Time and Space Complexity

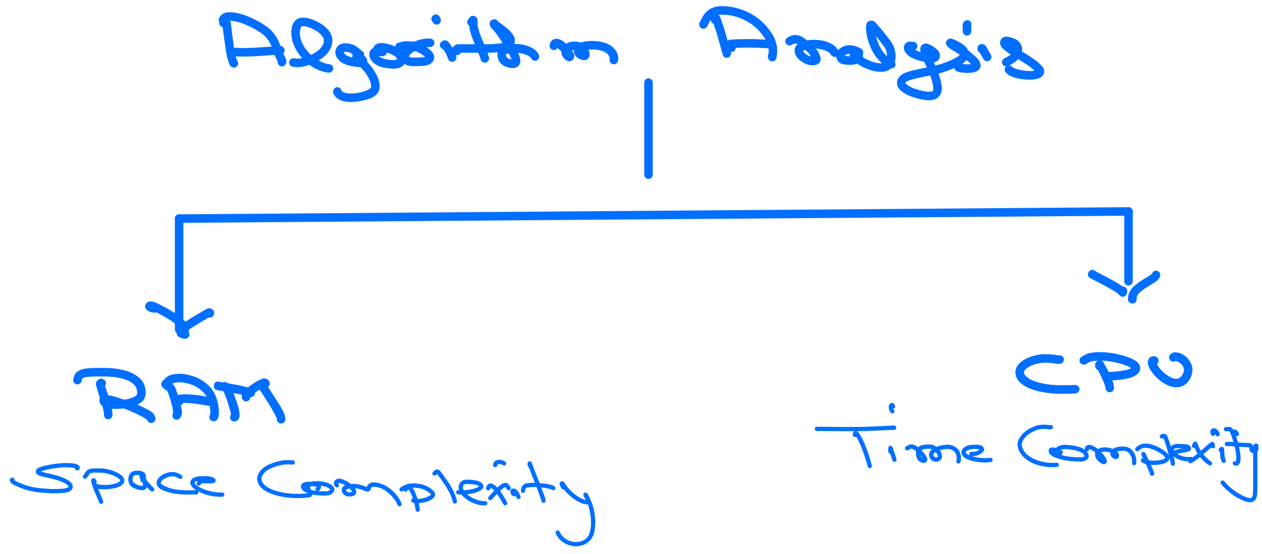
① How to measure efficiency of code

→ Memory

→ Execution speed

② Big O notation

③ Linear Search vs Binary Search



Algorithm Analysis

RAM
Space Complexity

CPU
Time Complexity

V1 ⇒ 3 sec

N1 ⇒ 5 sec

N1
2.5 GHz
V1 ⇒ 5 GHz

$$\text{Clock} = \frac{\text{No of ops}}{\text{Per second}} \Rightarrow 2.5 \times 1024 \times 1024 \times 1024$$

⇒ Big O notation

⇒ N inputs, No of instructs

⇒ N inputs, how much extra space an algo takes

$$\boxed{\text{Big O notation only cares about growth rate or trend}}$$

O(n)

↑

Polynomial term with highest degree

Ex: $x = \underbrace{3n^2}_{n^2} + 2n + c$

O(n) ⇒ n^2

Logarithic

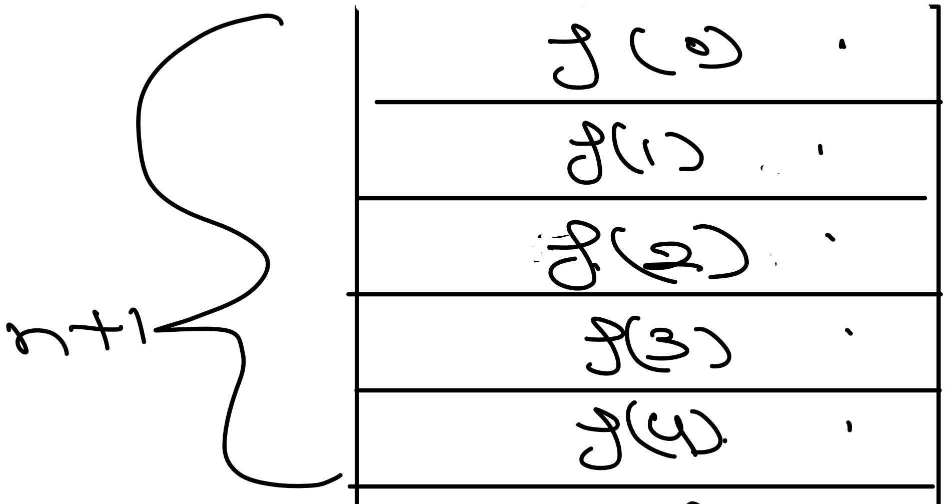
```
n=16
while n>0:
    print('Hello')
    n=n//2
```

