28/01/2026

DSA in Python

Data Structure & Algorithm

Data ↓ information

Structure ↓ Store → Array, Linked list, Stack, Queue, Tree, Graph

Algorithm ↓ Set of Algo instruction

Time Complexity

Comparison

$1 < \log \log n < \log n < n < n^2 < n^3 < n^4 \ldots 2^n$

n = size of input, if n is very large

Equation

1) Constant equation

$y = 5$

independent of x

x = input

y = output

2) Lineare $y = 2x + 5$

3) Quadratic $y = 2x^2 + 3x - 5$

4) Cubic $y = 3x^3 + 2x^2 + 10$

5) logrithmic $y = \log x + 2$

6) Exponential $y = 3^x + 3$

Most Dominating factor in these above equation is

x

x^2

x^3

$\log x$

3^x

Asymptotic Notation

We use kg, km, litre as unit that define the content. Similarly we use notation to measure time complexity.

Analysis :

1) Best case : Ω → omega Notation
2) Average case : θ → Theta Notation
3) Worst case : O → Big-O-Notation

$\Omega(n)$, $\Theta(n \log n)$, $O(n^2)$

Constant time :- $O(1)$

```
x = 5          → 1 unit  } 2 unit
Print (x)      → 1 unit  }
                                    } O(1)
if (x == 5):             }
    Print ("Hello")      } constant
else:                    }
    Print ("Welcome")    }
```

Linear Time :

$y = 2n + 10 \Rightarrow O(n)$

```
x = 5                                  } O(1) constant time
for i in range(0,5):  ----> n unit
    Print(i)          ----> n unit
```

$1 + n + n$

$= 2n + 1 \Rightarrow O(n)$

Another example -

```
n = 5                          } ----> O(1)
Print(n)                               +
for i in range(0,n):           } ----> O(n)        O(n)
    Print(i)                           +
for j in range(0,n):           } ----> O(n)
    Print(j * j)
for k in range(0,n/2):         } ----> O(n/2) } O(n)
    Print("Hello")
```

Quadratic Time complexity

$2n^2 + 3n + 5 \approx O(n^2)$

```
n = 5              ----> 1 unit
m = 10             ----> 1 unit
Print(n+m)         ----> 1 unit
for i in range(0,n):   ----> n times = 5 times
    for j in range(0,m): ----> n time = 10
        Print("Hello")     ----> 50 times
for k in range(0,n):
    Print("welcome")
```

$1 + 1 + 1 + n \times n$

$= \underbrace{n^2 + 3 + n}$

$O(n^2)$

Cubic Time Complexity

```
n = 5                  -> 1   }
m = 10                 -> 1   }  O(1)
Print(n+m)             -> 1   }
for i in range(0,n):   -> n        }
  for j in range(0,m): -> m        }  O(n^3)
for k in range(0,n):   -> n        }
  Print("Welcome")     -> n x m x n }

For a in range(0,n): } -> O(n)
  Print("Hello");
```

```
n = 5
if (n == 5):
   [O(n^2)]            }
else:                  }  O(n^3)
   [O(n^3)]            }
```

Logarithmic time complexity

```
n = 10                    n=10 , n=5 , n=2 , n=1
While (n >= 1):        }  hello  Hello  Hello  Hello
  Print("Hello")       }  k times = O(k) =
  n = n//2             }             O(log n)
```

$\frac{n}{2}, \frac{n}{2^2}, \frac{n}{2^3}, \ldots \frac{n}{2^k} = 1$

$n = 2^k$

$\log n = \log 2^k$

$\log n = k \log 2$

$k = \frac{\log n}{\log 2}$

Another example

$10 \to 5 \to 2 \to 1$

$1 \to 2 \to 4 \to 8$

```
n = 1
while (i < 10):
    Print ("Hello")
    i = i * 2
```

i=1, i=2, i=4, i=8

Hello Hello Hello Hello

$O(\log n)$

Question :-

```
n = 10
Print(n)                    } O(1)
for i in range (0,n):       } O(n)
    Print ("Hello")

for j in range (0,n):
    for k in range (0,n):   } O(n^2)
        Print ("Welcome")
while (n >= 1):
    Print ("Agastya")       } O(log n)
    n = n//2
```

$O(n^2)$

$O(1) < O(\log n) < O(n) < O(n^2)$