How can we know when our algorithm is optimal (asymptotically)?

Is there a sorting algorithm with asymptotic time complexity (worst-case or expected case) better than $O(n \log n)$?

We can't prove any limitation (lower bound) without basing it on some model of computation.

**Model of Computation**

Comparison-based model: you can only learn about relative order of elements through a comparison.

**Observation:**

For a comparison-based alg.,

\[ \text{time complexity} \geq \# \text{ of comparisons} \]

Can I prove a statement of form: Any comparison-based alg to sort $n$ elements makes $\geq F(n)$ comparisons?

**Adversary Lower Bound**

View as a game with 2 players

- **Adversary (Devil D)**
  - "Pick" \( \# \) between 1 and 100
  - "Yes or no (can't lie)"

- **Algorithm A**
  - To play "20 questions"
Define adversary strategy
Must describe how to reply to whatever question the alg asks in a way that all answers are consistent with some 'input' picked

Goal of adv: max # rounds
(one question answer/round)

Alg: tries to minimize the # of rounds

What's a good adv strategy for 20 questions.
Adv can make a list L with all possible #s

\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9 \}

Alg is # <= 9?
 If yes 1, 2, 3 alive
 No 4, 5, 6, 7, 8 alive

Correct
Alg cannot be done until \( |L| = 1 \).
Why not? Whatever the alg says the answer is, the adv can report that he had a different answer all along.

This answer is an input that causes the # comparisons to be # of rounds in the game

Goal:
For any comparison-based alg to solve problem P there exist an input of size \( \geq f(n) \)
of which computation in worst-case
Analyze adv. strategy we gave
For "20 questions" where # adv picks 1, ..., n
Initially |L_0| = n

Question: How many rounds must occur before |L_i| could be 1. (maybe more rounds are needed)

Initially |L_i| = n
If L_u is # of elements in |L_i|
after round i (L_0 = n)

L_{i+1} \geq \lceil L_u/2 \rceil

# rounds until |L_i| = 1 \geq \lceil \log_2 n \rceil

Ex: Q_1, Q_2, Q_3, Q_4
n = 16, 8, 4, 2, 1

\log_2 16 = 4

n = 17, Q_1, Q_2, Q_3, Q_4, Q_5
17 \rightarrow 9 \rightarrow 5 \rightarrow 3 \rightarrow 2 \rightarrow 1