write $\mathbb{E}[Y_v \mid 11] = 1$.

RandBoolGTE, Proof

Consider a $\vee$ node v that is evaluated by the algorithm. Let Y_v be a random variable that counts the number of v's children that are evaluated.



If the value of exactly one child of v is 1, then $\Pr[Y_v = 1] = \Pr[Y_v = 2] = \frac{1}{2}$.

We write $\mathbb{E}[Y_v \mid 01] = \mathbb{E}[Y_v \mid 10] = 1 \cdot \frac{1}{2} + 2 \cdot \frac{1}{2} = \frac{3}{2}$.

RandBoolGTE, Proof

Consider a $\vee$ node v that is evaluated by the algorithm. Let Y_v be a random variable that counts the number of v's children that are evaluated.

Let $\mathbb{E}[Y_v \mid x]$ denote the expected value of Y_v when v evaluates to x, then we have shown:

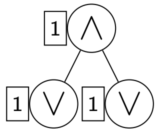
$$\begin{aligned}
\mathbb{E}[Y_v \mid 0] &= \mathbb{E}[Y_v \mid 00] = 2 \\
\mathbb{E}[Y_v \mid 1] &\le \max\left\{\mathbb{E}[Y_v \mid 11], \mathbb{E}[Y_v \mid 01], \mathbb{E}[Y_v \mid 10]\right\} \\
&= \max\left\{1, \frac{3}{2}, \frac{3}{2}\right\} = \frac{3}{2}
\end{aligned}$$

RandBoolGTE, Proof

Consider a $\wedge$ node v that is evaluated by the algorithm. Let X_v be a random variable that counts the number of v's grandchildren that are evaluated.

RandBoolGTE, Proof

Consider a $\wedge$ node v that is evaluated by the algorithm. Let X_v be a random variable that counts the number of v's grandchildren that are evaluated.

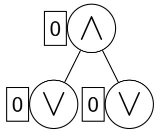


If the value at both children of v is 1, then $X_v = Y_1 + Y_2$.

$$\begin{aligned}\mathbb{E}[X_v \mid 11] &= \mathbb{E}[Y_1 \mid 1] + \mathbb{E}[Y_2 \mid 1] \\ &\leq \frac{3}{2} + \frac{3}{2} = 3\end{aligned}$$

RandBoolGTE, Proof

Consider a $\wedge$ node v that is evaluated by the algorithm. Let X_v be a random variable that counts the number of v's grandchildren that are evaluated.

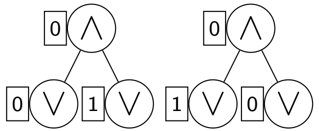


If the value at both children of v is 0, then $X_v = Y_1$.

$$\mathbb{E}[X_v \mid 00] = \mathbb{E}[Y_1 \mid 0] = 2$$

RandBoolGTE, Proof

Consider a $\wedge$ node v that is evaluated by the algorithm. Let X_v be a random variable that counts the number of v's grandchildren that are evaluated.



If the value of exactly one child v is 1, then $\Pr[X_v = \mathbb{E}[Y_1|0]] = \frac{1}{2}$ and $\Pr\big[X_v = (\mathbb{E}[Y_1|1] + \mathbb{E}[Y_2|0])\big] = \frac{1}{2}$.

$$\begin{aligned}
\mathbb{E}[X_v \mid 01] = \mathbb{E}[X_v \mid 10] &= \frac{1}{2} \cdot \mathbb{E}[Y_1 \mid 0] + \frac{1}{2} \cdot \big(\mathbb{E}[Y_1 \mid 1] + \mathbb{E}[Y_2 \mid 0]\big) \\
&\leq \frac{1}{2} \cdot 2 + \frac{1}{2} \cdot \left(\frac{3}{2} + 2\right) = \frac{11}{4}
\end{aligned}$$

RandBoolGTE, Proof

Consider a $\wedge$ node v that is evaluated by the algorithm. Let X_v be a random variable that counts the number of v's grandchildren that are evaluated.

We have now shown:

$$\begin{aligned}\mathbb{E}[X_v] &\leq \max\{\mathbb{E}[X_v|00], \mathbb{E}[X_v|01], \mathbb{E}[X_v|10], \mathbb{E}[X_v|11]\} \\ &= \max\left\{2, \frac{11}{4}, \frac{11}{4}, 3\right\} \leq 3\end{aligned}$$

RandBoolGTE, Proof

Let Z_v be the number of leaf reads done by $\textsc{RandBoolGTE}(v)$, and let $\mathcal{E}_v$ be the event that v is evaluated. We prove that $\mathbb{E}[Z_v|\mathcal{E}_v] \leq 3^k$ when $v = \text{root}(T_{2,k})$ by induction on k.

RandBoolGTE, Proof

Let Z_v be the number of leaf reads done by $\textsc{RandBoolGTE}(v)$, and let $\mathcal{E}_v$ be the event that v is evaluated. We prove that $\mathbb{E}[Z_v|\mathcal{E}_v] \leq 3^k$ when $v = \text{root}(T_{2,k})$ by induction on k.

$k = 0$: trivial, since $\mathbb{E}[Z_v|\mathcal{E}_v] = 1 = 3^0$.

RandBoolGTE, Proof

Let Z_v be the number of leaf reads done by $\textsc{RandBoolGTE}(v)$, and let $\mathcal{E}_v$ be the event that v is evaluated. We prove that $\mathbb{E}[Z_v|\mathcal{E}_v] \leq 3^k$ when $v = \text{root}(T_{2,k})$ by induction on k.

$k > 0$: Let C be the grandchildren of v, and suppose $\mathbb{E}[Z_c|\mathcal{E}_c] \leq 3^{k-1}$ for $c \in C$.

$$\begin{aligned}\mathbb{E}[Z_v|\mathcal{E}_v] &= \sum_{c\in C} \mathbb{E}[Z_c|\mathcal{E}_c] \cdot \Pr[\mathcal{E}_c|\mathcal{E}_v] \\ &\leq \sum_{c\in C} 3^{k-1} \cdot \Pr[\mathcal{E}_c|\mathcal{E}_v] \\ &= 3^{k-1} \cdot \mathbb{E}[X_v|\mathcal{E}_v] \leq 3^{k-1} \cdot 3 = 3^k\end{aligned}$$

Expand on blackboard?

$$\begin{aligned}\mathbb{E}[Z_v|\mathcal{E}_v] &= \sum_{d \text{ leaf below } v} \Pr[\mathcal{E}_d|\mathcal{E}_v] \\ &= \sum_{\substack{d \text{ leaf} \\ \text{below } v}} \left(\prod_{\substack{a \text{ ancestor to } d \\ \text{below } v}} \Pr[\mathcal{E}_a|\mathcal{E}_{p(a)}] \right) \\ &= \sum_{c\in C} \sum_{\substack{d \text{ leaf} \\ \text{below } c}} \left(\Pr[\mathcal{E}_c|\mathcal{E}_v] \cdot \left(\prod_{\substack{a \text{ ancestor to } d \\ \text{below } c}} \Pr[\mathcal{E}_a|\mathcal{E}_{p(a)}] \right)\right) \\ &= \sum_{c\in C} \Pr[\mathcal{E}_c|\mathcal{E}_v] \cdot \left(\sum_{\substack{d \text{ leaf} \\ \text{below } c}} \left(\prod_{\substack{a \text{ ancestor to } d \\ \text{below } c}} \Pr[\mathcal{E}_a|\mathcal{E}_{p(a)}] \right)\right) \\ &= \sum_{c\in C} \Pr[\mathcal{E}_c|\mathcal{E}_v] \cdot \mathbb{E}[Z_c|\mathcal{E}_c] \\ &\leq \sum \Pr[\mathcal{E}_c|\mathcal{E}_v] \cdot 3^{k-1} = \mathbb{E}[X_v|\mathcal{E}_v] \cdot 3^{k-1} \leq 3 \cdot 3^{k-1} = 3^k\end{aligned}$$

RandBoolGTE, Summary

More on Game Tree Evaluation later today

The expected cost of the naive randomized algorithm $\textsc{RandBoolGTE}$ on *any* instance of $T_{2,k}$ is at most $3^k = n^{\log_4 3} = n^{0.793\ldots}$ where $n = 4^k$ is the number of leaves.
This is better than the worst case cost for *any* deterministic algorithm, which is n.

Game Theory, Payoff Matrix Example

Consider a game of "Rock, Paper, Scissors". Suppose Roberta and Charles play, and the loser pays $1 to the winner. This is a *two-person zero-sum game* with *payoff matrix*:

$$\underbrace{\left.\begin{matrix} 0 & -1 & 1 \\ 1 & 0 & -1 \\ -1 & 1 & 0 \end{matrix}\right\}\text{Roberta}}_{\text{Charles}}$$

Entries are what Charles should pay to Roberta.
First row/column is Rock.
Second row/column is Paper.
Third row/column is Scissors.
It is zero-sum because the total amount of money among the players is unchanging.

Game Theory, Payoff Matrix Example

Consider a game of “Rock, Paper, Scissors”. Suppose Roberta and Charles play, and the loser pays \$1 to the winner. This is a *two-person zero-sum game* with *payoff matrix*:

$$\underbrace{\left.\begin{matrix} 0 & -1 & 1 \\ 1 & 0 & -1 \\ -1 & 1 & 0 \end{matrix}\right\}\text{Roberta}}_{\text{Charles}}$$

Entries are what Charles should pay to Roberta.
First row/column is Rock.
Second row/column is Paper.
Third row/column is Scissors.
It is zero-sum because the total amount of money among the players is unchanging.