$n \Rightarrow \overset{2^1}{2} \quad \overset{2^2}{4} \quad \overset{2^3}{8} \quad \overset{2^4}{16} \quad \overset{2^5}{32} \; --- \; 2^n$

$\text{ops} \Rightarrow 2 \quad 3 \quad 4 \quad 5 \quad 6 \quad n+1$

$$n \Rightarrow \log_2(n) + 1$$

$f(5) \Rightarrow$



1

$1 \times f(0) \Rightarrow 1$

$2 \times f(1) = 2$

$3 \times f(2) = 6$

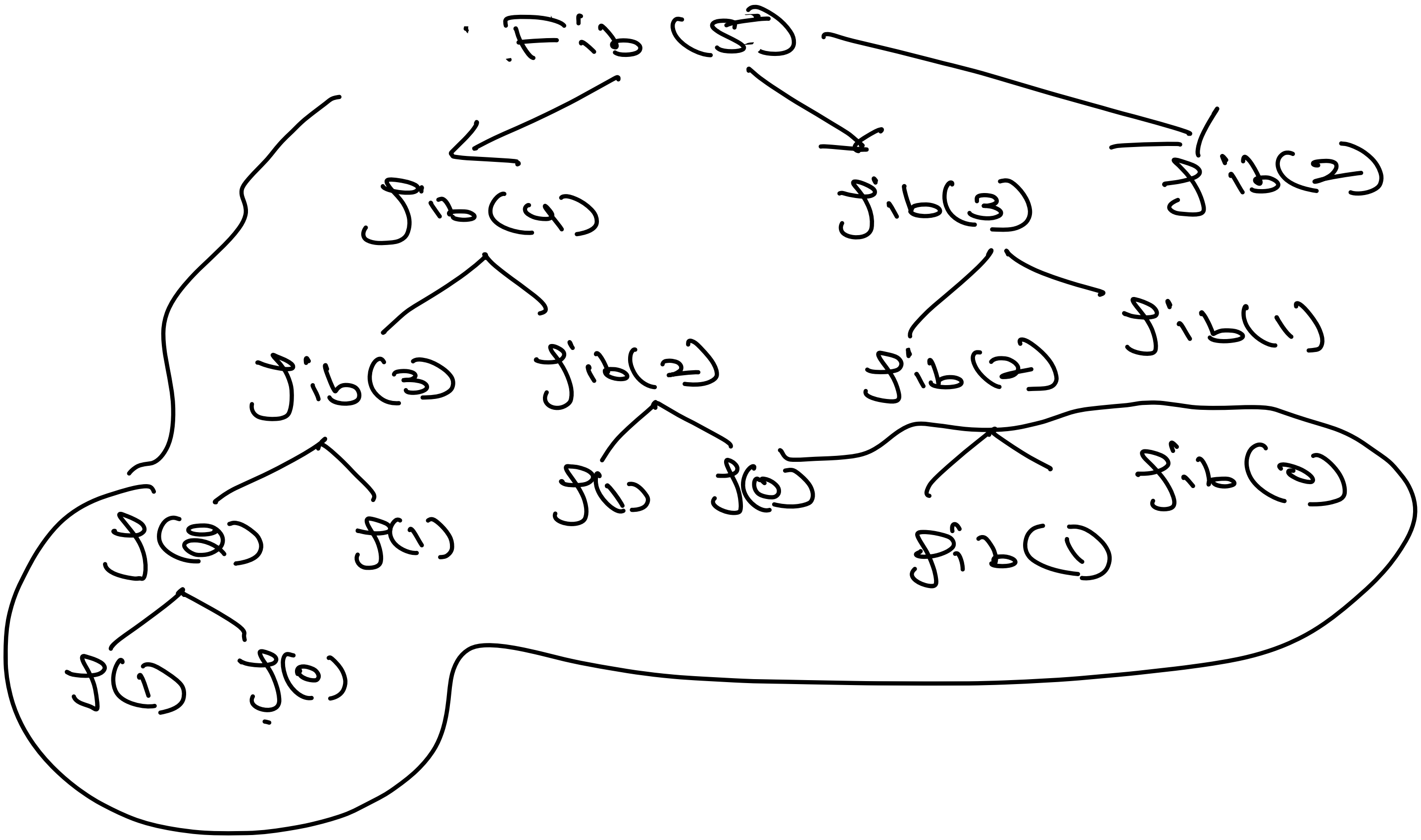
$4 \times f(3) = 24$

$5 \times f(4) = 120$

$TC \Rightarrow n+1 \Rightarrow O(n)$

SC => n+1 => o(n)

