UITS

Future will be better than thy past

An initiative of PHP

University of Information Technology & Sciences

DSA-1 Home Task

Course Title : Data Structures and Algorithms Lab

Course Code : CSE0613212

Topic : Asymptotic Big-O notation short explanations

Submitted to:

Teacher's Name : Saima Siddique Tashfia (SST)

Designation : Lecturer

Submitted by:

Student Name : Gaus Saraf Murady

Student ID : 0432410005101088

Semester : Spring, 2025

Batch : 55

Department : CSE

Date of Submission: 11.02.25

Home Task on Asymptotic Big-O notation shortly explained.

The following were discussed in the Lab :

$O(1)$ = constant time complexity

$O(N)$ = linear time complexity

$O(\sqrt{N})$ = squareroot time complexity (?)

$O(N^2)$ = Polynomial time complexity

$O(\log n)$ = Logarithmic time complexity

$O(n \log n)$ = Logarithmic " "

$O(2^n)$ = Exponential " "

which will be shortly explained here on,

Example of $O(1)$:

simple operations like value assigning and accessing array elements always takes constant amount of time, hence the $O(1)$ notation.

```
int a = 1; int b = 2;
int c = {1,2,3,4};
printf("%d", c[0]);
```

#Example of O(N): time of ~~code~~ algorithm running increases as time goes on or grows linearly.

```
for(int i = 1; i<=n; i++)   ---> checks n+1 times
{  //statements   ----------> checks n times
}
```

here, time complexity = $O(n+1+n)$

$= O(2n+1)$

$= O(n)$; constants are negligible

#Example of $O(\sqrt{N})$:

Algorithms whose logic divides the input value into smaller parts so the code runs for a smaller time than when run for n.

```
void func(int n){ printf("Division of %d", n);
    for(int i=1; i*i <=n ; i++){
     if (n%i==0){ printf("%d", i);
       if ( i!=n/i){ printf("%d", n/i); }}}
       printf("\n"); }
int main (){ int n =36; func(&n);
            return 0;}
```

Example of $O(2^n)$:

running time of an algorithm doubles with an algorithm doubles with each addition to the input data set.

```
void func(int arr[], int n)
{ for (int i = 0; i < (1<<n); i++) {
      for (int j = 0; j<n; j++){
        if (i & (1<<j)){
          cout << arr[j] << " ";
        } cout << endl;
      }}
```

Example of $O(N^2)$:

Algorithms like Bubb sort through each of the n values each cycle for n total cycles.

Meaning;

```
n = len(my_array)
for i in range(n-1):
    for j in range(n-i-1):
      if my_array[j] > my_array[j+1]:
          my_array[j], my_array[j+1] = my_array[j+1],
          my_array[j]
                          print("Sorted array:", my_array)
```

#Example of O(log n):

when each step is the multiple of the previous step in an algorithm.

```
for(i=1; i<=n; i*2){ }
for (i=256; i>=1; i=i/2){ }
```

Time complexity: for $\frac{N}{2^K} \rightarrow 1$

$$N = 2^K$$

$$\log_2 N = \log_2 2^K = K;$$ being the multiple

$$\therefore O(\log N)$$

#Example of O(n log n):

Algorithms like merge sort where an array is divided in half and processed each half independently. It is also used to optimize $O(n^2)$ algorithms.