(Practical 1)

Name :- Deshmukh Swapnil Shankarrao
Reg no :- 2020BCS029
Roll no :- A14 (AA)

Aim :- To study Asymptotic Notations.

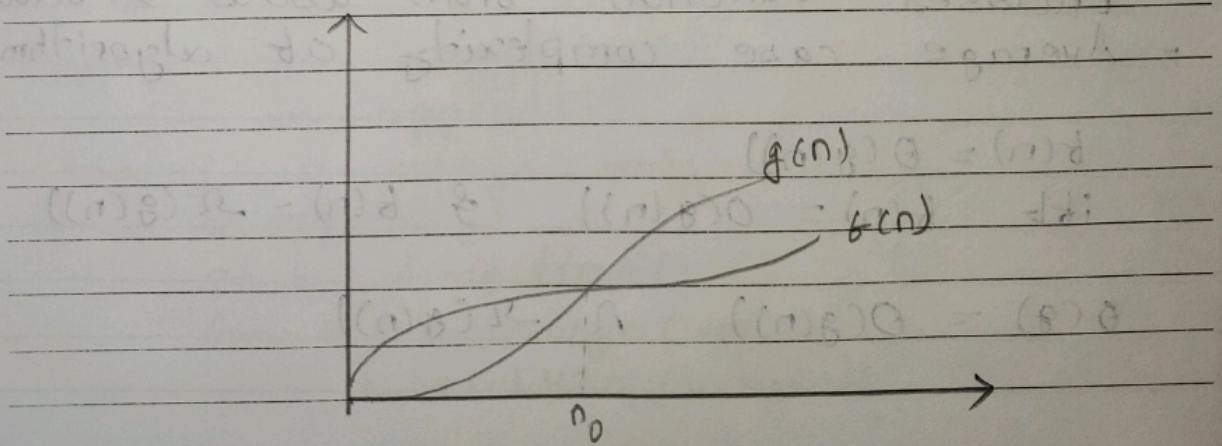
Theory :- Asymptotic notations are the mathematical notations used to describe running time of algorithm, when input tend towards particular value.

There are mainly 3 asymptotic notations :-

1) Big O notation :

Big O represents upper bound of running time of algorithm.

$f(n) = O(g(n))$
$f(n) \leq c \cdot g(n)$ for all $n \geq n_0$

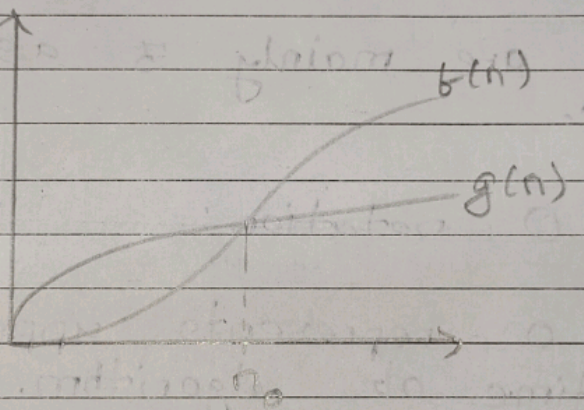


( )

2> Omega notation (Ω)

- Denotes lower bound of running time of algorithm.
- Provides best case complexity

$f(n) = \Omega(g(n))$
if there exists constants c & n_0 such that,
$f(n) \geq c \cdot g(n)$ for all $n \geq n_0$

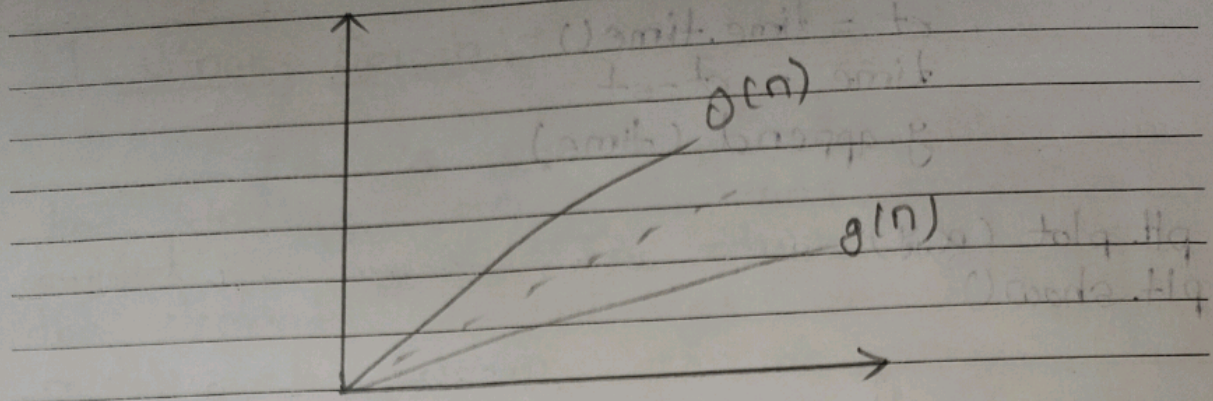


3> Theta notation (θ)

- Encloses function from above & below
- Average case complexity of algorithm.