=>



n => 2

exponential => $\boxed{2^n + c}$ TC

$[0, \underbrace{1]}$

$[0, 1, \underbrace{1]}$

in

→ list ⇒ O(n)

→ set and dicts ⇒ O(1)

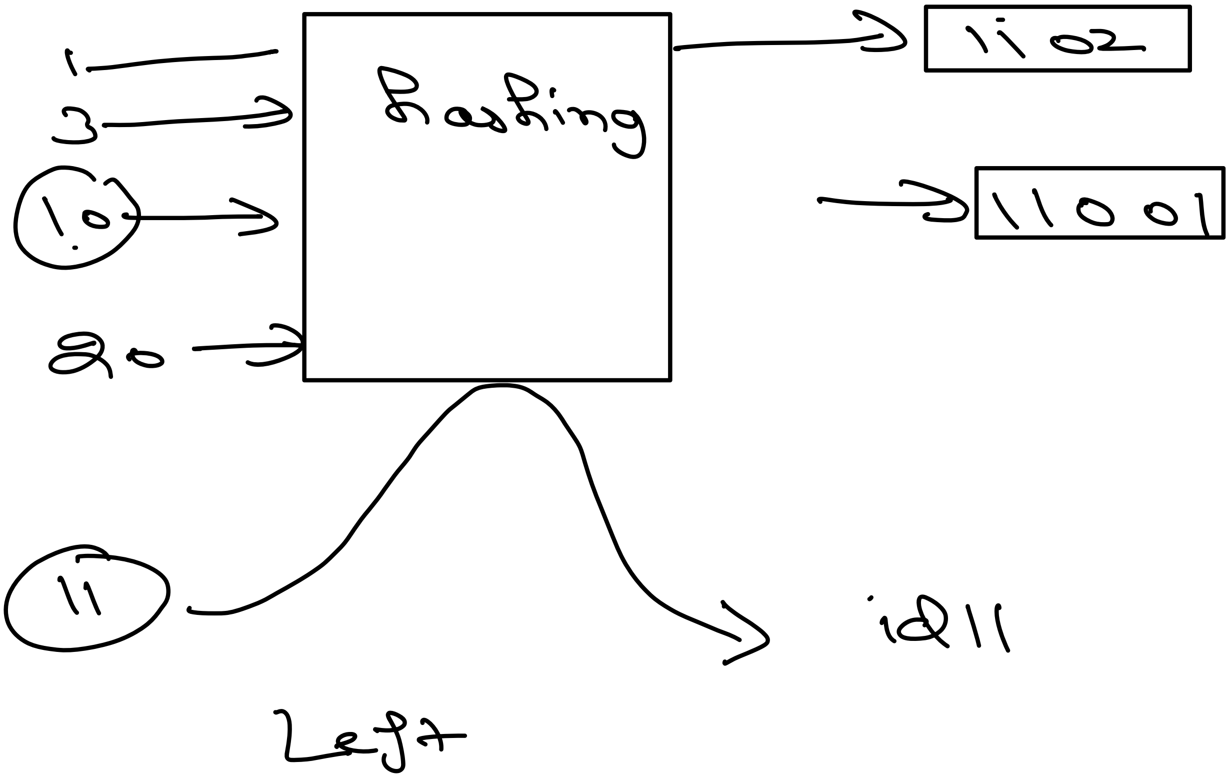
* in operation in list

```
for element in list1:
    if element == Target:
        return True
return False
```

* in operation works g hashing ds

{1 3 10 20}

Target = 11



nlogn()

[1 , 2 , 3 , 4 , 5 , 10]

i=0 j=N-1

Target => 11

```
if [mid] > target
    j = mid-1

else:
    i = mid+1
```

4 5 10

i j

5>11 i

10 mid=10 11

i i = N+1

j j > i