Game Theory, General Payoff Matrix

Any two-player zero-sum game has a payoff matrix $\mathbf{M} \in \mathbb{R}^{n \times m}$, where n and m are the number of *strategies* for each player. M_{ij} is how much column player using strategy $j \in \{1, \ldots, m\}$ must pay to row player playing strategy $i \in \{1, \ldots, n\}$.

Game Theory, Value

Row player R tries to maximize, Column player C tries to minimize.

If R plays strategy i, she gets at least $\min_j M_{ij}$. If C plays strategy j, he pays at most $\max_i M_{ij}$. Let

$$V_R = \max_i \min_j M_{ij} \qquad \text{(Min. payoff for } R \text{ with opt strategy.)}$$

$$V_C = \min_j \max_i M_{ij} \qquad \text{(Max. cost for } C \text{ with opt strategy.)}$$

Then $V_R \le V_C$ (Excercise 2.1).

Let k be an optimal strategy for R and ℓ be an optimal strategy for C.
Then $V_R = \min_j M_{kj} \le M_{k\ell} \le \max_i M_{i\ell} = V_C$.

Game Theory, Value

Row player R tries to maximize, Column player C tries to minimize.

If R plays strategy i, she gets at least $\min_j M_{ij}$. If C plays strategy j, he pays at most $\max_i M_{ij}$. Let

$$V_R = \max_i \min_j M_{ij} \qquad \text{(Min. payoff for } R \text{ with opt strategy.)}$$

$$V_C = \min_j \max_i M_{ij} \qquad \text{(Max. cost for } C \text{ with opt strategy.)}$$

Then $V_R \leq V_C$ (Excercise 2.1).

Let k be an optimal strategy for R and ℓ be an optimal strategy for C.
Then $V_R = \min_j M_{kj} \leq M_{k\ell} \leq \max_i M_{i\ell} = V_C$.

Game Theory, Value

Row player R tries to maximize, Column player C tries to minimize.

If R plays strategy i, she gets at least $\min_j M_{ij}$. If C plays strategy j, he pays at most $\max_i M_{ij}$. Let

$$V_R = \max_i \min_j M_{ij} \quad \text{(Min. payoff for } R \text{ with opt strategy.)}$$
$$V_C = \min_j \max_i M_{ij} \quad \text{(Max. cost for } C \text{ with opt strategy.)}$$

Then $V_R \leq V_C$ (Excercise 2.1).

Let k be an optimal strategy for R and ℓ be an optimal strategy for C.
Then $V_R = \min_j M_{kj} \leq M_{k\ell} \leq \max_i M_{i\ell} = V_C$.

Game Theory, Value Example

In general, $V_R < V_C$. E.g. in the Rock, Paper, Scissors example:

$$\begin{pmatrix} 0 & -1 & 1 \\ 1 & 0 & -1 \\ -1 & 1 & 0 \end{pmatrix}$$

What is V_R?
What is V_C?

Game Theory, Value Example

In general, $V_R < V_C$. E.g. in the Rock, Paper, Scissors example:

$$\begin{pmatrix} 0 & -1 & 1 \\ 1 & 0 & -1 \\ -1 & 1 & 0 \end{pmatrix}$$

What is V_R? $V_R = -1$ for $i \in \{1, \ldots, 3\}$
What is V_C?

Game Theory, Value Example

In general, $V_R < V_C$. E.g. in the Rock, Paper, Scissors example:

$$\begin{pmatrix} 0 & -1 & 1 \\ 1 & 0 & -1 \\ -1 & 1 & 0 \end{pmatrix}$$

What is V_R? $V_R = -1$ for $i \in \{1, \ldots, 3\}$
What is V_C? $V_C = +1$ for $j \in \{1, \ldots, 3\}$

Game Theory, Value Example

Another example:

$$\begin{pmatrix} 0 & 1 & 2 \\ -1 & 0 & 1 \\ -2 & -1 & 0 \end{pmatrix}$$

What is V_R?
What is V_C?

Game Theory, Value Example

Another example:

$$\begin{pmatrix} 0 & 1 & 2 \\ -1 & 0 & 1 \\ -2 & -1 & 0 \end{pmatrix}$$

What is V_R? $V_R = 0$ for $i = 1$

What is V_C?

Game Theory, Value Example

Another example:

$$\begin{pmatrix} 0 & 1 & 2 \\ -1 & 0 & 1 \\ -2 & -1 & 0 \end{pmatrix}$$

What is V_R? $V_R = 0$ for $i = 1$

What is V_C? $V_C = 0$ for $j = 1$

Game Theory, Value Example

Another example:

$$\begin{pmatrix} 0 & 1 & 2 \\ -1 & 0 & 1 \\ -2 & -1 & 0 \end{pmatrix}$$

What is V_R? $V_R = 0$ for $i = 1$
What is V_C? $V_C = 0$ for $j = 1$
We call $V = V_R = V_C$ the *value* of the game, and $(i, j) = (1, 1)$ a *solution* or *saddle point*. Solution may not be unique.

Game Theory, Mixed Strategy

If there is a solution, then each player has a clear optimal strategy that is independent of what the opponent chooses.

Otherwise no *deterministic* (aka. *pure*) strategy is optimal against all opponent strategies.

Instead a *randomized* (aka. *mixed*) strategy is needed.

Game Theory, Mixed Strategy

If there is a solution, then each player has a clear optimal strategy that is independent of what the opponent chooses.

Otherwise no *deterministic* (aka. *pure*) strategy is optimal against all opponent strategies.

Instead a *randomized* (aka. *mixed*) strategy is needed.

Game Theory, Mixed Strategy

If there is a solution, then each player has a clear optimal strategy that is independent of what the opponent chooses.
Otherwise no *deterministic* (aka. *pure*) strategy is optimal against all opponent strategies.
Instead a *randomized* (aka. *mixed*) strategy is needed.

Game Theory, Mixed Strategy

Let $\mathbf{p} = (p_1, \ldots, p_n)$ with $p_i = \Pr[R \text{ plays } i]$, and
let $\mathbf{q} = (q_1, \ldots, q_m)$ with $q_j = \Pr[C \text{ plays } j]$.
Then the payoff is a random variable, and

$$\mathbb{E}[\text{payoff}] = \mathbf{p}^T \mathbf{M} \mathbf{q} = \sum_{i=1}^{n} \sum_{j=1}^{m} p_i M_{ij} q_j$$

Game Theory, von Neumann

Theorem (von Neumann's Minimax Theorem)

For any two-person zero-sum game specified by a matrix $\mathbf{M}$,

$$\max_p \min_q \mathbf{p}^T\mathbf{M}\mathbf{q} = \min_q \max_p \mathbf{p}^T\mathbf{M}\mathbf{q}$$

Thus, there is an optimal mixed strategy that works against any opponent.

Note that if R knows that C plays according to mixed strategy q, then she has a *pure* strategy that is optimal.

Interestingly, we can also show the other direction.

Game Theory, Loomis

Let $\mathbf{e}_k$ denote the kth unit vector (in any dimension).

Theorem (Loomis' Theorem)

For any two-person zero-sum game specified by a matrix $\mathbf{M}$,

$$\max_p \min_j \mathbf{p}^T \mathbf{M} \mathbf{e}_j = \min_q \max_i \mathbf{e}_i^T \mathbf{M} \mathbf{q}$$

Thus, picking a mixed strategy that is optimal against any *pure* strategy, in fact gives you a strategy that is optimal against *any* strategy (pure or mixed).

Yao's Technique

Let $\mathcal{A}$ be a finite set of deterministic algorithms, and $\mathcal{I}$ a finite set of input instances. For $A \in \mathcal{A}, I \in \mathcal{I}$ let $C(I, A)$ denote the *cost* of running A on I. For distributions $\mathbf{p}$ over $\mathcal{A}$ and $\mathbf{q}$ over $\mathcal{I}$, let $A_\mathbf{p} \in \mathcal{A}$ be chosen according to $\mathbf{p}$ and $I_\mathbf{q} \in \mathcal{I}$ be chosen according to $\mathbf{q}$. Then

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

and

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

So, where are we going with all this game theory? We can think of randomized algorithm design as a game, with the algorithm designer (you) as the column player C, and the adversary choosing the inputs as the row player R.
We can then rewrite von Neumann's Minimax Theorem and Loomis' Theorem to

Yao's Technique

Let $\mathcal{A}$ be a finite set of deterministic algorithms, and $\mathcal{I}$ a finite set of input instances. For $A \in \mathcal{A}, I \in \mathcal{I}$ let $C(I, A)$ denote the *cost* of running A on I. For distributions $\mathbf{p}$ over $\mathcal{A}$ and $\mathbf{q}$ over $\mathcal{I}$, let $A_\mathbf{p} \in \mathcal{A}$ be chosen according to $\mathbf{p}$ and $I_\mathbf{q} \in \mathcal{I}$ be chosen according to $\mathbf{q}$. Then

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

and

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

So, where are we going with all this game theory?
We can think of randomized algorithm design as a game, with the algorithm designer (you) as the column player C, and the adversary choosing the inputs as the row player R.
We can then rewrite von Neumann's Minimax Theorem and Loomis' Theorem to

Yao's Minimax Principle

Theorem (Yao's Minimax Principle)

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

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

Thus, for any input distribution $\mathbf{p}$, the expected cost of the optimal deterministic algorithm on $I_{\mathbf{p}}$ is a lower bound on the cost of the optimal Las Vegas randomized algorithm $A_{\mathbf{q}}$.

This theorem immediately follows, since the largest possible value on the left is equal to the smallest possible value on the right.
But what does the theorem really say?
To prove a lower bound on randomized complexity, it suffices to pick an input distribution and prove a lower bound for the expected cost of the best deterministic algorithm on that distribution.

