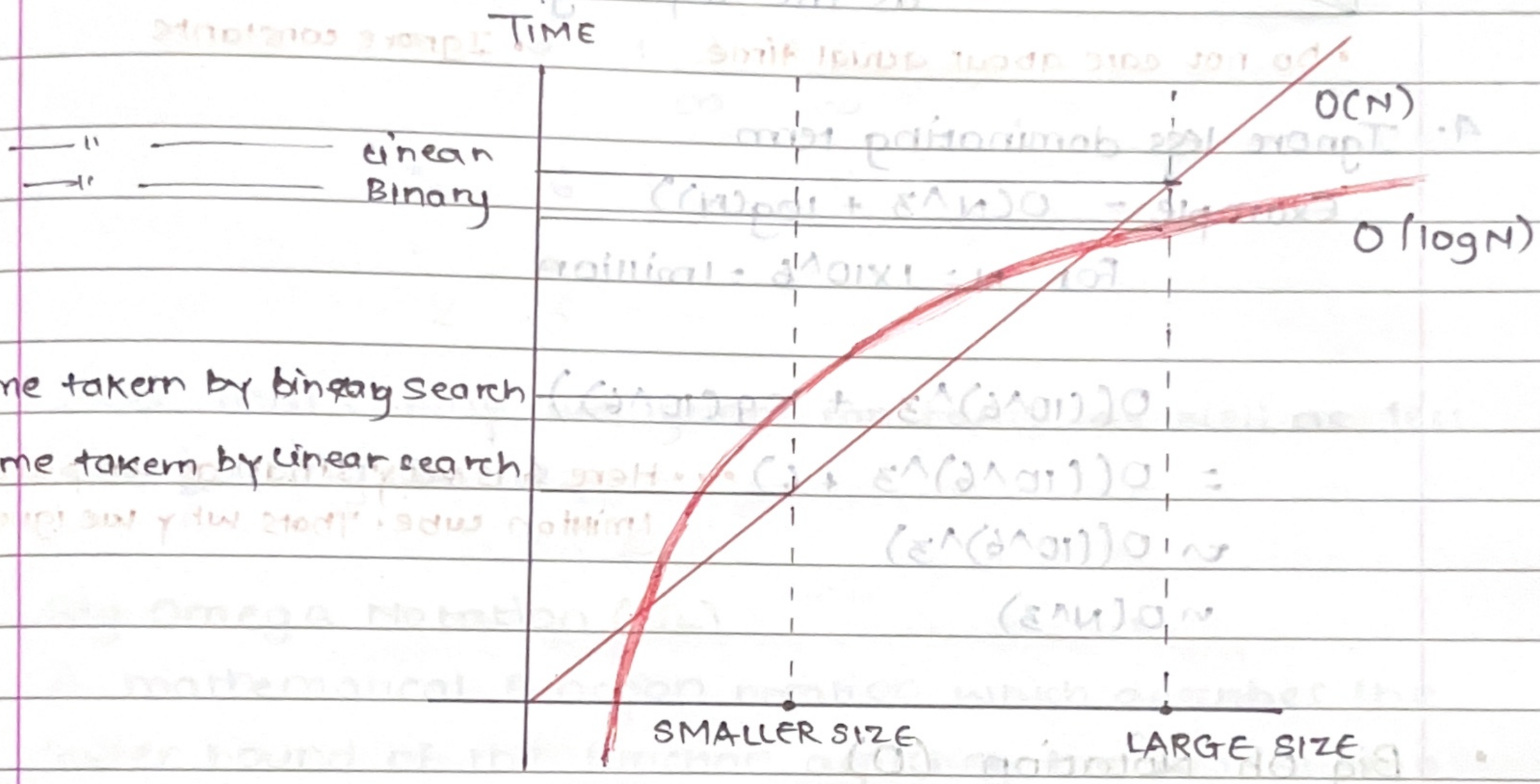
TIME AND SPACE COMPLEXITY

- TIME COMPLEXITY:-

It is a function which gives us the relationship about how the time will grow as the input sizes grows.

Observe the graph bellow:-

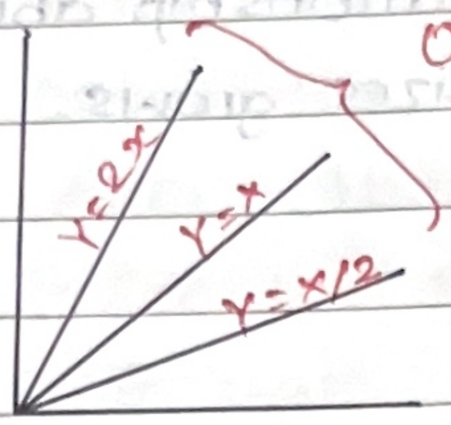


In the above example we are comparing linear search time complexity with the time complexity of binary search.

