Adversary Lower Bound

Adversary = Devil

Often we analyze alg based on some "resource" that dominates cost

time complexity measured by # statements executed
Lower bound \( \geq \) upper bound \( \leq \)

Show that any alg must use a given resource \( \geq f(n) \) times.
N coins, 1 fake coin (light)

Prove any alg
must use scale \( \geq \lceil \log_3 n \rceil \)

\( \Rightarrow \)

Any alg must execute \( \geq \lceil \log_3 n \rceil \) statements

\( \Rightarrow \) Time complexity \( \Omega (\log n) \)
Adversary strategy

Keep a list \( L \) of possible ans. (\( n \) possibilities for fake coin)

Until \( |L| = 1 \)

Alg asks a question (weighs)

Adv For each element in \( L \)

Write down answer for question

Picks most frequent answer \( x \)

Remove from \( L \) possibilities don't answer \( x \)
Let $b$ be the number of possible answers to the question.

Scale, $b = 3$

Let $x$ be $|L|^{a_1, a_2, a_3}$

Possible solutions $x \in \{ \frac{1}{2}, \frac{1}{3}, a_1, a_2, a_3 \}$

Majority gets $\geq \sqrt[3]{\frac{x}{3}}$ in general.
Initially, we divide $115$ by $6$ (taking ceiling). We can rewrite: \[
\begin{align*}
115 &= 115 \div 6 \\
20 &= 115 \div 6
\end{align*}
\]

Initially, \[
\begin{align*}
L - 1 &\rightarrow \sqrt{L/b} \\
\sqrt{115} &\rightarrow \sqrt{115/b}
\end{align*}
\]

after question. How many times can \[
\frac{115}{b} \leq \frac{L}{b}
\]
# questions made by any alg

\[ \geq \log_b \left\lfloor \frac{n}{2} \right\rfloor \]

L must be a list of inputs/sols that are distinct

initial list size
Comparison-based alg

resource is $A[i] \leq A[j]$?

Minimum

Sort $A[1], \ldots, A[n]$?

how big can you make L $\leq n!$.

1, 2, 3

1, 3, 2
Sorting

time \geq \# \text{ comparisons} \geq \lceil \log_2 n! \rceil

\text{Stirling's approx} \quad n! \approx (\frac{n}{e})^n

\# \text{ comparisons} \geq \lceil \log_2 (\frac{n}{e})^n \rceil = \Omega(n \log n)

n(\log_2 n - \log_2 e) \approx n \log_2 n
Finding \( \text{min} \)

\[ |L| = n \]

\# comparisons \( \geq \sqrt{\log_2 n} \)
\[ n \log_2 n \]

\[ n - 1 \]

\[ \log_2 n \]

Possible

Others
Sort in $O(n)$ time

Sort 1,000,000,000 phone numbers according to area code

Counting sort
Stable sort

c, a, b, c, d → a, b, b, c, d

equal elements stay in same relative order
Counting Sort (input, output, k)

For (i = 0; i < k; i++)
    count[i] = 0

for (j = 0; j < n; j++)
    Count[input[j]]++;

for (i = 1; i < k; i++)
    count[i] += count[i-1];

for (j = n-1; j >= 0; j--)
    Output[--count[input[j]]] = input[j]

O(n+k)
What if you want to sort Social Security number

*** XXXX XXXX 9 digits

What is k? $10^9$
radix sort