# 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**:

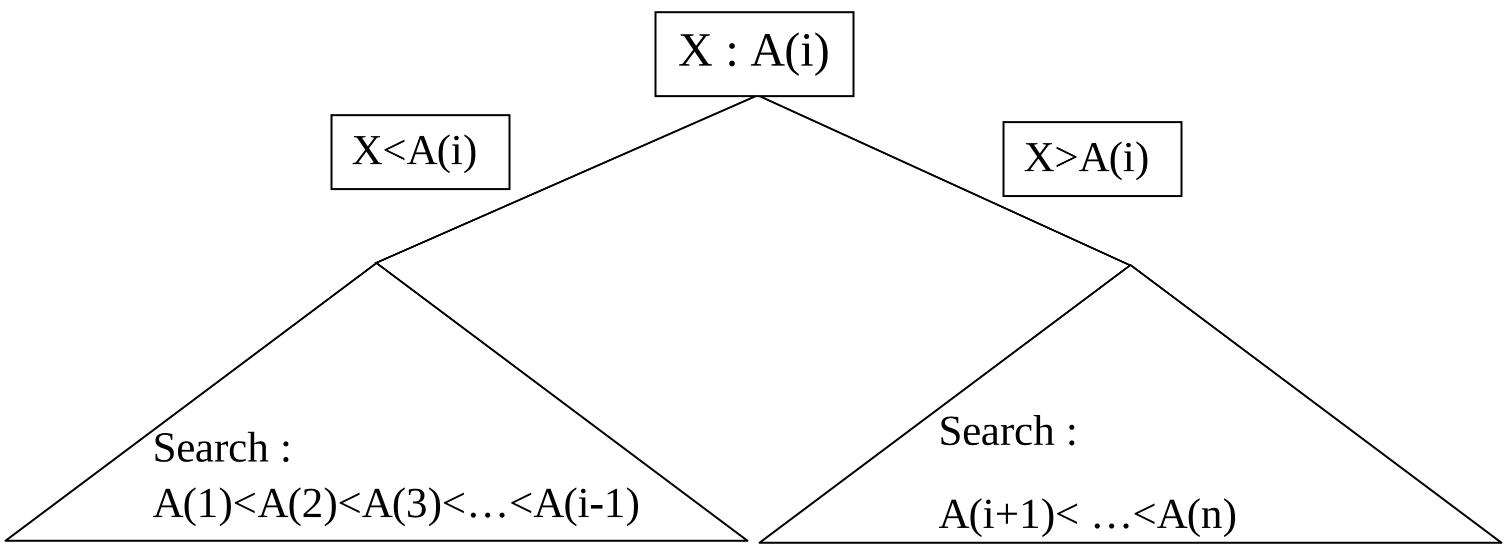
| Problem | Lower bound | Tightness |
|---|---|---|
| sorting (comparison-based) | $\Omega$ (nlog n) | yes |
| searching in a sorted array | $\Omega$ (log n) | yes |
| n-digit integer multiplication | $\Omega$ (n) | unknown |
| multiplication of n-by-n matrices | $\Omega$ ($n^2$) | unknown |

- **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:**
  - Objective: A(1) <A(2)<A(3)<…<A(n)
  - Examples:
    - Comparison-based search algorithms
    - Search list by comparing target element with list elements
    - Sequential search: order n
    - Binary search: order log2n

  - Comparison Tree:




    - Let us denote TB(n) be the worst case for best algorithm.
    - TB(n) = highest of the best tree = 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) ≥ $\lceil \log (n+1) \rceil - 1$

- **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.

