Agenda

- Arithmatic and Geometric Progression
- Time Complexity : Speed of Execution
  - Big O Notation
  - Comparison of Order
- Space Complexity : Memory

Arithmatic and Geometric Progression

Concept 1 Logarithmic iterations

$$N \rightarrow N/2 \rightarrow N/4 \; \cdots\cdots\cdots\cdots \; 1$$

$S \qquad 1 \qquad 2 \qquad \cdots \qquad k \rightarrow \log_2 N$

Q How many steps will it take to reach 1 ?

$$N \rightarrow N/2 \rightarrow N/4 \; \cdots\cdots\cdots\cdots \; 1$$

$$N \rightarrow N/2 \rightarrow N/2^2 \rightarrow N/2^3 \cdots\cdots N/2^k$$

$$N/2^k \Rightarrow 1$$

$$N = 2^k \Rightarrow \log_2(N) = \log_2(2^k$$

$$\boxed{\log_x x = 1}$$

$$\log_2 N = K \log_2 (2)$$

$$K = \log_2 N$$

Ex: 8, 4, 2, 1

Concept 2

$[a, b] = (b - a + 1) \rightarrow K$

Q: How many numbers are between 'a' and 'b'? (inclusive)

Ex: $[3, 10] = 3, 4, 5, 6, 7, 8, 9, 10$

Concept 3

Arithmatic Progression

$a, a+d, a+2d, a+3d$ ......

$$\sum AP \Rightarrow \frac{N}{2}\left[2a + (N-1) \times d\right]$$

Q: Sum of N numbers of AP?

Ex: 4, 7, 10, 13, 16 ....

+3 +3 +3 +3

$a, a+3, a+3\times 2, a+3\times 3$ ....

Concept 4 Geometric progression

$a , ar , ar^2 , ar^3 \ldots\ldots$

Q: Sum of N numbers of GP?

$$\sum GP = \frac{a \times (r^n - 1)}{r - 1}$$

Ex: 5 , 10 , 20 , 40 - - - - - -

$5 \quad 5\times2 \quad 5\times2^2 \quad 5\times2^3 \; - \; - \; -$

$a \quad ar \quad ar^2 \quad ar^3 \; - \; - \; -$

Quiz:

3**7 = 2187

1 , 3 , 9 , 27 - - - - -

$$\frac{a \times (r^n - 1)}{r - 1} \Rightarrow \frac{1 \times (2187 - 1)}{3 - 1}$$

$$\Rightarrow \frac{2186}{2}$$

Time Complexity

- measure of Efficiency

(No of ops performed w.r.t Size of input)

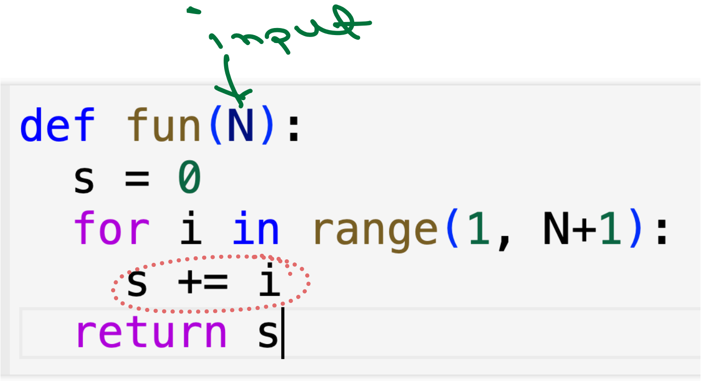
- Growth ⇒ How num-ops grow as input-size increases

+ ×

- Constants and Co-efficients ✗ : Focus on trend and ignore Constants Co-efficients

- Trend ✓

* Big O Notation : Only the polynomial of Highest degree is considered relevant



input

```
def fun(N):
  s = 0
  for i in range(1, N+1):
    s += i
  return s
```

[1, N] ⇒ N-~~1+1~~

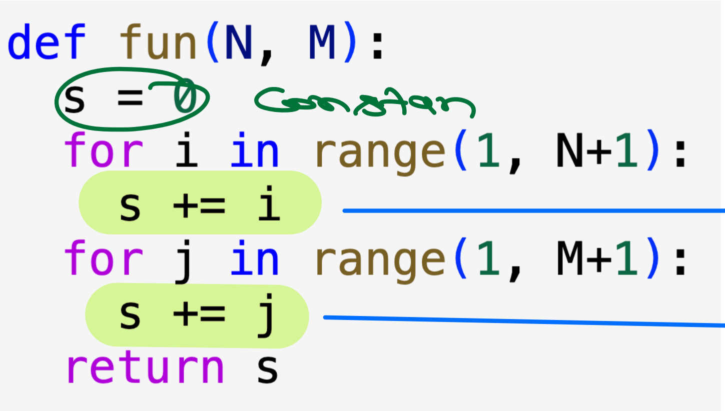
⇒ N

N ops

O(N)

