Execution Time ? ⇒ Time taken for the program to run some input data set

Anjali	Nitesh
15 sec (Mac)	20 sec (Windows XP)
15 sec (Mac)	10 sec (Mac)
python	c++
8 sec (c++)	10 sec (c++)
8 sec (-)	7 sec (Ac)

python < C++

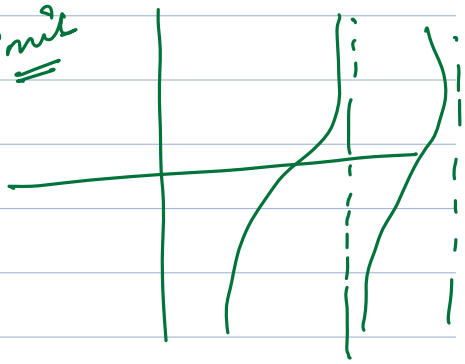
Execution Time ✗

↓

iterations

Anjali	Nitesh
$10 \log_2 N$	$N/10$

for large inputs ⇒ Asymptotic Analysis



worst-case ← Big-O Notation

↳ upper-bound

as close as possible to actual value

$a = 5$

$a <= 7$

(#) calculate number of iteration's in the terms of input

(#) Neglect lower order terms

(#) Neglect constant coefficients

for large inputs

$10N^2 + 100N + 10$

$N^2 > 100N$ $N^2 > 10$

$10N^2$ ↳ rate of growth

$O(N^2)$

N & $N/2$

N & $\sqrt{N}$

$$F(N) = 4N^2 + 3N + 1$$

$$\approx O(N^2)$$

$$F(N) = 4N + 3N\log_2 N + 1$$

$N < N\log N$ — depends on input

$N = 10^6$

$\log_2 N \approx 20$

$10^6 \quad 20 * 10^6$

$$O(N \log_2 N)$$

$N < N \log N$

$N * 1 < N * \log N$

$N \log_2 N \rightarrow T = 1$ Logn inputs

$> N$

$\log 10^6 = 20$

$\log 10^9 \approx 30$

$\log 10^{18} = 64$

$$F(N) = 4N\log N + 3N\sqrt{N} + 1$$

N	$64 * N$
N^2	$N \log N$
$N * N$	$N * \log N$
10^{18}	64

$N \log N$	
$N = 10^6$	20
$N = 10^9$	30
$N = 10^{18}$	64

$N\sqrt{N}$
10^3
$10^{4.5}$
10^9

$N > \log_2 N$

$\log_2 N > 1$

$$O(F(N)) = O(N\sqrt{N})$$

$F(N) = 4$ constant

$O(1) \rightarrow$ const

$F(N) = 4N + 3$

$\approx O(N)$ ≈ linear

$O(N^2) \approx$ quadratic

$\log N = \log_2 N$

$O(1) < O(\log_2 N) < O(\sqrt{N}) < O(N) < O(N \log_2 N) < O(N\sqrt{N}) < O(N^2) < O(N^3) < O(2^N)$

$N^2 / N^3 \rightarrow N\sqrt{N}$

$N \log N$

N

$\sqrt{N}$

$\log_2 N$

$O(1)$

10:25

$N^2 + 10N$

$N = 100$ $\quad (100)^2 + 10 * 100$

$\frac{10N}{N^2+10N}_{*100} = \frac{10^3}{10^4+10^2} \approx 10\%$

$N = 10^5$ $\quad 10^{10} + 10 * 10^5$ $\quad * \quad \frac{10^6}{10^{10}} \approx 10^{-4} \in 0.01\%$

Logic

$10N^2 + 4N + 5$ $\quad \downarrow$ $\quad O(N^2)$

$7N^2 + 8N + 6$ $\quad \downarrow$ $\quad O(N^2)$

Space complexity

No of use

```
void fun( int N)
{
    int x;          // 4 bytes
    double y;       // 8 bytes
    float z;        // 4 bytes
    bool a;         // 1 byte
}
```

17 bytes

Some space

O(1) space complexity

```
void fun( int N) {
    int arr[10];    // 4*10 = 40 bytes
    float f;        // 4 bytes
    int arr2[N];    // 4N bytes
}                   // 4 bytes
```

4N bytes

4 bytes

4*N + 44

O(N)

input

```
void fun( int arr[N], int N)
{
    int sum=0;
    for( int i=0 → n)
        sum+=arr[i];
}
```

O(1)

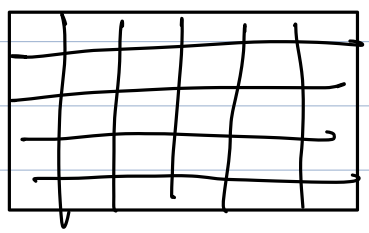
O(N)

input data storge → we generally don't take it as extra space

$O(N^2)$

int arr2[N][N];

└ 2D Matrix



```
sum=0;
for( int i=0; i<n; i++)
    sum+=arr[i];
```

O(N) → Time

Best case
worst case

search an element k in array

```
for(int i=0, i<n; i++)
{
    if(arr[i]==k)
        return true;
}
```

best case ≡ O(1)

worst case ≡ O(N)