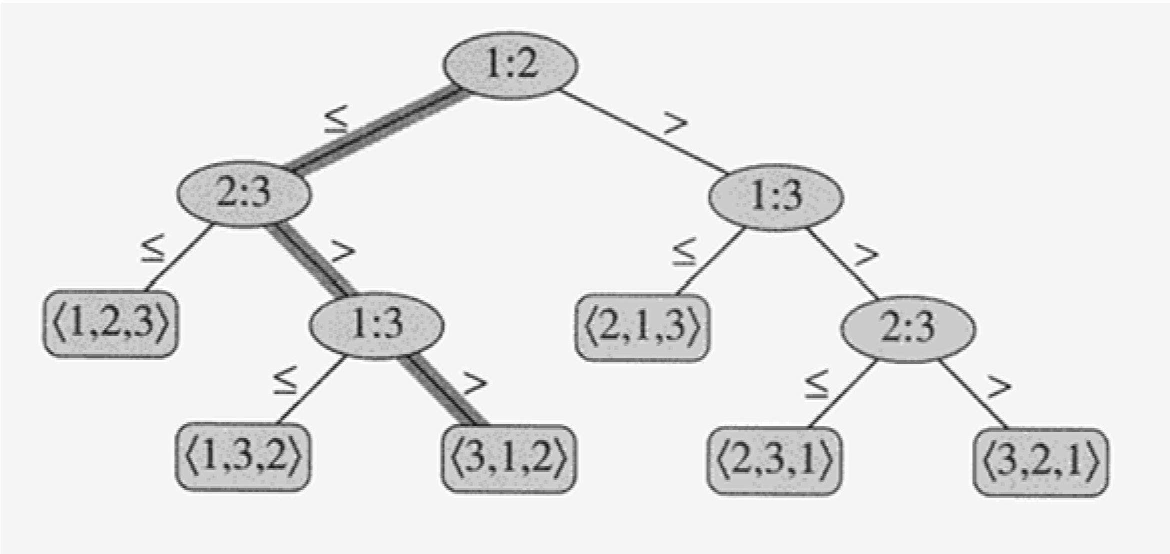
        - We **<u>can improve</u>** both the upper and the lower bound to:

$$\lceil 3n/2 \rceil - 2$$

        - 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:
  - $\left\lceil \boldsymbol{\frac{n}{2}} \right\rceil$**:** Build S and L
  - $\left\lceil \boldsymbol{\frac{n}{2}} \right\rceil$-1: Compute min S
  - $\left\lceil \boldsymbol{\frac{n}{2}} \right\rceil$-1: Compute max L
  - $\left\lceil \frac{n}{2} \right\rceil + \left\lceil \frac{n}{2} \right\rceil - 1 + \left\lceil \frac{n}{2} \right\rceil - 1 = \lceil 3n/2 \rceil - 2$

- **Sorting**
    - Decision tree for sorting 3 elements (Your textbook)



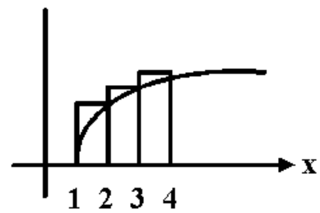

    - To find the lower bound, we have to find the smallest depth of a binary tree.
    - We have n! distinct permutations: n! leaf nodes in the binary decision tree.
    - The balanced tree has the smallest depth which the lowest bound for sorting:

$$\lceil \log(n!) \rceil = \Omega(n \log n)$$

    - **Method 1:**

- log(n!) = log(n(n–1)…1)

  = log2 + log3 +…+ log n > $\int_1^n logx\, dx$




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

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

  = log e(n ln n – n + 1)

  = n log n – n log e + 1.44

  ≥ n log n – 1.44n

  =Ω(n log n)

- <u>lower bound for sorting: Ω(n log n)</u>

- **Method 2:**
  - Using Sterling Approximation

    $n! \approx \sqrt{2\pi n}(\frac{n}{e})^n$

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

    ≈ n log n ≈ Ω(n log n)

- **Merging two sorted lists:**

- Merge two sorted sequences A and B with lengths m and n.
- <u>Binary decision tree:</u>

  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:

$$\left\lceil \log\binom{m+n}{n} \right\rceil \leq m + n - 1 \text{ (conventional merging)}$$

- When m = n

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

- Using Stirling approximation

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

$$\log((2n)!) \approx 1/2\log 4p + 1/2\log n + 2n\log(2n/e)$$

$$= 1/2\log n + 2n\log 2 + 2n\log(n/e) + O(1)$$

$\log(n!) \approx 1/2\log n + n\log(n/e) + O(1)$

So,

$$\log\binom{m+n}{n} \approx 1/2\log n + 2n\log 2 + 2n\log(n/e)$$

$$- \ \log n + 2n\log(n/e)$$

$$= 2n\log 2 - 1/2\log n + O(1) = \Omega( (n)$$

Thus,

$$\log\binom{m+n}{n} \approx \ \Omega( (n)$$