- Rajat (Astronaut) $\xrightarrow{1500km}$ Delhi $\xrightarrow{8000km}$ Florida ↓ 20km NASA → 5million → moon

* O represents ⇒ most Relevant portion of Code



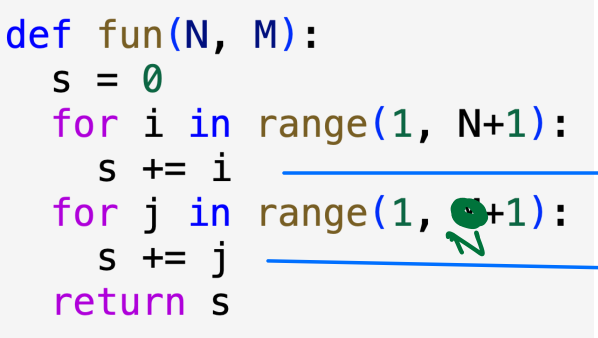
```
def fun(N, M):
  s = 0
  for i in range(1, N+1):
    s += i
  for j in range(1, M+1):
    s += j
  return s
```

constan

N

M

⊙ O(N+M)



```
def fun(N, M):
  s = 0
  for i in range(1, N+1):
    s += i
  for j in range(1, M+1):
    s += j
  return s
```

N

N

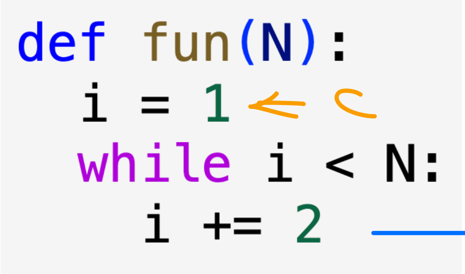
N

Co-effici

⊙ O(2N)

O(N)

i



```
def fun(N):
  i = 1
  while i < N:
    i += 2
```

← c

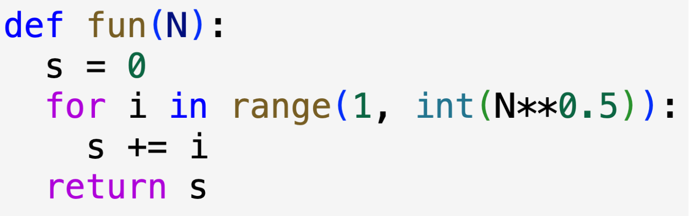
1, 2, 4, 6 . . . . .

(N/2) + i

Co-eff

O(N/2) $O(\frac{1}{2} \times N)$

O(N)

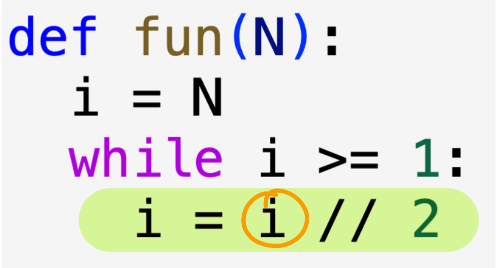


```
def fun(N):
  s = 0
  for i in range(1, int(N**0.5)):
    s += i
  return s
```

⇒

$O(N^{1/2})$

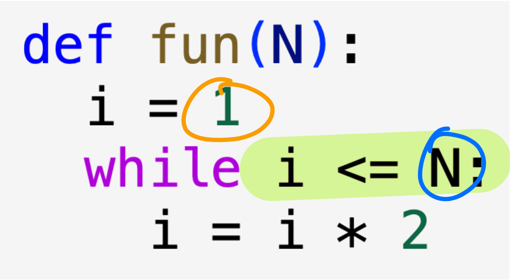
$O(\sqrt{N})$



```
def fun(N):
    i = N
    while i >= 1:
        i = i // 2
```

i-before	iteration	i-after
N	1	N//2
N//2	2	$N//4 = N//2^2$
N//4	3	$N//8 \Rightarrow N//2^3$
		⋮
	k	$N//2^k < 1$

$\boxed{O(\log_2(N))}$



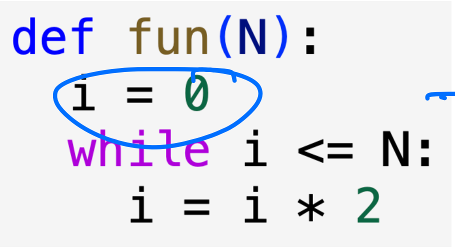
```
def fun(N):
    i = 1
    while i <= N:
        i = i * 2
```

