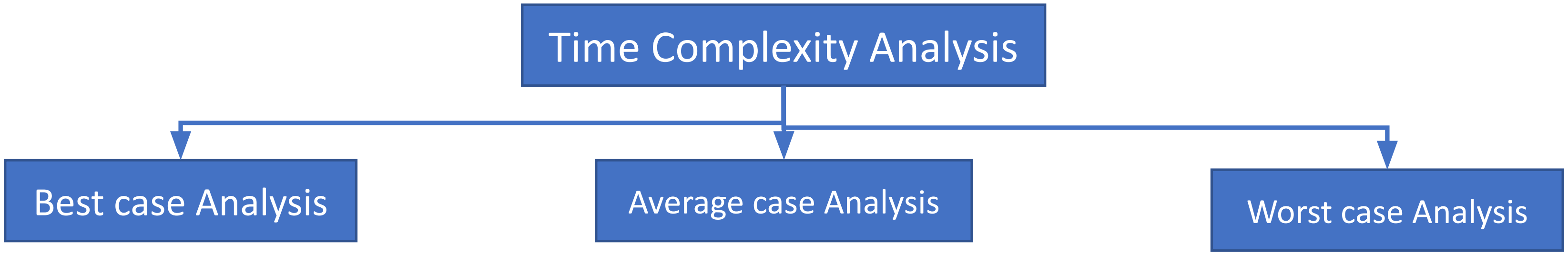
Outline : Lecture 2

- Different cases of Time Complexities
  - ✔ Best case
  - ✔ Average case
  - ✔ Worst case
  - ✔ Example
- Asymptotic Notations (O, o, Ω, ω, ϴ)
  - ✔ Why do we study asymptotic notations?
  - ✔ Big oh (O)
  - ✔ Small oh (o)
  - ✔ Big omega (Ω)
  - ✔ Small omega (ω)
  - ✔ Theta (ϴ)

Different cases of Time Complexities Analysis:

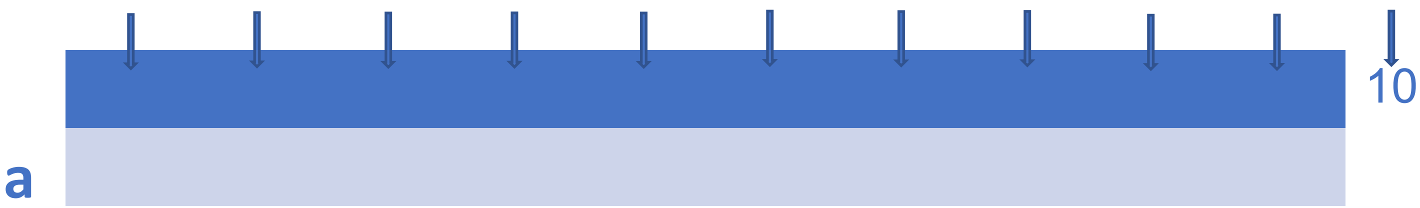


- Best case Analysis: It defines the input for which an algorithm will take minimum time to execute.
- Worst case Analysis: It defines the input for which an algorithm will take maximum time to execute.
- Average case Analysis: It is usually harder to analyze the average behaviour of an algorithm than to analyze the behaviour in the worst case. In this case, we assume that all inputs of a given size are equally likely and do the probabilistic analysis for the average case.

Example to understand the best case, worst case and average case analysis of an Algorithm:

- Write an algorithm to search whether a given element present in the given list of n given elements or not.

Here, $n = 10$



Searching Element (x)	No. of comparison	Array Index value (i)	Case Analysis	Conclusion in terms of n
$x = 10$	1	$i = 0$	*Best Case*	*Constant*
$x = 59$	10	$i = 9$	*Worst Case*	n
$x = 69$	11	$i = 10$	*Worst Case*	$n+1$

Linear Search Algorithm:

```
int search ( int x, int n)
{
   int i, flag=0;
   for (i=0; ((i<n) && (!flag)); i++)
    {
       if (x==a[i])
        {
            flag=1;
            break;
         }
    }
  if(flag)
   return(i);
  else
   return (-1);  // Invalid position to indicate unsuccessful search
}
```

- **Here, in this function, we have used a flag variable to indicate the status of searching operation at the end.**
- **The initial value of flag=0 indicate unsuccessful search.**

```
int search ( int x, int n)
{
   int i;
   for (i=0; ((i<n) && (x!=a[i]));   i++);
 if(i<n)
   return(i);
  else
  return (-1);
}
```