Yao's Minimax Principle

Theorem (Yao's Minimax Principle)

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

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

Thus, for any input distribution $\mathbf{p}$, the expected cost of the optimal deterministic algorithm on $I_{\mathbf{p}}$ is a lower bound on the cost of the optimal Las Vegas randomized algorithm $A_{\mathbf{q}}$.

This theorem immediately follows, since the largest possible value on the left is equal to the smallest possible value on the right.
But what does the theorem really say?
To prove a lower bound on randomized complexity, it suffices to pick an input distribution and prove a lower bound for the expected cost of the best deterministic algorithm on that distribution.

Yao's Minimax Principle, Proof

However, this all relies on von Neumann's and Loomis' theorems. Here is a simple direct proof, taken from Wikipedia but translated to use the notation from the book.

$$
\begin{aligned}
&\max_{I \in \mathcal{I}} \underset{q}{\mathbb{E}}[C(I, A_q)] \\
&= 1 \cdot \max_{I \in \mathcal{I}} \underset{q}{\mathbb{E}}[C(I, A_q)] \\
&= \left(\sum_{I' \in \mathcal{I}} \Pr[I_p = I'] \right) \cdot \max_{I \in \mathcal{I}} \underset{q}{\mathbb{E}}[C(I, A_q)] \\
&= \sum_{I' \in \mathcal{I}} \Pr[I_p = I'] \cdot \max_{I \in \mathcal{I}} \underset{q}{\mathbb{E}}[C(I, A_q)] \\
&\geq \sum_{I' \in \mathcal{I}} \Pr[I_p = I'] \cdot \underset{q}{\mathbb{E}}[C(I', A_q)] \\
&= \sum_{I' \in \mathcal{I}} \Pr[I_p = I'] \cdot \sum_{A' \in \mathcal{A}} \Pr[A_q = A'] \cdot C(I', A') \\
&= \sum_{I' \in \mathcal{I}} \sum_{A' \in \mathcal{A}} \Pr[I_p = I'] \cdot \Pr[A_q = A'] \cdot C(I', A')
\end{aligned}
$$

$$
\begin{aligned}
&= \sum_{A' \in \mathcal{A}} \sum_{I' \in \mathcal{I}} \Pr[A_q = A'] \cdot \Pr[I_p = I'] \cdot C(I', A') \\
&= \sum_{A' \in \mathcal{A}} \Pr[A_q = A'] \cdot \sum_{I' \in \mathcal{I}} \Pr[I_p = I'] \cdot C(I', A') \\
&= \sum_{A' \in \mathcal{A}} \Pr[A_q = A'] \cdot \underset{p}{\mathbb{E}}[C(I_p, A')] \\
&\geq \sum_{A' \in \mathcal{A}} \Pr[A_q = A'] \cdot \min_{A \in \mathcal{A}} \underset{p}{\mathbb{E}}[C(I_p, A)] \\
&= \left(\sum_{A' \in \mathcal{A}} \Pr[A_q = A'] \right) \cdot \min_{A \in \mathcal{A}} \underset{p}{\mathbb{E}}[C(I_p, A)] \\
&= 1 \cdot \min_{A \in \mathcal{A}} \underset{p}{\mathbb{E}}[C(I_p, A)] \\
&= \min_{A \in \mathcal{A}} \underset{p}{\mathbb{E}}[C(I_p, A)]
\end{aligned}
$$

Yao's Minimax Principle, Proof

$$
\begin{aligned}
&\max_{I\in\mathcal{I}} \underset{q}{\mathbb{E}}[C(I,A_q)] \\
&= 1 \cdot \max_{I\in\mathcal{I}} \underset{q}{\mathbb{E}}[C(I,A_q)] \\
&= \left(\sum_{I'\in\mathcal{I}} \Pr[I_p = I']\right) \cdot \max_{I\in\mathcal{I}} \underset{q}{\mathbb{E}}[C(I,A_q)] \\
&= \sum_{I'\in\mathcal{I}} \Pr[I_p = I'] \cdot \max_{I\in\mathcal{I}} \underset{q}{\mathbb{E}}[C(I,A_q)] \\
&\geq \sum_{I'\in\mathcal{I}} \Pr[I_p = I'] \cdot \underset{q}{\mathbb{E}}[C(I',A_q)] \\
&= \sum_{I'\in\mathcal{I}} \Pr[I_p = I'] \cdot \sum_{A'\in\mathcal{A}} \Pr[A_q = A'] \cdot C(I',A') \\
&= \sum_{I'\in\mathcal{I}} \sum_{A'\in\mathcal{A}} \Pr[I_p = I'] \cdot \Pr[A_q = A'] \cdot C(I',A')
\end{aligned}
$$

$$
\begin{aligned}
&= \sum_{A'\in\mathcal{A}} \sum_{I'\in\mathcal{I}} \Pr[A_q = A'] \cdot \Pr[I_p = I'] \cdot C(I',A') \\
&= \sum_{A'\in\mathcal{A}} \Pr[A_q = A'] \cdot \sum_{I'\in\mathcal{I}} \Pr[I_p = I'] \cdot C(I',A') \\
&= \sum_{A'\in\mathcal{A}} \Pr[A_q = A'] \cdot \underset{p}{\mathbb{E}}[C(I_p,A')] \\
&\geq \sum_{A'\in\mathcal{A}} \Pr[A_q = A'] \cdot \min_{A\in\mathcal{A}} \underset{p}{\mathbb{E}}[C(I_p,A)] \\
&= \left(\sum_{A'\in\mathcal{A}} \Pr[A_q = A']\right) \cdot \min_{A\in\mathcal{A}} \underset{p}{\mathbb{E}}[C(I_p,A)] \\
&= 1 \cdot \min_{A\in\mathcal{A}} \underset{p}{\mathbb{E}}[C(I_p,A)] \\
&= \min_{A\in\mathcal{A}} \underset{p}{\mathbb{E}}[C(I_p,A)]
\end{aligned}
$$

However, this all relies on von Neumann's and Loomis' theorems. Here is a simple direct proof, taken from Wikipedia but translated to use the notation from the book.
Multiply by 1.

Yao's Minimax Principle, Proof

$$\max_{I \in \mathcal{I}} \mathop{\mathbb{E}}_q[C(I, A_q)]$$
$$= 1 \cdot \max_{I \in \mathcal{I}} \mathop{\mathbb{E}}_q[C(I, A_q)]$$
$$= \left(\sum_{I' \in \mathcal{I}} \Pr[I_p = I'] \right) \cdot \max_{I \in \mathcal{I}} \mathop{\mathbb{E}}_q[C(I, A_q)]$$
$$= \sum_{I' \in \mathcal{I}} \Pr[I_p = I'] \cdot \max_{I \in \mathcal{I}} \mathop{\mathbb{E}}_q[C(I, A_q)]$$
$$\geq \sum_{I' \in \mathcal{I}} \Pr[I_p = I'] \cdot \mathop{\mathbb{E}}_q[C(I', A_q)]$$
$$= \sum_{I' \in \mathcal{I}} \Pr[I_p = I'] \cdot \sum_{A' \in \mathcal{A}} \Pr[A_q = A'] \cdot C(I', A')$$
$$= \sum_{I' \in \mathcal{I}} \sum_{A' \in \mathcal{A}} \Pr[I_p = I'] \cdot \Pr[A_q = A'] \cdot C(I', A')$$
$$= \sum_{A' \in \mathcal{A}} \sum_{I' \in \mathcal{I}} \Pr[A_q = A'] \cdot \Pr[I_p = I'] \cdot C(I', A')$$
$$= \sum_{A' \in \mathcal{A}} \Pr[A_q = A'] \cdot \sum_{I' \in \mathcal{I}} \Pr[I_p = I'] \cdot C(I', A')$$
$$= \sum_{A' \in \mathcal{A}} \Pr[A_q = A'] \cdot \mathop{\mathbb{E}}_p[C(I_p, A')]$$
$$\geq \sum_{A' \in \mathcal{A}} \Pr[A_q = A'] \cdot \min_{A \in \mathcal{A}} \mathop{\mathbb{E}}_p[C(I_p, A)]$$
$$= \left(\sum_{A' \in \mathcal{A}} \Pr[A_q = A'] \right) \cdot \min_{A \in \mathcal{A}} \mathop{\mathbb{E}}_p[C(I_p, A)]$$
$$= 1 \cdot \min_{A \in \mathcal{A}} \mathop{\mathbb{E}}_p[C(I_p, A)]$$
$$= \min_{A \in \mathcal{A}} \mathop{\mathbb{E}}_p[C(I_p, A)]$$

However, this all relies on von Neumann's and Loomis' theorems. Here is a simple direct proof, taken from Wikipedia but translated to use the notation from the book.

The sum of probabilities is 1.