$f(n) = \theta(g(n))$
iff $f(n) = O(g(n))$ & $f(n) = \Omega(g(n))$

$\theta(g) = O(g(n)) \cap \Omega(g(n))$



1] Linear Search :-

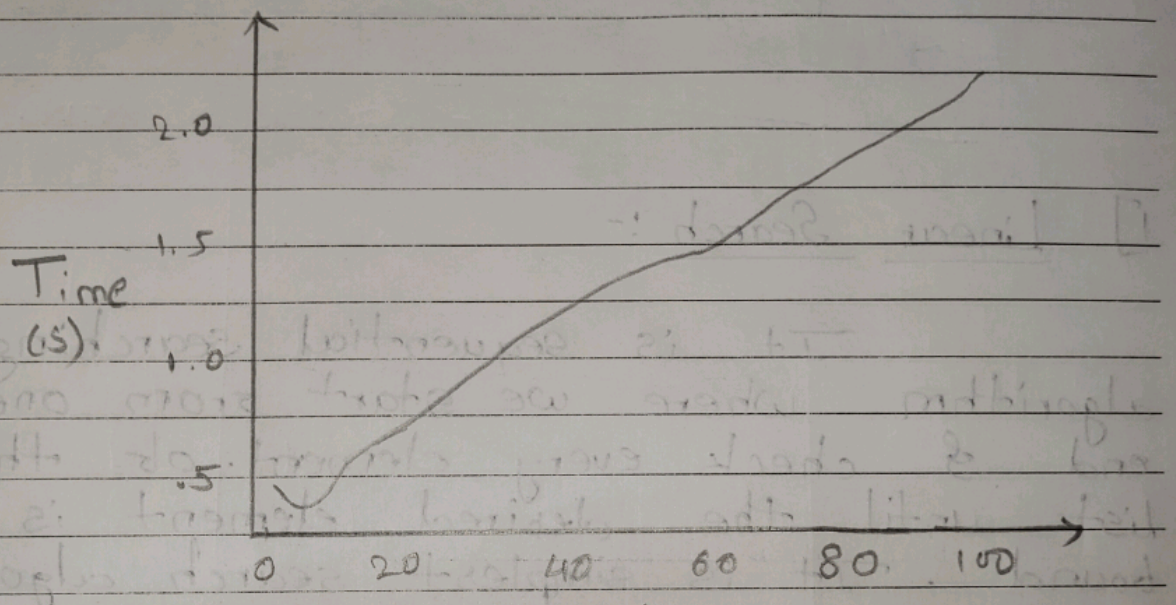
It is sequential searching algorithm where we start from one end & check every element of the list until the desired element is found. It is simplest search algorithm.

Program :- (Python)

```
arr = []
x = []
y = []
for i in range(1000):
        arr.append(i)
for j in range(100):
        x.append(j)
        st = time.time()
        for k in range(1000):
                if arr[k] = j: break
```

```
et = time.time()
time = et-st
y.append (time)

plt.plot (x,y)
plt.show()
```



Time complexity = O(n)
Space complexity = O(1)

2] Binary Search :-

It is searching algorithms for finding an elements po's in sorted array. In this approach, the element is always searched in middle of portion of an array.

