Adversary Lower Bound Technique

Let's first complete our discussion of quicksort.

Expected # of comparisons
\[ \sum_{i=0}^{n-2} \sum_{j=i+1}^{n-1} \text{prob (elements in positions } i \text{ and } j \text{ are compared)} \]

over all \( i,j \) pairs
What is probability that the elements in positions $i$ and $j$ are compared.

Sort this subarray and it contains pos $i$ and pos $j$.

This occurs if $u_i$ or $u_j$ are selected as pivot.

\[
\text{Prob}[u_i \text{ vs. } u_j \text{ compared}] = \frac{2}{\text{# elements in subarray}} \leq \frac{2}{j-i+1}
\]
So,

\[
\sum_{i=0}^{n-2} \sum_{j=i+1}^{n-1} \text{prob} (\text{elements in positions } i \text{ and } j \text{ are compared}) \leq \sum_{i=0}^{n-2} \sum_{j=i+1}^{n-1} \frac{1}{j-i+1} = \sum_{i=0}^{n-2} 2 \left( \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n-i} \right) \\
\leq \sum_{i=0}^{n-2} 2 \left( 1 + \frac{1}{2} + \cdots + \frac{1}{n} \right) \\
= \sum_{i=0}^{n-2} 2 (\ln n + 1) = 2 (n-1) (\ln n + 1) = \Theta (n \log n)
\]
One common optimization is as follows.

1. Terminate when subarray size is relatively small (n ~ 30)

   not sorted  not sorted  ...  not sorted

   ~30   ~30   ~30

   all ≤ all ≤ ... ≤ all

2. Run insertion sort after quick sort is done
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

"Picks" a # between 1 to 100

Is your # <= x

Yes or no (can't "lie")

Adversary
(Devil D)

Round

Comparison-based Algorithm A
to play "20 Question"
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 numbers:

\[
\{1, 2, 3, 4, 5, 6, 7, 8, \_\, 10\}
\]

Alg: Is \( \# < 9 \)?

- Yes means 1-8 “alive”
- No means 9-10 “alive”

Is \( \# < 4 \)?

- 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 # of comparisons to be # of rounds in the game.
Goal:

For any comparison-based alg to solve problem P, there exist an input of size $n$ for which computation time is $\geq F(n)$ in worst-case.
Analyze adv. strategy we gave
For “20 questions” where #
adv picks 1, ..., n

Initially |L| = N

Question: How many rounds
must occur before |L| could
be 1. (maybe more rounds
are needed)
Initially \( |L_1| = n \)

If \( L_i \) is the number of elements in \( |L| \) after round \( i \) (\( L_0 = n \))

\[
L_{i+1} \geq \lceil \frac{L_i}{2} \rceil
\]

\# rounds until \( |L| = 1 \)

\[
\geq \lceil \log_2 n \rceil
\]
Ex

\[ n = 16, 8, 4, 2, 1 \]

\[ \log_2 16 = 4 \]

\[ n = 17, 9, 5, 3, 2, 1 \]
Return to problem of sorting $n$ comparable elements with a comparison based alg.

As adversary, what should I put in list $L$?

$\# \text{ comparisons} \geq \lceil \log_2 n! \rceil$
Ex) $n=3$

<table>
<thead>
<tr>
<th>No</th>
<th>1, 2, 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>No</td>
<td>2, 3, 1</td>
</tr>
<tr>
<td>Yes</td>
<td>3, 1, 2</td>
</tr>
<tr>
<td>Yes</td>
<td>2, 1, 3</td>
</tr>
<tr>
<td>No</td>
<td>1, 3, 2</td>
</tr>
</tbody>
</table>

From alg $a_0 > a_1$?

Yes

There are all different permutations.

What alg will output is something like $a_1 < a_2 < a_0$.

can't have both $[3, 2, 1, 0, 8, 5]$. 
In general $L$ begins with $N!$ inputs (one for each permutation).

Adv strategy: Answer each comparison based on majority in $L$. 
Stirling's approximation

\[ n! \geq \left( \frac{n}{e} \right)^n \]

\[ \text{euler's constant} \]

\# \text{comp} \geq \left\lceil \log_2 \left( \frac{n}{e} \right)^n \right\rceil \]

\[ = \left\lceil n \left( \log_2 n - \log_2 e \right) \right\rceil \]

\[ = \left\lceil n \log_2 n - n \log_2 e \right\rceil = \Theta(n \log n) \]
Consider finding minimum element in an array of \( n \) elements.

What's wrong with starting with the following list when \( n = 3 \)?

\[
\begin{align*}
\text{min} & \quad \{1, 2, 3\} \\
& \quad 2, 1, 3, 2 \\
& \quad 3, 1, 2, \underline{1}, 3, 2, 1
\end{align*}
\]
Adv can have a list L with N inputs where min element is in a different position.

\[ \Rightarrow \text{# comparisons to find min} \geq \lceil \log_2 n \rceil \]
others 2, 3, 4, 5, 6
Different adversary strategy

possible positions for min:

0, 1, 2, 3, ..., n-1

Any question can be answered to remove only one possible candidate for min.
possible mins

\[ 1, 2, 3, \ldots, n-1, a_5 < a_7 \]

not possible min

pos min < pos min: Answer "yes"
move larger to not possible min

pos min < not pos min: Yes
not pos min < not pos min: be consistent w/ past
there's n elements initially
in possible mins

Each comparison moves ≤ 1
to not possible min.

How many times do we subtract
1 from n until only 1 left.

\[ n - 1 \]