Observations

1. For smaller input sizes the time taken by linear search is less compared to the time taken by binary search
2. For larger input sizes, the time taken by linear search is more as compared to the time taken by binary search. And it keeps on increasing as compared to the time taken by binary search
3. We only consider the case having large number of input size i.e the worst case because it allows us to know whether the algorith is efficient in the long term or not. Worst case analysis helps us to determine real world scenario.
4. Thus we can say that binary search is more efficient compared to linear search.

Key points to consider when dealing with time complexity

1. Always look for worst case scenario.
2. Always take large input sizes/infinity into consideration.
3. 

Here, the actual time is different but in all cases the time is growing linearly. as the input grows
   - Do not care about actual time
   - Ignore constants
4. Ignore less dominating term

   Example - $O(N^3 + \log(N))$

   For $N = 1 \times 10^6 = 1$ million

   $O((10^6)^3 + \log(10^6))$

   $= O((10^6)^3 + 6)$ ... Here 6 is very small as compared to 1 million cube. Thats why we ignored it

   $\approx O((10^6)^3)$

   $\approx O(N^3)$

- Big Oh Notation (O)

  A mathematical notation which describes the upper limit of the function depicting the relationship between the time and input size.

  Example - $O(N^3)$

  The time complexity cannot exceed $O(N^3)$. It can be $O(N^2)$, $O(N^1)$, $O(\log N)$ but it cant be greater than $O(N^3)$.

  Mathematical part

  IF

  $f(n) = O(g(n))$

  then,

$$\lim_{n \to \infty} \frac{f(n)}{g(n)} < \infty$$

$$\lim_{n \to \infty} \frac{6(n^3) + 3n + 5 = O(n^3)}{n^3}$$

$$\lim_{n \to \infty} \frac{6 + \frac{3}{n^2} + \frac{5}{n^3}}{}$$

Substituting n = infinity

$$= \frac{6 + \frac{3}{\infty} + \frac{5}{\infty}}{}$$

$$= 6 + 0 + 0$$

$$= 6$$

$$< \infty$$

And that's why we ignore constants as well as less dominating terms.

- Big Omega Notation (Ω)

A mathematical function notation which describes the lower bound of the function depicting the relationship between the time and input size.

Example - $\Omega(N^3)$

The time complexity can exceed $\Omega(N^3)$, It can be N^4, $N^3 * 2^n$ etc, but it cant be lesser than $\Omega(N^3)$

Mathematical part

$$\lim_{n \to \infty} \frac{F(n)}{g(n)} > 0$$

- Theta Notation (Θ) - Combinging both

$$0 < \lim_{n \to \infty} \frac{F(n)}{g(n)} < \infty$$

- Little Oh notation (o)

Little o notation is used to describe an upper bound that cannot be tight. In other words, loose upper bound of f(n).

Comparison

Big Oh (O)	Little Oh (o)
$f = O(g)$	$f = o(g)$
$f \leq g$	$f < g$ (strictly slower)

Mathematical part

$\lim_{n \to \infty} \frac{f(n)}{g(n)} = 0$

$f = N^2$ $g = N^3$

$$\lim_{n \to \infty} \frac{N^2}{N^3}$$

$$\lim_{n \to \infty} \frac{1}{N} = 0$$

- Little omega notation (ω)

Little ω notation is used to describe a lower bound of f(n).

Comparison

	Big Omega (Ω)	Big Little Omega (ω)
	$f = \Omega(g)$	$f = \omega(g)$
	$f \geq g$	$f > g$

Mathematical part

$$\lim_{n \to \infty} \frac{f(n)}{g(n)} = \infty$$

- Space Complexity or Auxiliary Space?
  Auxiliary Space is the extra or temporary space used by an algorithm

Space complexity of an algorithm is total space taken by the algorith with respect to the input size. Space complexity includes both Auxiliary space and space used by input.

For example, If you want to compare standard sorting algorithm on the basis of space, then Auxiliary space would be a better criteria then space complexity. Merge sort uses $O(n)$ auxiliary space, Insertion sort and heap sort uses $O(1)$ auxiliary space Space complexity of all these sorting algorithm is $O(n)$ though.