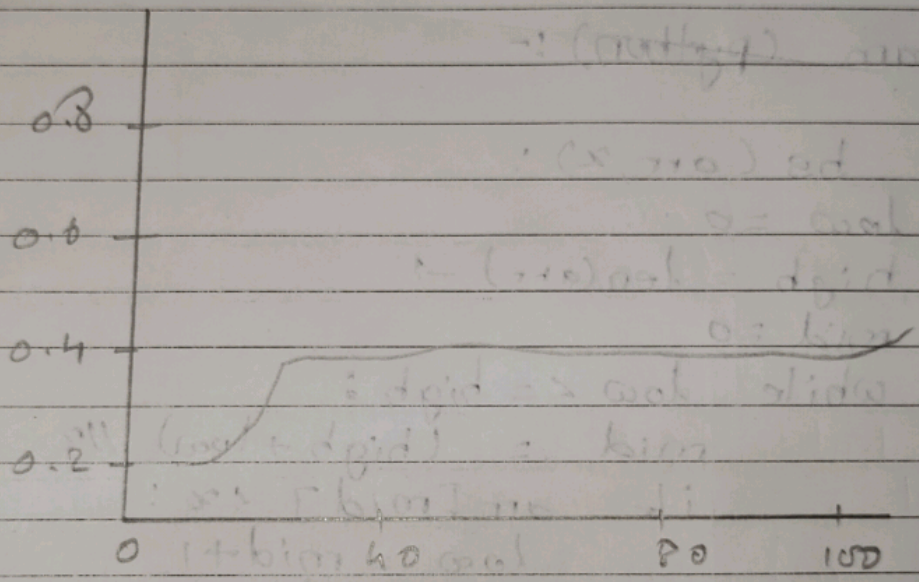
Program (python) :-

```
def bs(arr, x):
    low = 0
    high = len(arr) - 1
    mid = 0
    while low <= high:
        mid = (high + low) // 2
        if arr[mid] < x:
            low = mid + 1
        elif arr[mid] > x:
            high = mid - 1
        else: return mid
    return -1

arr = []
x = []
y = []
for i in range(1000): arr.append(i)
for j in range(100):
    x.append(j)
    st = time.time()
```

```
        bs(arr,j)
        et = time.time()
        time = et-st
        y.append(time)
plt.plot(x,y)
plt.xlabel('Input size')
plt.show()
```

O/P :-



Time complexity = $O(\log n)$

Space complexity = $O(1)$

3] Bubble Sort :-

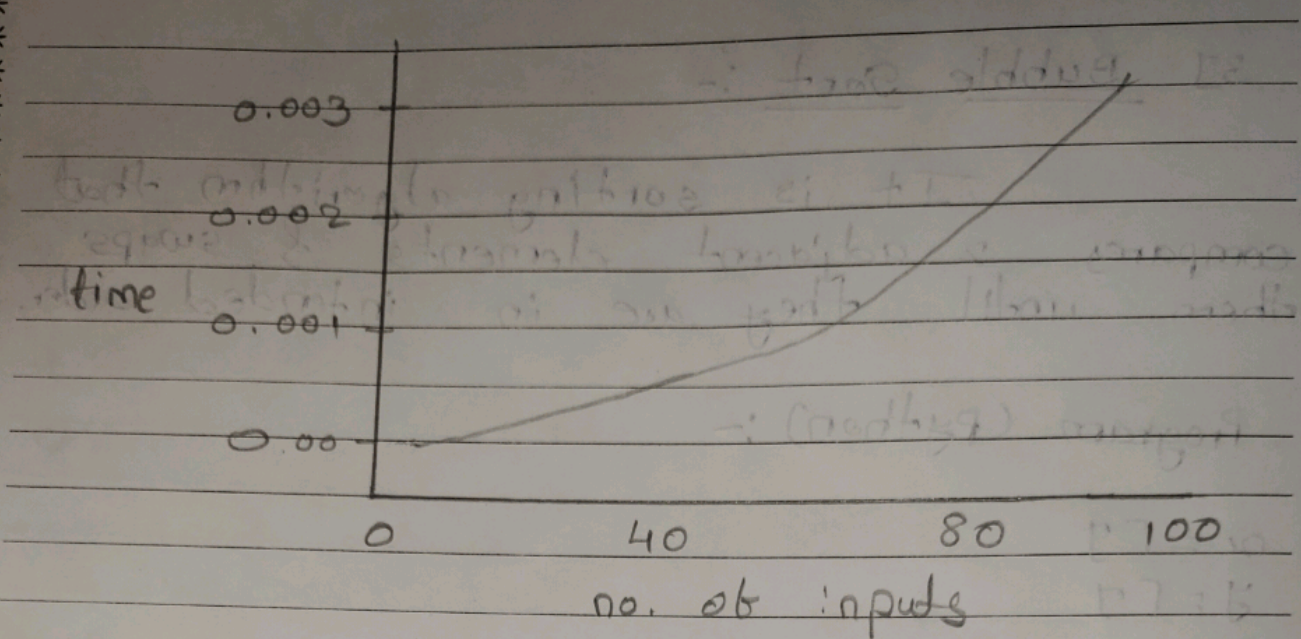
It is sorting algorithm that compares 2 adjacent elements & swaps them until they are in intended order.

Program (Python) :-

```
x = []
y = []

for k in range(100):
    x.append(k)
    arr = []
    for i in range(k):
        arr.append(100-i)
    st = time.time()
    for i in range(n):
        for j in range(0, n-i-1):
            if arr[j] > arr[j+1]:
                arr[j] = arr[j+1]
                arr[j+1] = arr[j]
    et = time.time()
    time = et-st
    y.append(time)

plt.plot(x,y)
plt.show()
```



Time complexity = $O(n^2)$
Space complexity = $O(1)$

Conclusion :- Hence, we have studied asymptotic notation with 3 diff algorithms with their time complexities.