$\boxed{O(\log_2(N))}$

i-before	iteration	i-after
1	1	2
2	2	$4 \Rightarrow 2^2$
4	3	$8 \Rightarrow 2^3$
8	4	$16 \Rightarrow 2^4$
	⋮	⋮
	k	2^k

$i \Rightarrow 2^k \leq N \Rightarrow \log(2^k) = \log N$

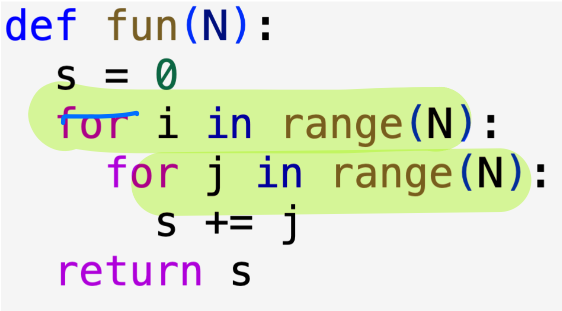
$k \times \underset{1}{\cancel{\log_2(2)}} \Rightarrow \log_2 N$



```
def fun(N):
    i = 0
    while i <= N:
        i = i * 2
```

Infinite loop

$O(\infty)$

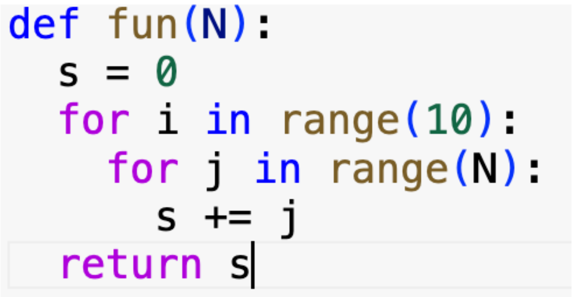


```
def fun(N):
  s = 0
  for i in range(N):
    for j in range(N):
      s += j
  return s
```

$O(N^2)$

i	j	iterations
1	[1, N]	N
2	[1, N]	N
:	[1, N]	N
N	[1, N]	N

$$\underbrace{N + N + N + - - - N}_{n} = N \times N$$

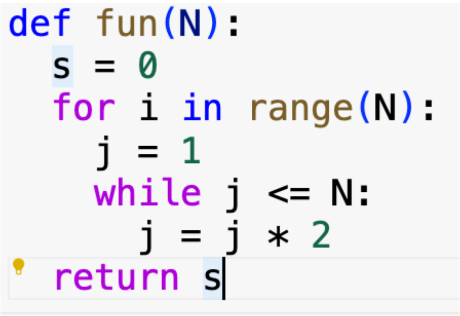


```
def fun(N):
  s = 0
  for i in range(10):
    for j in range(N):
      s += j
  return s
```

$O(N)$

i	j	iterations
1	[1, N]	N
2	[1, N]	N
3	[1, N]	N
:		
10	[1, N]	N

$10 \times (N)$

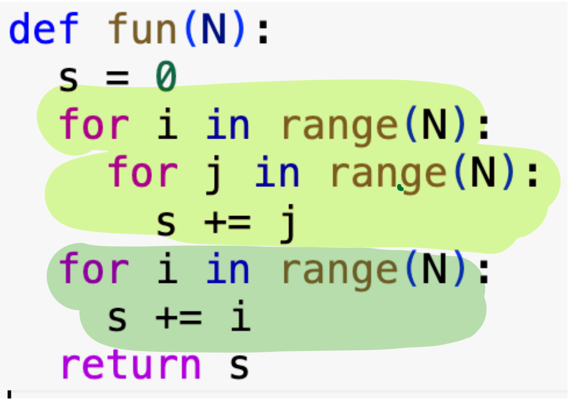


```
def fun(N):
  s = 0
  for i in range(N):
    j = 1
    while j <= N:
      j = j * 2
  return s
```

i	j	iterations
1	[1, N]	$\log_2(N)$
2	[1, N]	$\log_2(N)$
⋮		
N	[1, N]	$\log_2(N)$

$O(n \times \log_2 N)$

$$\underbrace{\log_2 N + \log_2 N + ---- \log_2 N}_{n}$$



```
def fun(N):
  s = 0
  for i in range(N):
    for j in range(N):
      s += j
  for i in range(N):
    s += i
  return s
```