Yao's Minimax Principle, Proof

$$\begin{aligned}
&\max_{I\in\mathcal{I}} \mathop{\mathbb{E}}_{q}[C(I, A_q)] \\
&= 1 \cdot \max_{I\in\mathcal{I}} \mathop{\mathbb{E}}_{q}[C(I, A_q)] \\
&= \left(\sum_{I'\in\mathcal{I}} \Pr[I_p = I']\right) \cdot \max_{I\in\mathcal{I}} \mathop{\mathbb{E}}_{q}[C(I, A_q)] \\
&= \sum_{I'\in\mathcal{I}} \Pr[I_p = I'] \cdot \max_{I\in\mathcal{I}} \mathop{\mathbb{E}}_{q}[C(I, A_q)] \\
&\geq \sum_{I'\in\mathcal{I}} \Pr[I_p = I'] \cdot \mathop{\mathbb{E}}_{q}[C(I', A_q)] \\
&= \sum_{I'\in\mathcal{I}} \Pr[I_p = I'] \cdot \sum_{A'\in\mathcal{A}} \Pr[A_q = A'] \cdot C(I', A') \\
&= \sum_{I'\in\mathcal{I}} \sum_{A'\in\mathcal{A}} \Pr[I_p = I'] \cdot \Pr[A_q = A'] \cdot C(I', A')
\end{aligned}$$

$$\begin{aligned}
&= \sum_{A'\in\mathcal{A}} \sum_{I'\in\mathcal{I}} \Pr[A_q = A'] \cdot \Pr[I_p = I'] \cdot C(I', A') \\
&= \sum_{A'\in\mathcal{A}} \Pr[A_q = A'] \cdot \sum_{I'\in\mathcal{I}} \Pr[I_p = I'] \cdot C(I', A') \\
&= \sum_{A'\in\mathcal{A}} \Pr[A_q = A'] \cdot \mathop{\mathbb{E}}_{p}[C(I_p, A')] \\
&\geq \sum_{A'\in\mathcal{A}} \Pr[A_q = A'] \cdot \min_{A\in\mathcal{A}} \mathop{\mathbb{E}}_{p}[C(I_p, A)] \\
&= \left(\sum_{A'\in\mathcal{A}} \Pr[A_q = A']\right) \cdot \min_{A\in\mathcal{A}} \mathop{\mathbb{E}}_{p}[C(I_p, A)] \\
&= 1 \cdot \min_{A\in\mathcal{A}} \mathop{\mathbb{E}}_{p}[C(I_p, A)] \\
&= \min_{A\in\mathcal{A}} \mathop{\mathbb{E}}_{p}[C(I_p, A)]
\end{aligned}$$

However, this all relies on von Neumann's and Loomis' theorems. Here is a simple direct proof, taken from Wikipedia but translated to use the notation from the book.
We can move a constant into the sum.

Yao's Minimax Principle, Proof

$$\max_{I \in \mathcal{I}} \mathbb{E}_q[C(I, A_q)]$$
$$= 1 \cdot \max_{I \in \mathcal{I}} \mathbb{E}_q[C(I, A_q)]$$
$$= \left(\sum_{I' \in \mathcal{I}} \Pr[I_p = I'] \right) \cdot \max_{I \in \mathcal{I}} \mathbb{E}_q[C(I, A_q)]$$
$$= \sum_{I' \in \mathcal{I}} \Pr[I_p = I'] \cdot \max_{I \in \mathcal{I}} \mathbb{E}_q[C(I, A_q)]$$
$$\geq \sum_{I' \in \mathcal{I}} \Pr[I_p = I'] \cdot \mathbb{E}_q[C(I', A_q)]$$
$$= \sum_{I' \in \mathcal{I}} \Pr[I_p = I'] \cdot \sum_{A' \in \mathcal{A}} \Pr[A_q = A'] \cdot C(I', A')$$
$$= \sum_{I' \in \mathcal{I}} \sum_{A' \in \mathcal{A}} \Pr[I_p = I'] \cdot \Pr[A_q = A'] \cdot C(I', A')$$
$$= \sum_{A' \in \mathcal{A}} \sum_{I' \in \mathcal{I}} \Pr[A_q = A'] \cdot \Pr[I_p = I'] \cdot C(I', A')$$
$$= \sum_{A' \in \mathcal{A}} \Pr[A_q = A'] \cdot \sum_{I' \in \mathcal{I}} \Pr[I_p = I'] \cdot C(I', A')$$
$$= \sum_{A' \in \mathcal{A}} \Pr[A_q = A'] \cdot \mathbb{E}_p[C(I_p, A')]$$
$$\geq \sum_{A' \in \mathcal{A}} \Pr[A_q = A'] \cdot \min_{A \in \mathcal{A}} \mathbb{E}_p[C(I_p, A)]$$
$$= \left(\sum_{A' \in \mathcal{A}} \Pr[A_q = A'] \right) \cdot \min_{A \in \mathcal{A}} \mathbb{E}_p[C(I_p, A)]$$
$$= 1 \cdot \min_{A \in \mathcal{A}} \mathbb{E}_p[C(I_p, A)]$$
$$= \min_{A \in \mathcal{A}} \mathbb{E}_p[C(I_p, A)]$$

However, this all relies on von Neumann's and Loomis' theorems. Here is a simple direct proof, taken from Wikipedia but translated to use the notation from the book.
The maximum is $\geq$ every element.

Yao's Minimax Principle, Proof

$$\max_{I\in\mathcal{I}} \mathbb{E}_q[C(I,A_q)]$$
$$= 1\cdot \max_{I\in\mathcal{I}} \mathbb{E}_q[C(I,A_q)]$$
$$= \left(\sum_{I'\in\mathcal{I}} \Pr[I_p=I']\right)\cdot \max_{I\in\mathcal{I}} \mathbb{E}_q[C(I,A_q)]$$
$$= \sum_{I'\in\mathcal{I}} \Pr[I_p=I']\cdot \max_{I\in\mathcal{I}} \mathbb{E}_q[C(I,A_q)]$$
$$\geq \sum_{I'\in\mathcal{I}} \Pr[I_p=I']\cdot \mathbb{E}_q[C(I',A_q)]$$
$$= \sum_{I'\in\mathcal{I}} \Pr[I_p=I']\cdot \sum_{A'\in\mathcal{A}} \Pr[A_q=A']\cdot C(I',A')$$
$$= \sum_{I'\in\mathcal{I}} \sum_{A'\in\mathcal{A}} \Pr[I_p=I']\cdot \Pr[A_q=A']\cdot C(I',A')$$
$$= \sum_{A'\in\mathcal{A}} \sum_{I'\in\mathcal{I}} \Pr[A_q=A']\cdot \Pr[I_p=I']\cdot C(I',A')$$
$$= \sum_{A'\in\mathcal{A}} \Pr[A_q=A']\cdot \sum_{I'\in\mathcal{I}} \Pr[I_p=I']\cdot C(I',A')$$
$$= \sum_{A'\in\mathcal{A}} \Pr[A_q=A']\cdot \mathbb{E}_p[C(I_p,A')]$$
$$\geq \sum_{A'\in\mathcal{A}} \Pr[A_q=A']\cdot \min_{A\in\mathcal{A}} \mathbb{E}_p[C(I_p,A)]$$
$$= \left(\sum_{A'\in\mathcal{A}} \Pr[A_q=A']\right)\cdot \min_{A\in\mathcal{A}} \mathbb{E}_p[C(I_p,A)]$$
$$= 1\cdot \min_{A\in\mathcal{A}} \mathbb{E}_p[C(I_p,A)]$$
$$= \min_{A\in\mathcal{A}} \mathbb{E}_p[C(I_p,A)]$$

However, this all relies on von Neumann's and Loomis' theorems. Here is a simple direct proof, taken from Wikipedia but translated to use the notation from the book.
The definition of expected value.

Yao's Minimax Principle, Proof

$$
\begin{aligned}
&\max_{I \in \mathcal{I}} \mathbb{E}_q[C(I, A_q)] \\
&= 1 \cdot \max_{I \in \mathcal{I}} \mathbb{E}_q[C(I, A_q)] \\
&= \left(\sum_{I' \in \mathcal{I}} \Pr[I_p = I'] \right) \cdot \max_{I \in \mathcal{I}} \mathbb{E}_q[C(I, A_q)] \\
&= \sum_{I' \in \mathcal{I}} \Pr[I_p = I'] \cdot \max_{I \in \mathcal{I}} \mathbb{E}_q[C(I, A_q)] \\
&\geq \sum_{I' \in \mathcal{I}} \Pr[I_p = I'] \cdot \mathbb{E}_q[C(I', A_q)] \\
&= \sum_{I' \in \mathcal{I}} \Pr[I_p = I'] \cdot \sum_{A' \in \mathcal{A}} \Pr[A_q = A'] \cdot C(I', A') \\
&= \sum_{I' \in \mathcal{I}} \sum_{A' \in \mathcal{A}} \Pr[I_p = I'] \cdot \Pr[A_q = A'] \cdot C(I', A')
\end{aligned}
$$

$$
\begin{aligned}
&= \sum_{A' \in \mathcal{A}} \sum_{I' \in \mathcal{I}} \Pr[A_q = A'] \cdot \Pr[I_p = I'] \cdot C(I', A') \\
&= \sum_{A' \in \mathcal{A}} \Pr[A_q = A'] \cdot \sum_{I' \in \mathcal{I}} \Pr[I_p = I'] \cdot C(I', A') \\
&= \sum_{A' \in \mathcal{A}} \Pr[A_q = A'] \cdot \mathbb{E}_p[C(I_p, A')] \\
&\geq \sum_{A' \in \mathcal{A}} \Pr[A_q = A'] \cdot \min_{A \in \mathcal{A}} \mathbb{E}_p[C(I_p, A)] \\
&= \left(\sum_{A' \in \mathcal{A}} \Pr[A_q = A'] \right) \cdot \min_{A \in \mathcal{A}} \mathbb{E}_p[C(I_p, A)] \\
&= 1 \cdot \min_{A \in \mathcal{A}} \mathbb{E}_p[C(I_p, A)] \\
&= \min_{A \in \mathcal{A}} \mathbb{E}_p[C(I_p, A)]
\end{aligned}
$$

However, this all relies on von Neumann's and Loomis' theorems. Here is a simple direct proof, taken from Wikipedia but translated to use the notation from the book.
We can move a constant into the sum.

