Yao's minimax principle

In computational complexity theory, **Yao's principle** or **Yao's minimax principle** states that the expected cost of a randomized algorithm on the worst-case input, is no better than a worst-case random probability distribution of the deterministic algorithm which performs best for that distribution. Thus, to establish a lower bound on the performance of randomized algorithms, it suffices to find an appropriate distribution of difficult inputs, and to prove that no deterministic algorithm can perform well against that distribution. This principle is named after Andrew Yao, who first proposed it.

Yao's principle may be interpreted in game theoretic terms, via a two-player zero sum game in which one player, Alice, selects a deterministic algorithm, the other player, Bob, selects an input, and the payoff is the cost of the selected algorithm on the selected input. Any randomized algorithm \( R \) may be interpreted as a randomized choice among deterministic algorithms, and thus as a strategy for Alice. By von Neumann's minimax theorem, Bob has a randomized strategy that performs at least as well against \( R \) as it does against the best pure strategy Alice might choose; that is, Bob's strategy defines a distribution on the inputs such that the expected cost of \( R \) on that distribution (and therefore also the worst case expected cost of \( R \)) is no better than the expected cost of any single deterministic algorithm against the same distribution.

Statement

Let us state the principle for **Las Vegas** randomized algorithms, i.e., distributions over deterministic algorithms that are correct on every input but have varying costs. It is straightforward to adapt the principle to **Monte Carlo** algorithms, i.e., distributions over deterministic algorithms that have bounded costs but can be incorrect on some inputs.

Consider a problem over the inputs \( \mathcal{X} \), and let \( \mathcal{A} \) be the set of all possible deterministic algorithms that correctly **solve the problem**. For any algorithm \( a \in \mathcal{A} \) and input \( x \in \mathcal{X} \), let \( c(a, x) \geq 0 \) be the cost of algorithm \( a \) run on input \( x \).

Let \( P \) be a probability distributions over the algorithms \( \mathcal{A} \), and let \( A \) denote a random algorithm chosen according to \( P \). Let \( Q \) be a probability distribution over the inputs \( \mathcal{X} \), and let \( X \) denote a random input chosen according to \( Q \). Then,

\[
\max_{x \in \mathcal{X}} E[c(A, x)] \geq \min_{a \in \mathcal{A}} E[c(a, X)],
\]

i.e., the worst-case expected cost of the randomized algorithm is at least the cost of the best deterministic algorithm against input distribution \( Q \).

Proof
Let \( C = \max_{x \in X} \mathbb{E}[c(A, x)] \). For every input \( x \), we have \( \sum_a p_a c(a, x) \leq C \). Therefore,

\[
\sum_x q_x \sum_a p_a c(a, x) \leq C.
\]

Using Fubini’s theorem, since all terms are non-negative we can switch the order of summation, giving

\[
\sum_a p_a \sum_x q_x c(a, x) \leq C.
\]

By the Pigeonhole principle, there must exist an algorithm \( a \) so that

\[
\sum_x q_x c(a, x) \leq C
\]

concluding the proof.