- **Here, in this function, we have used the value of array index i to indicate the status of searching operation at the end.**

Worst case time Complexity of Linear Search Algorithm:

```
int search ( int x, int n)
{
   int i;
   for ( i=0; ((i<n) && (x!=a[i])); i++ );
  if(i<n)
     return(i);
   else
   return (-1);
}
```

Unit Time

1

2x(n+1)

n

1

1

$$T(n) = 1 + 2 * (n + 1) + n + 1 + 1 = 3n + 5$$

Best case time Complexity of Linear Search Algorithm:

$T(n) = 5$ (Constant time which is independent of the problem input size n)

Average case time Complexity of Linear Search Algorithm:

$$A(n) = 1 * \frac{1}{n} + 2 * \frac{1}{n} + 3 * \frac{1}{n} + \cdots + \mathrm{n} * \frac{1}{n}$$
$$= \sum_{i=1}^{n} i * \frac{1}{n} = \frac{1}{n} * \sum_{i=1}^{n} i$$
$$= \frac{1}{n} * (1 + 2 + 3 + \cdots + n)$$
$$= \frac{(n+1)}{2}$$ [Here, we have two assumptions]

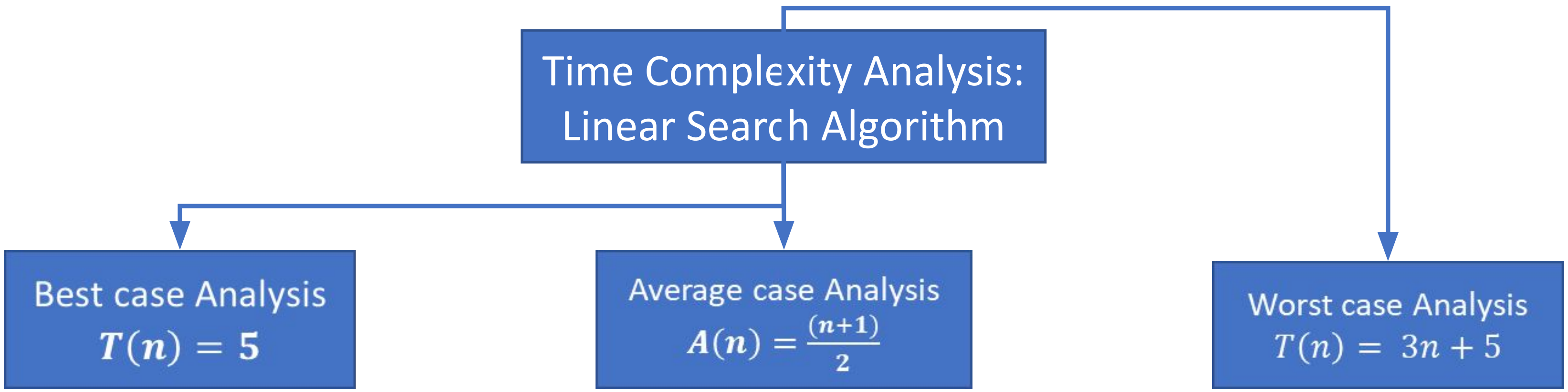
- If we are not sure that whether the searching element present in the given list or not, then the probability of the searching element may be present at a particular position is $\frac{p}{n}$, where p is the probability that the given element is present in the given list.
- p : represents the probability of successful search
- (p-1): represents the probability of unsuccessful search

Average case time Complexity of Linear Search Algorithm: (Cont..)

- $$A(n) = 1 * \frac{p}{n} + 2 * \frac{p}{n} + 3 * \frac{p}{n} + \cdots + \mathrm{n} * \frac{p}{n}$$
$$= \sum_{i=1}^{n} i * \frac{p}{n} = \frac{p}{n} * \sum_{i=1}^{n} i$$
$$= \frac{p}{n} * (1 + 2 + 3 + \cdots + n) = \frac{p*(n+1)}{2}$$

- **All the above two different cases of the average case time complexity of linear search algorithm, we made an assumption.**
- **The question is that what assumption, we have considered for the above two cases?**
- **The assumption was that the probability of a given element may be present in the given list is equable probable (no bias factor)**
- **When we don't consider the equable probable concept, then**
- $\boldsymbol{A(n) = 1 * p_1 + 2 * p_2 + \cdots + n * p_n,\quad where \sum_{i=1}^{n} p_i = p}$ **represents the probability of successful search and each $p_i$represents the probability that the searching element may be present at the i^{th} position in the list.**

Different cases of Time Complexities of Linear Search Algorithm:



- Most of the time, when we analyze the time complexity of an algorithm, we are concerned with the worst case time complexity

Asymptotic Notations (O, o, Ω, ω, ϴ):

Why study Asymptotic Notations ?