Yao's Minimax Principle, Proof

$$
\begin{aligned}
&\max_{I\in\mathcal{I}} \mathbb{E}_q[C(I,A_q)] \\
&= 1 \cdot \max_{I\in\mathcal{I}} \mathbb{E}_q[C(I,A_q)] \\
&= \left(\sum_{I'\in\mathcal{I}} \Pr[I_p = I']\right) \cdot \max_{I\in\mathcal{I}} \mathbb{E}_q[C(I,A_q)] \\
&= \sum_{I'\in\mathcal{I}} \Pr[I_p = I'] \cdot \max_{I\in\mathcal{I}} \mathbb{E}_q[C(I,A_q)] \\
&\geq \sum_{I'\in\mathcal{I}} \Pr[I_p = I'] \cdot \mathbb{E}_q[C(I',A_q)] \\
&= \sum_{I'\in\mathcal{I}} \Pr[I_p = I'] \cdot \sum_{A'\in\mathcal{A}} \Pr[A_q = A'] \cdot C(I',A') \\
&= \sum_{I'\in\mathcal{I}} \sum_{A'\in\mathcal{A}} \Pr[I_p = I'] \cdot \Pr[A_q = A'] \cdot C(I',A')
\end{aligned}
$$

$$
\begin{aligned}
&= \sum_{A'\in\mathcal{A}} \sum_{I'\in\mathcal{I}} \Pr[A_q = A'] \cdot \Pr[I_p = I'] \cdot C(I',A') \\
&= \sum_{A'\in\mathcal{A}} \Pr[A_q = A'] \cdot \sum_{I'\in\mathcal{I}} \Pr[I_p = I'] \cdot C(I',A') \\
&= \sum_{A'\in\mathcal{A}} \Pr[A_q = A'] \cdot \mathbb{E}_p[C(I_p,A')] \\
&\geq \sum_{A'\in\mathcal{A}} \Pr[A_q = A'] \cdot \min_{A\in\mathcal{A}} \mathbb{E}_p[C(I_p,A)] \\
&= \left(\sum_{A'\in\mathcal{A}} \Pr[A_q = A']\right) \cdot \min_{A\in\mathcal{A}} \mathbb{E}_p[C(I_p,A)] \\
&= 1 \cdot \min_{A\in\mathcal{A}} \mathbb{E}_p[C(I_p,A)] \\
&= \min_{A\in\mathcal{A}} \mathbb{E}_p[C(I_p,A)]
\end{aligned}
$$

However, this all relies on von Neumann's and Loomis' theorems. Here is a simple direct proof, taken from Wikipedia but translated to use the notation from the book.

Both addition and multiplication is commutative.

Yao's Minimax Principle, Proof

$$\begin{aligned}
&\max_{I\in\mathcal{I}} \mathop{\mathbb{E}}_{q}[C(I,A_q)] \\
&= 1 \cdot \max_{I\in\mathcal{I}} \mathop{\mathbb{E}}_{q}[C(I,A_q)] \\
&= \left(\sum_{I'\in\mathcal{I}} \Pr[I_p = I']\right) \cdot \max_{I\in\mathcal{I}} \mathop{\mathbb{E}}_{q}[C(I,A_q)] \\
&= \sum_{I'\in\mathcal{I}} \Pr[I_p = I'] \cdot \max_{I\in\mathcal{I}} \mathop{\mathbb{E}}_{q}[C(I,A_q)] \\
&\geq \sum_{I'\in\mathcal{I}} \Pr[I_p = I'] \cdot \mathop{\mathbb{E}}_{q}[C(I',A_q)] \\
&= \sum_{I'\in\mathcal{I}} \Pr[I_p = I'] \cdot \sum_{A'\in\mathcal{A}} \Pr[A_q = A'] \cdot C(I',A') \\
&= \sum_{I'\in\mathcal{I}} \sum_{A'\in\mathcal{A}} \Pr[I_p = I'] \cdot \Pr[A_q = A'] \cdot C(I',A')
\end{aligned}$$

$$\begin{aligned}
&= \sum_{A'\in\mathcal{A}} \sum_{I'\in\mathcal{I}} \Pr[A_q = A'] \cdot \Pr[I_p = I'] \cdot C(I',A') \\
&= \sum_{A'\in\mathcal{A}} \Pr[A_q = A'] \cdot \sum_{I'\in\mathcal{I}} \Pr[I_p = I'] \cdot C(I',A') \\
&= \sum_{A'\in\mathcal{A}} \Pr[A_q = A'] \cdot \mathop{\mathbb{E}}_{p}[C(I_p,A')] \\
&\geq \sum_{A'\in\mathcal{A}} \Pr[A_q = A'] \cdot \min_{A\in\mathcal{A}} \mathop{\mathbb{E}}_{p}[C(I_p,A)] \\
&= \left(\sum_{A'\in\mathcal{A}} \Pr[A_q = A']\right) \cdot \min_{A\in\mathcal{A}} \mathop{\mathbb{E}}_{p}[C(I_p,A)] \\
&= 1 \cdot \min_{A\in\mathcal{A}} \mathop{\mathbb{E}}_{p}[C(I_p,A)] \\
&= \min_{A\in\mathcal{A}} \mathop{\mathbb{E}}_{p}[C(I_p,A)]
\end{aligned}$$

However, this all relies on von Neumann's and Loomis' theorems. Here is a simple direct proof, taken from Wikipedia but translated to use the notation from the book.
We can move a constant out of the sum.

Yao's Minimax Principle, Proof

$$\max_{I \in \mathcal{I}} \mathbb{E}_q[C(I, A_q)]$$
$$= 1 \cdot \max_{I \in \mathcal{I}} \mathbb{E}_q[C(I, A_q)]$$
$$= \left(\sum_{I' \in \mathcal{I}} \Pr[I_p = I'] \right) \cdot \max_{I \in \mathcal{I}} \mathbb{E}_q[C(I, A_q)]$$
$$= \sum_{I' \in \mathcal{I}} \Pr[I_p = I'] \cdot \max_{I \in \mathcal{I}} \mathbb{E}_q[C(I, A_q)]$$
$$\geq \sum_{I' \in \mathcal{I}} \Pr[I_p = I'] \cdot \mathbb{E}_q[C(I', A_q)]$$
$$= \sum_{I' \in \mathcal{I}} \Pr[I_p = I'] \cdot \sum_{A' \in \mathcal{A}} \Pr[A_q = A'] \cdot C(I', A')$$
$$= \sum_{I' \in \mathcal{I}} \sum_{A' \in \mathcal{A}} \Pr[I_p = I'] \cdot \Pr[A_q = A'] \cdot C(I', A')$$
$$= \sum_{A' \in \mathcal{A}} \sum_{I' \in \mathcal{I}} \Pr[A_q = A'] \cdot \Pr[I_p = I'] \cdot C(I', A')$$
$$= \sum_{A' \in \mathcal{A}} \Pr[A_q = A'] \cdot \sum_{I' \in \mathcal{I}} \Pr[I_p = I'] \cdot C(I', A')$$
$$= \sum_{A' \in \mathcal{A}} \Pr[A_q = A'] \cdot \mathbb{E}_p[C(I_p, A')]$$
$$\geq \sum_{A' \in \mathcal{A}} \Pr[A_q = A'] \cdot \min_{A \in \mathcal{A}} \mathbb{E}_p[C(I_p, A)]$$
$$= \left(\sum_{A' \in \mathcal{A}} \Pr[A_q = A'] \right) \cdot \min_{A \in \mathcal{A}} \mathbb{E}_p[C(I_p, A)]$$
$$= 1 \cdot \min_{A \in \mathcal{A}} \mathbb{E}_p[C(I_p, A)]$$
$$= \min_{A \in \mathcal{A}} \mathbb{E}_p[C(I_p, A)]$$

However, this all relies on von Neumann's and Loomis' theorems. Here is a simple direct proof, taken from Wikipedia but translated to use the notation from the book.
The definition of expected value.

Yao's Minimax Principle, Proof

$$
\begin{aligned}
&\max_{I\in\mathcal{I}} \mathbb{E}_q[C(I,A_q)] \\
&= 1 \cdot \max_{I\in\mathcal{I}} \mathbb{E}_q[C(I,A_q)] \\
&= \left(\sum_{I'\in\mathcal{I}} \Pr[I_p = I']\right) \cdot \max_{I\in\mathcal{I}} \mathbb{E}_q[C(I,A_q)] \\
&= \sum_{I'\in\mathcal{I}} \Pr[I_p = I'] \cdot \max_{I\in\mathcal{I}} \mathbb{E}_q[C(I,A_q)] \\
&\geq \sum_{I'\in\mathcal{I}} \Pr[I_p = I'] \cdot \mathbb{E}_q[C(I',A_q)] \\
&= \sum_{I'\in\mathcal{I}} \Pr[I_p = I'] \cdot \sum_{A'\in\mathcal{A}} \Pr[A_q = A'] \cdot C(I',A') \\
&= \sum_{I'\in\mathcal{I}} \sum_{A'\in\mathcal{A}} \Pr[I_p = I'] \cdot \Pr[A_q = A'] \cdot C(I',A')
\end{aligned}
$$

$$
\begin{aligned}
&= \sum_{A'\in\mathcal{A}} \sum_{I'\in\mathcal{I}} \Pr[A_q = A'] \cdot \Pr[I_p = I'] \cdot C(I',A') \\
&= \sum_{A'\in\mathcal{A}} \Pr[A_q = A'] \cdot \sum_{I'\in\mathcal{I}} \Pr[I_p = I'] \cdot C(I',A') \\
&= \sum_{A'\in\mathcal{A}} \Pr[A_q = A'] \cdot \mathbb{E}_p[C(I_p,A')] \\
&\geq \sum_{A'\in\mathcal{A}} \Pr[A_q = A'] \cdot \min_{A\in\mathcal{A}} \mathbb{E}_p[C(I_p,A)] \\
&= \left(\sum_{A'\in\mathcal{A}} \Pr[A_q = A']\right) \cdot \min_{A\in\mathcal{A}} \mathbb{E}_p[C(I_p,A)] \\
&= 1 \cdot \min_{A\in\mathcal{A}} \mathbb{E}_p[C(I_p,A)] \\
&= \min_{A\in\mathcal{A}} \mathbb{E}_p[C(I_p,A)]
\end{aligned}
$$

However, this all relies on von Neumann's and Loomis' theorems. Here is a simple direct proof, taken from Wikipedia but translated to use the notation from the book.