N^2

N

$\Rightarrow O(N^2 + N)$

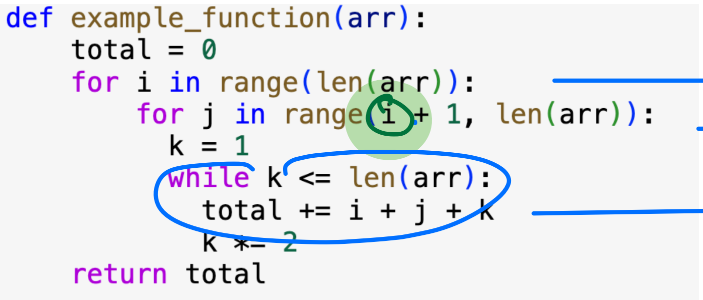
$O(N^2)$

$$\text{num\_ops} = 100n^2 + 30N^3 + 1000$$

× $n^2 + n^3$ × c ×

↓

$Tc(O) \Rightarrow O(n^3)$



```python
def example_function(arr):
    total = 0
    for i in range(len(arr)):
        for j in range(i + 1, len(arr)):
            k = 1
            while k <= len(arr):
                total += i + j + k
                k *= 2
    return total
```

1 2 3 N

1 → N, 2 → N, 3 → N

$\log_2 N$

N → N

$1+N + 2+N + 3+N$ ------

$$N(N) + \underbrace{1 + 2 + 3 \text{ ---- } N}_{\frac{n \times (n+1)}{2}} \rightarrow AP$$

$$\left(n^2 + \frac{n^2}{\cancel{2}} + \cancel{\frac{n}{2}} \right) \quad 2(N^2)$$

$$O(n^2) \times \log_2 N$$

Comparison of Order of Time Complexity

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

$$N < N\log N < N^2 < 2^N < N!$$

$N = 32$

$$O(1) < \log_2 32 < \sqrt{32} < 32$$

5 5.65

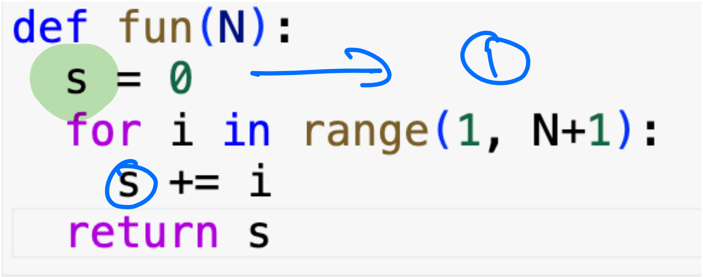
$$32 < 3 \times 5.65 < 32^2 < 2^{32} < 32!$$

180 1024

Space Complexity

→ memory a program needs to complete Execution

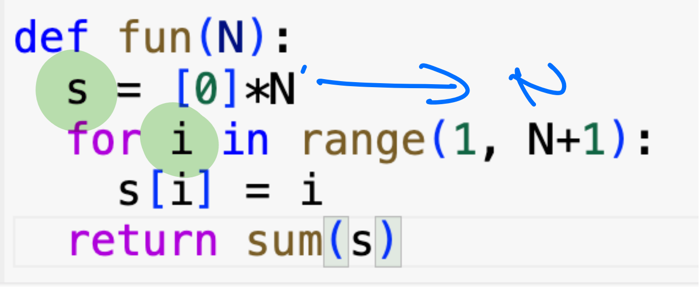
→ Similar to TC, SC is also calculated as Big O notation



```
def fun(N):
  s = 0
  for i in range(1, N+1):
    s += i
  return s
```

→ 1

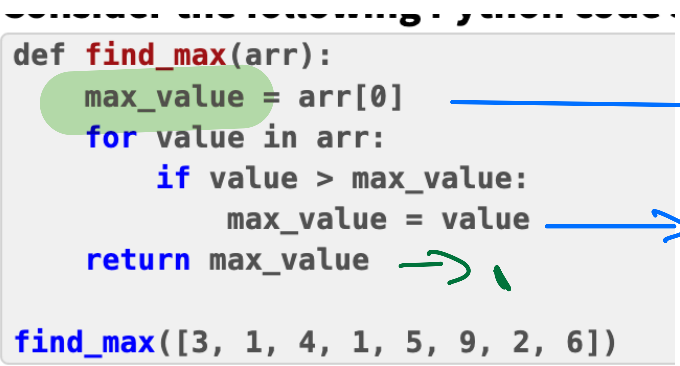
SC → O(1)
Since we have one Variable storing a Single Value Only



```
def fun(N):
  s = [0]*N
  for i in range(1, N+1):
    s[i] = i
  return sum(s)
```

→ N

SC → O(N)



Consider the following Python code:

```
def find_max(arr):
    max_value = arr[0]
    for value in arr:
        if value > max_value:
            max_value = value
    return max_value

find_max([3, 1, 4, 1, 5, 9, 2, 6])
```

→ 1

→ 1

→ 1

O(1)

```
def func(email)
    if ______
    if _ _ _
        return True or False

sort(filter( Func , list_email))
```