Lower Bound Theory

- **Lower bound**: an estimate of a number of operations needed to solve a given problem

- **Tight Lower Bound**:  
  - There exists an algorithm with the same efficiency as the lower bound

- **Examples**:

<table>
<thead>
<tr>
<th>Problem</th>
<th>Lower bound</th>
<th>Tightness</th>
</tr>
</thead>
<tbody>
<tr>
<td>sorting (comparison-based)</td>
<td>$\Omega(n\log n)$</td>
<td>yes</td>
</tr>
<tr>
<td>searching in a sorted array</td>
<td>$\Omega(\log n)$</td>
<td>yes</td>
</tr>
<tr>
<td>n-digit integer multiplication</td>
<td>$\Omega(n)$</td>
<td>unknown</td>
</tr>
<tr>
<td>multiplication of n-by-n matrices</td>
<td>$\Omega(n^2)$</td>
<td>unknown</td>
</tr>
</tbody>
</table>

- **Methods of establishing lower bounds**:  
  - Trivial lower bounds  
    - Sorting  
  - Information-theoretic arguments (decision trees)  
    - Any comparison sorting algorithm (i.e., bubble sort)
- A convenient model of algorithms involving comparisons in which (Like sorting):
  - Internal nodes represent comparisons
  - Leaves represent outcomes

  **Adversary arguments:**
  - Merging two sorted lists
  - It’s a game between the adversary and the (unknown) algorithm.
  - The adversary has the input and the algorithm asks questions to the adversary about the input.
  - The adversary tries to make the algorithm work the hardest by adjusting the input (consistently).
  - It wins the “game” after the lower bound time (lower bound proven) if it is able to come up with two different inputs.
• **Searching in a sorted list:**
  o Objective: \(A(1) < A(2) < A(3) < \ldots < A(n)\)
  o Examples:
    - Comparison-based search algorithms
    - Search list by comparing target element with list elements
    - Sequential search: order \(n\)
    - Binary search: order \(\log_2 n\)

  o Comparison Tree:

    - Let us denote \(TB(n)\) be the worst case for best algorithm.
    - \(TB(n) = \text{highest of the best tree} = \text{Shortest highest of trees}\)
    - Each node will have \(n\) nodes
Lemma: A tree of nodes and height $h$, then
\[ h \geq \lceil \log (n + 1) \rceil - 1 \]

Then, $TB(n) \geq \lceil \log (n + 1) \rceil - 1$
Finding the Minimum and Maximum
  - Let’s consider the complexity of finding the largest and smallest elements. More formally, Given a sequence $X = \langle x_1, x_2, \ldots, x_n \rangle$ of $n$ distinct numbers, find indices $i$ and $j$ such that $x_i = \min(X)$ and $x_j = \max(X)$.

  - How many comparisons do we need to solve this problem?

    - An upper bound of $2n - 3$ is easy:
      - Find the minimum in $n - 1$ comparisons, and then find the maximum of everything else in $n - 2$ comparisons.
      - Similarly, a lower bound of $n - 1$ is easy, since any algorithm that finds the min and the max certainly finds the max.

    - We can improve both the upper and the lower bound to:
      $$\lceil \frac{3n}{2} \rceil - 2$$

    - The upper bound is established by the following algorithm:
      - Compare all $\lceil n/2 \rceil$ consecutive pairs of elements $x_{2i-1}$ and $x_{2i}$
      - Put the smaller element into a set $S$
• Put the larger element into a set $L$ and if $n$ is odd, put $x_n$ into both $L$ and $S$.
• Then find the smallest element of $S$ and the largest element of $L$.
• The total number of comparisons is at most:
  - $\left\lfloor \frac{n}{2} \right\rfloor$: Build $S$ and $L$
  - $\left\lceil \frac{n}{2} \right\rceil - 1$: Compute min $S$
  - $\left\lceil \frac{n}{2} \right\rceil - 1$: Compute max $L$
  - $\left\lfloor \frac{n}{2} \right\rfloor + \left\lfloor \frac{n}{2} \right\rfloor - 1 + \left\lfloor \frac{n}{2} \right\rfloor - 1 = \lceil \frac{3n}{2} \rceil - 2$
• Sorting
  o Decision tree for sorting 3 elements (Your textbook)

  o To find the lower bound, we have to find the smallest depth of a binary tree.
  o We have n! distinct permutations: n! leaf nodes in the binary decision tree.
  o The balanced tree has the smallest depth which the lowest bound for sorting:

  \[ \lceil \log(n!) \rceil = \Omega(n \log n) \]

  o Method 1:
• $\log(n!) = \log(n(n-1)\ldots1)$
  
  = $\log2 + \log3 + \ldots + \log n > \int_1^n \log x \, dx$

  
  = $\log e \int_1^n \ln x \, dx$

  = $\log e [x \ln x - x]_1^n$

  = $\log e(n \ln n - n + 1)$

  = $n \log n - n \log e + 1.44$

  $\geq n \log n - 1.44n$

  = $\Omega(n \log n)$

  • lower bound for sorting: $\Omega(n \log n)$

  o Method 2:

  • Using Sterling Approximation

  \[ n! \approx \sqrt{2\pi n} \left(\frac{n}{e}\right)^n \]

  \[ \log n! \approx \log \sqrt{2\pi} + \frac{1}{2} \log n + n \log \frac{n}{e} \]

  \[ \approx n \log n \approx \Omega(n \log n) \]

  • Merging two sorted lists:
o Merge two sorted sequences A and B with lengths m and n.

- **Binary decision tree:**
  
  There are \( \binom{m+n}{n} \) leaf nodes in the binary tree

- There are \( \binom{m+n}{n} \) ways!

- So the lower bound for merging:

  \[
  \left\lfloor \log\binom{m+n}{n} \right\rfloor \leq m + n - 1 \text{ (conventional merging)}
  \]

- When \( m = n \)

  \[
  \log\binom{m+n}{n} = \log\frac{(2n)!}{(n!)^2} = \log((2n)!) - 2\log n!
  \]

- Using Stirling approximation

  \[
  n! \approx \sqrt{2\pi n} \left(\frac{n}{e}\right)^n
  \]

  \[
  \log((2n)!) \approx 1/2\log 4p + 1/2\log n + 2\log(2n/e) = 1/2\log n + 2\log 2 + 2\log(n/e) + O(1)
  \]
\[ \log(n!) \approx \frac{1}{2}\log n + n\log(n/e) + O(1) \]

So,

\[
\log\left(\frac{m + n}{n}\right) \approx \frac{1}{2}\log n + 2n\log 2 + 2n\log(n/e) - \log n + 2n\log(n/e)
\]

\[= 2n\log 2 - \frac{1}{2}\log n + O(1) = \Omega(\ n)\]

Thus,

\[\log\left(\frac{m + n}{n}\right) \approx \Omega(\ n)\]