Every element is $\geq$ the minimum.

Yao's Minimax Principle, Proof

$$\max_{I \in \mathcal{I}} \mathbb{E}_q[C(I, A_q)]$$
$$= 1 \cdot \max_{I \in \mathcal{I}} \mathbb{E}_q[C(I, A_q)]$$
$$= \left(\sum_{I' \in \mathcal{I}} \Pr[I_p = I']\right) \cdot \max_{I \in \mathcal{I}} \mathbb{E}_q[C(I, A_q)]$$
$$= \sum_{I' \in \mathcal{I}} \Pr[I_p = I'] \cdot \max_{I \in \mathcal{I}} \mathbb{E}_q[C(I, A_q)]$$
$$\geq \sum_{I' \in \mathcal{I}} \Pr[I_p = I'] \cdot \mathbb{E}_q[C(I', A_q)]$$
$$= \sum_{I' \in \mathcal{I}} \Pr[I_p = I'] \cdot \sum_{A' \in \mathcal{A}} \Pr[A_q = A'] \cdot C(I', A')$$
$$= \sum_{I' \in \mathcal{I}} \sum_{A' \in \mathcal{A}} \Pr[I_p = I'] \cdot \Pr[A_q = A'] \cdot C(I', A')$$
$$= \sum_{A' \in \mathcal{A}} \sum_{I' \in \mathcal{I}} \Pr[A_q = A'] \cdot \Pr[I_p = I'] \cdot C(I', A')$$
$$= \sum_{A' \in \mathcal{A}} \Pr[A_q = A'] \cdot \sum_{I' \in \mathcal{I}} \Pr[I_p = I'] \cdot C(I', A')$$
$$= \sum_{A' \in \mathcal{A}} \Pr[A_q = A'] \cdot \mathbb{E}_p[C(I_p, A')]$$
$$\geq \sum_{A' \in \mathcal{A}} \Pr[A_q = A'] \cdot \min_{A \in \mathcal{A}} \mathbb{E}_p[C(I_p, A)]$$
$$= \left(\sum_{A' \in \mathcal{A}} \Pr[A_q = A']\right) \cdot \min_{A \in \mathcal{A}} \mathbb{E}_p[C(I_p, A)]$$
$$= 1 \cdot \min_{A \in \mathcal{A}} \mathbb{E}_p[C(I_p, A)]$$
$$= \min_{A \in \mathcal{A}} \mathbb{E}_p[C(I_p, A)]$$

However, this all relies on von Neumann's and Loomis' theorems. Here is a simple direct proof, taken from Wikipedia but translated to use the notation from the book.
We can move a constant out of the sum.

Yao's Minimax Principle, Proof

$$
\begin{aligned}
&\max_{I\in\mathcal{I}} \underset{q}{\mathbb{E}}[C(I, A_q)] \\
&= 1 \cdot \max_{I\in\mathcal{I}} \underset{q}{\mathbb{E}}[C(I, A_q)] \\
&= \left(\sum_{I'\in\mathcal{I}} \Pr[I_p = I']\right) \cdot \max_{I\in\mathcal{I}} \underset{q}{\mathbb{E}}[C(I, A_q)] \\
&= \sum_{I'\in\mathcal{I}} \Pr[I_p = I'] \cdot \max_{I\in\mathcal{I}} \underset{q}{\mathbb{E}}[C(I, A_q)] \\
&\geq \sum_{I'\in\mathcal{I}} \Pr[I_p = I'] \cdot \underset{q}{\mathbb{E}}[C(I', A_q)] \\
&= \sum_{I'\in\mathcal{I}} \Pr[I_p = I'] \cdot \sum_{A'\in\mathcal{A}} \Pr[A_q = A'] \cdot C(I', A') \\
&= \sum_{I'\in\mathcal{I}} \sum_{A'\in\mathcal{A}} \Pr[I_p = I'] \cdot \Pr[A_q = A'] \cdot C(I', A')
\end{aligned}
$$

$$
\begin{aligned}
&= \sum_{A'\in\mathcal{A}} \sum_{I'\in\mathcal{I}} \Pr[A_q = A'] \cdot \Pr[I_p = I'] \cdot C(I', A') \\
&= \sum_{A'\in\mathcal{A}} \Pr[A_q = A'] \cdot \sum_{I'\in\mathcal{I}} \Pr[I_p = I'] \cdot C(I', A') \\
&= \sum_{A'\in\mathcal{A}} \Pr[A_q = A'] \cdot \underset{p}{\mathbb{E}}[C(I_p, A')] \\
&\geq \sum_{A'\in\mathcal{A}} \Pr[A_q = A'] \cdot \min_{A\in\mathcal{A}} \underset{p}{\mathbb{E}}[C(I_p, A)] \\
&= \left(\sum_{A'\in\mathcal{A}} \Pr[A_q = A']\right) \cdot \min_{A\in\mathcal{A}} \underset{p}{\mathbb{E}}[C(I_p, A)] \\
&= 1 \cdot \min_{A\in\mathcal{A}} \underset{p}{\mathbb{E}}[C(I_p, A)] \\
&= \min_{A\in\mathcal{A}} \underset{p}{\mathbb{E}}[C(I_p, A)]
\end{aligned}
$$

However, this all relies on von Neumann's and Loomis' theorems. Here is a simple direct proof, taken from Wikipedia but translated to use the notation from the book.
The sum of probabilities is 1.

Yao's Minimax Principle, Proof

$$
\begin{aligned}
&\max_{I \in \mathcal{I}} \mathop{\mathbb{E}}_{q}[C(I, A_q)] \\
&= 1 \cdot \max_{I \in \mathcal{I}} \mathop{\mathbb{E}}_{q}[C(I, A_q)] \\
&= \left(\sum_{I' \in \mathcal{I}} \Pr[I_p = I'] \right) \cdot \max_{I \in \mathcal{I}} \mathop{\mathbb{E}}_{q}[C(I, A_q)] \\
&= \sum_{I' \in \mathcal{I}} \Pr[I_p = I'] \cdot \max_{I \in \mathcal{I}} \mathop{\mathbb{E}}_{q}[C(I, A_q)] \\
&\geq \sum_{I' \in \mathcal{I}} \Pr[I_p = I'] \cdot \mathop{\mathbb{E}}_{q}[C(I', A_q)] \\
&= \sum_{I' \in \mathcal{I}} \Pr[I_p = I'] \cdot \sum_{A' \in \mathcal{A}} \Pr[A_q = A'] \cdot C(I', A') \\
&= \sum_{I' \in \mathcal{I}} \sum_{A' \in \mathcal{A}} \Pr[I_p = I'] \cdot \Pr[A_q = A'] \cdot C(I', A')
\end{aligned}
$$

$$
\begin{aligned}
&= \sum_{A' \in \mathcal{A}} \sum_{I' \in \mathcal{I}} \Pr[A_q = A'] \cdot \Pr[I_p = I'] \cdot C(I', A') \\
&= \sum_{A' \in \mathcal{A}} \Pr[A_q = A'] \cdot \sum_{I' \in \mathcal{I}} \Pr[I_p = I'] \cdot C(I', A') \\
&= \sum_{A' \in \mathcal{A}} \Pr[A_q = A'] \cdot \mathop{\mathbb{E}}_{p}[C(I_p, A')] \\
&\geq \sum_{A' \in \mathcal{A}} \Pr[A_q = A'] \cdot \min_{A \in \mathcal{A}} \mathop{\mathbb{E}}_{p}[C(I_p, A)] \\
&= \left(\sum_{A' \in \mathcal{A}} \Pr[A_q = A'] \right) \cdot \min_{A \in \mathcal{A}} \mathop{\mathbb{E}}_{p}[C(I_p, A)] \\
&= 1 \cdot \min_{A \in \mathcal{A}} \mathop{\mathbb{E}}_{p}[C(I_p, A)] \\
&= \min_{A \in \mathcal{A}} \mathop{\mathbb{E}}_{p}[C(I_p, A)]
\end{aligned}
$$

However, this all relies on von Neumann's and Loomis' theorems. Here is a simple direct proof, taken from Wikipedia but translated to use the notation from the book.
Drop multiplication by 1.

Yao's Minimax Principle, MC

A similar theorem exists for Monte Carlo algorithms. Let $C_\epsilon(I, A)$ denote the cost of running deterministic algorithm A on input I to get error probability at most ϵ.

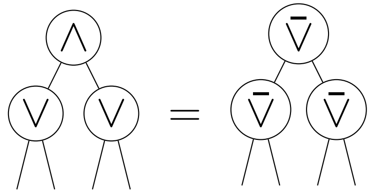
Theorem

For all distributions $\mathbf{p}$ *over* $\mathcal{I}$ *and* $\mathbf{q}$ *over* $\mathcal{A}$ *and any* $\epsilon \in [0, \frac{1}{2}]$,

$$\frac{1}{2} \min_{A \in \mathcal{A}} \mathop{\mathbb{E}}_{\mathbf{p}}[C_{2\epsilon}(I_{\mathbf{p}}, A)] \leq \max_{I \in \mathcal{I}} \mathop{\mathbb{E}}_{\mathbf{q}}[C_\epsilon(I, A_{\mathbf{q}})]$$

GTE Lower Bound, NOR tree

Define a boolean NOR tree as a rooted tree where every internal node is labelled $\bar{\vee}$ (NOR = NOT OR), and every leaf is $\in \{0, 1\}$.