Dipak Sourav Ranjit Dipak, Ranjit, and Sourav

Who is the tallest among these three?

Why study Asymptotic Notations ? (Cont..)

Who is the tallest among these three?

Calculating the individual height, we can get the answer. (Using Exact individual computation)

Without Calculating the exact individual height, we can get the answer using approximation. (Using the idea of individual solution)

- This is very time consuming process with respect to other (More time is required to get the answer)

- This is very easy process with respect to other (less time is required to get the answer)

- The asymptotic notations give us the **idea** about the time complexity of an algorithm.
- With the help of this beautiful idea, we find the best (efficient) algorithm

Asymptotic Notations (O, o, Ω, ω, ϴ): (Cont..)

To understand the Asymptotic notations, we must understand the following:

- Upper bound of a function
- Loose upper bound of a function
- Lower bound of a function
- Loose lower bound of a function

Let $f(n) = \left\{ \frac{1}{n} : n \in Z \right\} = \{ 1, \frac{1}{2}, \frac{1}{3}, \frac{1}{4}, \dots \}$

If $u \geq x, \forall x \in f(n)$, then u is called an upper bound of $f(n)$.

- Here $u = 1$ or $u = 2$, $u = 3$, $u = 4$ ....etc are the upper bounds of the $f(n)$
- Here $u = 1$ is called tight upper bound and $\{2,3,4, \dots\}$ are called loose upper bounds of the $f(n)$.
- It means that an upper bound may be tight or may be loose

If $l \leq x, \forall x \in f(n)$, then l is called an lower bound of $f(n)$.

- Here l =0 or $l = -1$, $u = -2$, $u = -3$ ....etc are the lower bounds of the $f(n)$
- Here $l = 0$ is called tight lower bound and $\{-1, -2, -3, -4, \dots\}$ are called loose lower bounds of the $f(n)$.
- It means that an lower bound may be tight or may be loose

Understanding the upper & Lower Bounds Concepts

- Who is this person?
- What is his height?
- Whatever statement will be made by you about this question, that must be true.
- The actual height of Abhinandan Varthamanan is 5’ 7”.
- The possible following statement are:
- *His height is more than 4’*
- *His height is more than 3’*
- *His height is more than 5’2”*
- *His height is less than 7’*
- *His height is less than 6’5”*
- So many possible answer (correct statement) may come
- $\{\mathbf{5'7''}, 6'5'', \mathbf{7'}, \ldots\}$: *Set of upper bounds*
- $\{6'5'', 7', \ldots\}$: *Set of **loose** upper bounds*
- $5'7''$ *is considered as a **tight** upper bound*
- $\{\mathbf{5'7''}, 5'2'', \mathbf{4'}, \mathbf{3'} \ldots\}$: *Set of lower bounds*
- $\{5'2'', 4', 3' \ldots\}$: *Set of **loose** lower bounds*
- $5'7''$ *is considered as a **tight** lower bound*
- $5'7''$ *is also called as a **tight bound***

Abhinandan Varthamanan,

Fighter Pilot,

Indian Air Force

Asymptotic Notations (O, o, Ω, ω, ϴ): (Cont..)

Big Oh (O): The big oh of a function gives us the idea of the **asymptotic upper bound** of a function.

Small oh(o): The small oh of a function gives us the idea of the **asymptotic loose upper bound** of a function.

Big Omega (Ω): The big omega of a function gives us the idea of the **asymptotic** lower bound of a function.

Small omega (ω): The small omega of a function gives us the idea of the **asymptotic** loose lower bound of a function.

Theta (ϴ): The theta of a function gives us the idea the **asymptotic** tight bound of a function.

Asymptotic Notations (O, o, Ω, ω, Θ): (Cont..)

What does asymptotic mean?

- The definition of asymptotic is a line that approaches a curve but never intersect (never crossing).
- *A curve and a line that get closer but do not intersect are examples of a curve and a line that are asymptotic to each other.*