worst, Average and Best case Time complexities

after writing a code we must analyze it can to check it's efficiency in terms of time and space.
So, that if required we can Improved it further.

analysis it done by asymptotic analysis. the best way to do it.

after analysis it $\overset{has}{\text{give}}$ three cases :-
1. worst case
2. Averge case
3. Best case

Let's take an Eg :- of linear search :-

1. worst case :- here, we calculate upper bound on running time of an algo.
it'll happen when the element x (which we are finding in Array) is not present.

2. Average case :-
here, we sum all possible values and Devides it with no. of inputs. (n+1)

$$\frac{\theta((n+1)^{*}(n+2/_{2}))}{(n+1)} = \theta(n)$$

3. Best case Analysis
we calculate lower bound on running time of an algorithm.
in linear search
The best case occurs when thee X is present in the first Indexed Itself.

no. need to check further. we got our element in first place only.

Time complexity (Big O-notation)

- Best case: O(1) :- this will take place if the element is present in first Indexed. no. of comparision in this case is 1.

- Average case :- O(n), This will take place if X is present on the middle of Array.

- worst case : O(n) : this will take place
  (i) when we get the element at the last Index
  (ii) when the element is not peresent.

Importan note :-
(i) we do worst case in most cases because definetly we found a upper bound on the running time (good piece of info)

(ii) avegage case :- we do sometime it not easy
(iii) best case :- here we will get lower bound but it doesn't provides any info.