$T_{2,k}$ is equivalent to a binary boolean NOR tree where every leaf is at depth $2k$ (Exercise 2.2).

We have seen a technique for deriving lower bounds for randomized algorithms. Let's use it on the game tree evaluation problem.

GTE Lower Bound, Input

Suppose we have chosen the leaves of a binary NOR tree randomly such that for each child c of a node v

$$\Pr[c = 1] = p = \frac{3-\sqrt{5}}{2}$$

Then

$$\Pr[v = 1] = (1-p)^2 = \Big(\frac{2-(3-\sqrt{5})}{2}\Big)^2$$
$$= \Big(\frac{\sqrt{5}-1}{2}\Big)^2 = \frac{5+1-2\sqrt{5}}{4} = p$$

GTE Lower Bound, Input

If we choose the leaves of a binary NOR tree randomly such that each for each leaf node v

$$\Pr[v = 1] = p = \frac{3 - \sqrt{5}}{2}$$

Then every node evaluates to 1 with probability p

GTE Lower Bound, Algorithm

Call a NOR-Tree evaluation algorithm *depth-first pruning* if, for each node v that it evaluates, it completely evaluates one of its children before evaluating any leaves in the subtree rooted at the other child.

GTE Lower Bound, Algorithm

Call a NOR-Tree evaluation algorithm *depth-first pruning* if, for each node v that it evaluates, it completely evaluates one of its children before evaluating any leaves in the subtree rooted at the other child.

Lemma

For any input distribution where leaves are independently 1 *with fixed probability* q*, some optimal deterministic algorithm is depth-first pruning.*

We are not going to prove this lemma, but it allows us to restrict our attention to depth-first pruning algorithms.

GTE Lower Bound, Analysis

Let each leaf be independently 1 with probability $p = \frac{3-\sqrt{5}}{2}$. Let $W(h)$ be the expected number of leaves inspected when evaluating a node at height h. Then

$$W(h) = \begin{cases} 1 & \text{if } h = 0\text{; otherwise} \\ W(h-1) + (1-p) \cdot W(h-1) \end{cases}$$

$$= (2-p)^h = \left(\frac{1+\sqrt{5}}{2}\right)^h = \varphi^h$$

$$W(\log_2 n) = \varphi^{\log_2 n} = n^{\log_2 \varphi} = n^{0.694\ldots}$$

Note the appearance of the golden ratio

GTE Lower Bound, Summary

We have shown that the expected cost of $\textsc{RandBoolGTE}$ is at most $n^{log_4 3} = n^{0.793\ldots}$, and we have shown that the cost for *any* Las Vegas GTE algorithm is at least $n^{\log_2 \varphi} = n^{0.694\ldots}$.

GTE Lower Bound, Summary

We have shown that the expected cost of $\textsc{RandBoolGTE}$ is at most $n^{log_4 3} = n^{0.793\dots}$, and we have shown that the cost for *any* Las Vegas GTE algorithm is at least $n^{\log_2 \varphi} = n^{0.694\dots}$.

Why the gap? (The book is misleading!)

Our distribution has each node independently 1 with probability p, so there is a p^2 chance both children of a node are 1 which is the easy case.

A harder distribution eliminates this possibility by making the nodes dependent. This is much harder to analyze however.

Similarly, our analysis of $\textsc{RandBoolGTE}$ is a bit sloppy because it ignores that we know there is a difference in the expected values depending on what the final value is.

GTE Lower Bound, Summary

We have shown that the expected cost of $\textsc{RandBoolGTE}$ is at most $n^{log_4 3} = n^{0.793\ldots}$, and we have shown that the cost for *any* Las Vegas GTE algorithm is at least $n^{\log_2 \varphi} = n^{0.694\ldots}$.

Why the gap? (The book is misleading!)

True expected cost of $\textsc{RandBoolGTE}$ is $\mathcal{O}\left(n^{\log_2\left(\frac{1+\sqrt{33}}{4}\right)}\right) = \mathcal{O}(n^{0.754\ldots})$, and this is optimal [Saks & Wigderson'86].

Our distribution has each node independently 1 with probability p, so there is a p^2 chance both children of a node are 1 which is the easy case.
A harder distribution eliminates this possibility by making the nodes dependent. This is much harder to analyze however.
Similarly, our analysis of $\textsc{RandBoolGTE}$ is a bit sloppy because it ignores that we know there is a difference in the expected values depending on what the final value is.

When is Randomness Necessary?

When can we derandomize?

When is Randomness Necessary?

When can we derandomize?

This question is hard!

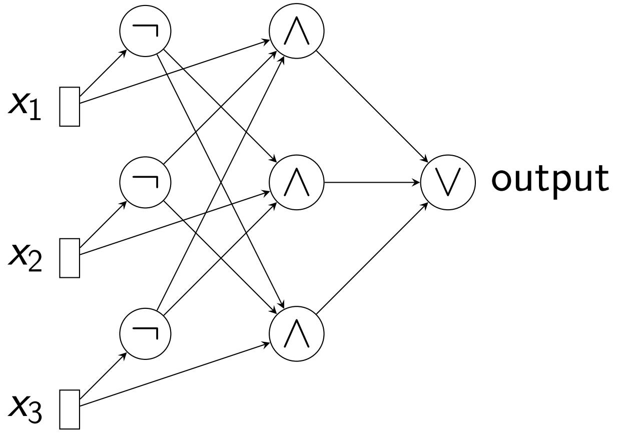
Boolean Circuit, Definition

A *Boolean Circuit* for $f(x_1, \ldots, x_n)$ is a directed graph where:

- n *input* vertices have in-degree 0. A single *output* vertex has out-degree 0.
- Non-input vertices are marked "$\vee$", "$\wedge$", or "$\neg$". A "$\neg$" vertex has in-degree 1.
- The size of a circuit is the total number of vertices.

output $= f(x_1, \ldots, x_n)$

Boolean Circuit, Example



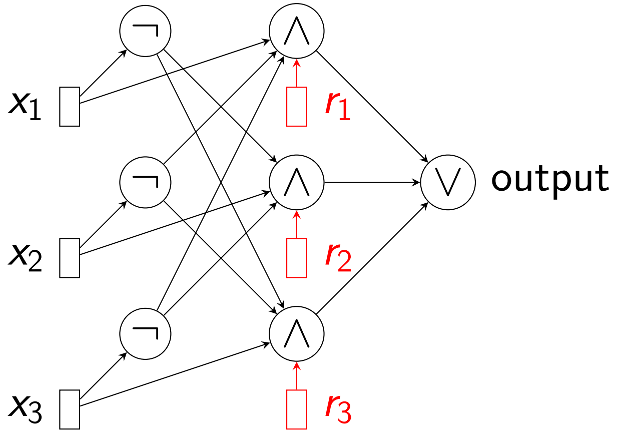
Randomized Boolean Circuit, Def.

A *Randomized Boolean Circuit* for $f(x_1, \ldots, x_n)$ is a directed graph where:

- *n input*, and *m random* vertices have in-degree 0. A single *output* vertex has out-degree 0.
- Non-input vertices are marked "$\vee$", "$\wedge$", or "$\neg$". A "$\neg$" vertex has in-degree 1.
- The size of a circuit is the total number of vertices.

$$\Pr[\text{output} = f(x_1, \ldots, x_n)] \geq \begin{cases} 1 & \text{if } f(x_1, \ldots, x_n) = 0 \\ \frac{1}{2} & \text{if } f(x_1, \ldots, x_n) = 1 \end{cases}$$

Rand. Boolean Circuit, Example



Circuit Family, Definition

Given a function $f : \{0,1\}^\star \to \{0,1\}$, let f_n be the restriction of f to $\{0,1\}^n$. A sequence $\mathcal{C} = C_1, C_2, \ldots$ of boolean circuits is a *circuit family* for f if C_n computes $f_n(x_1, \ldots, x_n)$.
It is *polynomially bounded*, if $|C_n| \leq p(n)$ for some polynomial p.

Randomized Circuit Family, Def.

Given a function $f : \{0,1\}^\star \to \{0,1\}$, let f_n be the restriction of f to $\{0,1\}^n$. A sequence $\mathcal{C} = C_1, C_2, \ldots$ of randomized boolean circuits is a *randomized circuit family* for f if C_n computes $f_n(x_1, \ldots, x_n)$.
It is *polynomially bounded*, if $|C_n| \leq p(n)$ for some polynomial p.

Adleman's Theorem

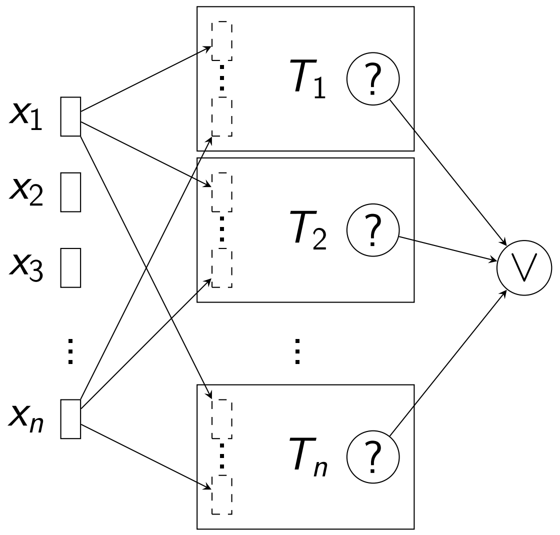
Theorem

If a boolean function has a randomized polynomial-sized circuit family, then it has a polynomial-sized circuit family.

Adleman's Theorem, Proof

Idea: Convert each randomized circuit C_n of size $|C_n| \leq p(n)$ into a deterministic circuit D_n of size $|D_n| \leq n \cdot |C_n|$.
We do this by constructing a sequence of n boolean circuits $T_1, \ldots, T_n$ where each T_i correctly answers at least half the inputs not answered by $T_1, \ldots, T_{i-1}$, and then combining the answers with a single $\vee$ node.

Adleman's Theorem, Proof



Adleman's Theorem, Proof

Each T_i is constructed from C_n by *fixing* each random node r_j to some value $r_j(i)$, and *reducing* the circuit. The resulting T_i has size $|T_i| \leq |C_n| - m$.

Thus

$|D_i| \leq n \cdot (|C_n| - (m + n)) + n + 1 \leq n \cdot |C_n|$.

Need to prove we can pick $r_j(i)$.

Adleman's Theorem, Proof

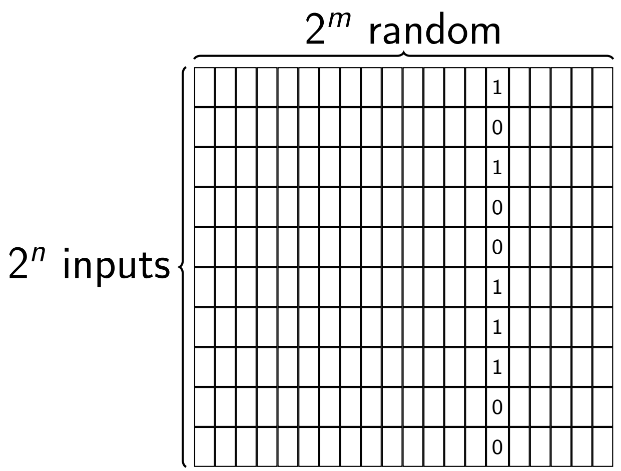
Let M be a $2^n \times 2^m$ boolean matrix where rows correspond to all possible inputs, columns correspond to all possible random choices, and $M_{ij} = 1$ iff the random bits corresponding to j is a *witness* (C_n computes a 1) for the input corresponding to i. Eliminate each row i where f_n is 0 on the corresponding input.

Adleman's Theorem, Proof

By definition of randomized circuit, at least half the entries of each remaining row is 1. Thus at least half of *all* bits are 1, and there must be a column j where at least half the bits are 1. Let $r_1(1), \ldots, r_m(1)$ be the random bits corresponding to j.

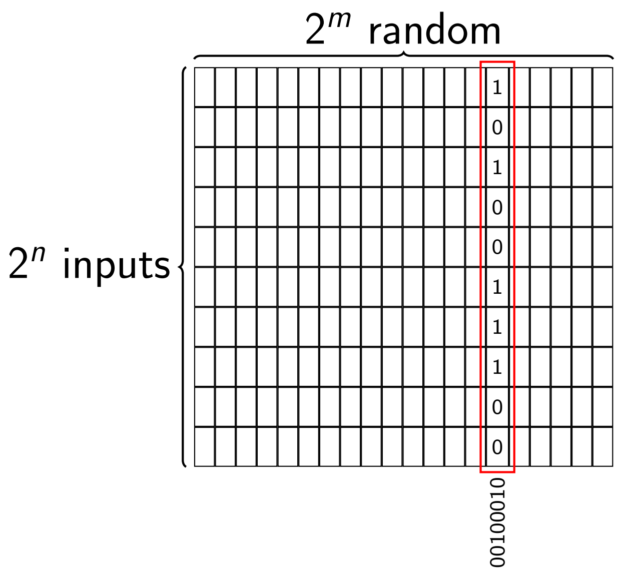
Delete column j and all rows where column j had a 1 from M. Remaining rows still have at least half the bits set, so we can repeat to compute each $r_1(i), \ldots, r_m(i)$. $\square$

Adleman's Theorem, Proof



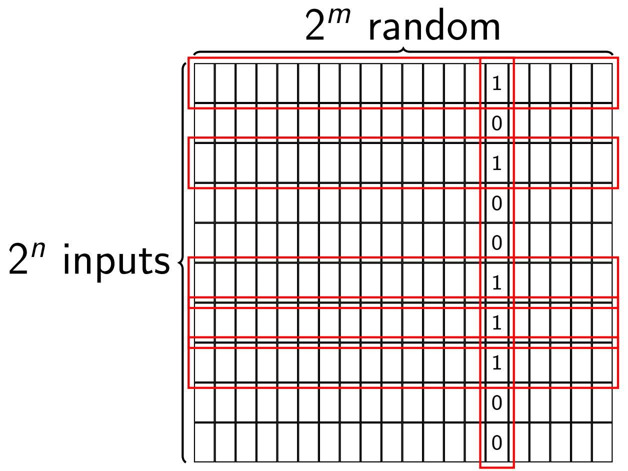
At least half of each row is 1.
So at least half of everything is 1.
So at least one column j has at least half its entries 1.
Pich the random bits corresponding to column j.
After deleting column j and every row with a 1 in column j, the remaining matrix still has at least half its entries 1, so repeat.

Adleman's Theorem, Proof



At least half of each row is 1.
So at least half of everything is 1.
So at least one column j has at least half its entries 1.
Pich the random bits corresponding to column j.
After deleting column j and every row with a 1 in column j, the remaining matrix still has at least half its entries 1, so repeat.

Adleman's Theorem, Proof



At least half of each row is 1.
So at least half of everything is 1.
So at least one column j has at least half its entries 1.
Pich the random bits corresponding to column j.
After deleting column j and every row with a 1 in column j, the remaining matrix still has at least half its entries 1, so repeat.

Adleman's Theorem, Summary

Does this mean we can always eliminate randomness?

Adleman's Theorem, Summary

Does this mean we can always eliminate randomness? No!
Why not?

Adleman's Theorem, Summary

Does this mean we can always eliminate randomness? No!
Why not? Non-uniformity: The transformation works for each specific n, the circuit for $n+1$ may be completely different.

Complexity Theory, Uniformity

Let $a(n) : \mathbb{N} \to \Sigma^\star$. We say algorithm A *decides* language L with advice a if on an input x it uses the read-only string $a(|x|)$ to decide the membership of x in L.
An algorithm is *non-uniform* if it needs advice, and *uniform* otherwise.

Complexity Theory, Uniformity

For complexity class **P** define **P**/ poly to be all languages that have a non-uniform polynomial-time algorithm A using advice a such that $|a(n)| \in \mathcal{O}(\text{poly}(n))$.

Complexity Theory, Uniformity

For complexity class **RP** define **RP**/ poly to be all languages that have a non-uniform randomized polynomial-time algorithm A using advice a such that $|a(n)| \in \mathcal{O}(\text{poly}(n))$.

Complexity Theory, Summary

Adleman's Theorem can be interpreted in the language of complexity theory as saying $\mathbf{RP}/\,\text{poly} \subseteq \mathbf{P}/\,\text{poly}$.

Combining with the obvious $\mathbf{RP} \subseteq \mathbf{RP}/\,\text{poly}$ we get $\mathbf{RP} \subseteq \mathbf{P}/\,\text{poly}$.

Complexity Theory, Summary

Adleman's Theorem can be interpreted in the language of complexity theory as saying $\mathbf{RP}/\,\mathrm{poly} \subseteq \mathbf{P}/\,\mathrm{poly}$.

Combining with the obvious $\mathbf{RP} \subseteq \mathbf{RP}/\,\mathrm{poly}$ we get $\mathbf{RP} \subseteq \mathbf{P}/\,\mathrm{poly}$.

What does this mean?

Summary

- We saw a randomized algorithm beat the best deterministic algorithm for GTE.
- Then we saw a general technique (Yao's principle) that can be used to convert a lower bound for a deterministic algorithm on random inputs to a lower bound for randomized algorithms on any input.
- We used this to show a lower bound for GTE.
- Then we saw that for any fixed n, a randomized boolean circuit with n inputs can always be derandomized.
- Finally, we saw why that does not in practice allow us to derandomize after all.

Summary

- We saw a randomized algorithm beat the best deterministic algorithm for GTE.
- Then we saw a general technique (Yao's principle) that can be used to convert a lower bound for a deterministic algorithm on random inputs to a lower bound for randomized algorithms on any input.
- We used this to show a lower bound for GTE.
- Then we saw that for any fixed n, a randomized boolean circuit with n inputs can always be derandomized.
- Finally, we saw why that does not in practice allow us to derandomize after all.

Summary

- We saw a randomized algorithm beat the best deterministic algorithm for GTE.
- Then we saw a general technique (Yao's principle) that can be used to convert a lower bound for a deterministic algorithm on random inputs to a lower bound for randomized algorithms on any input.
- We used this to show a lower bound for GTE.
- Then we saw that for any fixed n, a randomized boolean circuit with n inputs can always be derandomized.
- Finally, we saw why that does not in practice allow us to derandomize after all.

Summary

- We saw a randomized algorithm beat the best deterministic algorithm for GTE.
- Then we saw a general technique (Yao's principle) that can be used to convert a lower bound for a deterministic algorithm on random inputs to a lower bound for randomized algorithms on any input.
- We used this to show a lower bound for GTE.
- Then we saw that for any fixed n, a randomized boolean circuit with n inputs can always be derandomized.
- Finally, we saw why that does not in practice allow us to derandomize after all.

Summary

- We saw a randomized algorithm beat the best deterministic algorithm for GTE.
- Then we saw a general technique (Yao's principle) that can be used to convert a lower bound for a deterministic algorithm on random inputs to a lower bound for randomized algorithms on any input.
- We used this to show a lower bound for GTE.
- Then we saw that for any fixed n, a randomized boolean circuit with n inputs can always be derandomized.
- Finally, we saw why that does not in practice allow us to derandomize after all.