Time Complexity

time taken by the program to execute

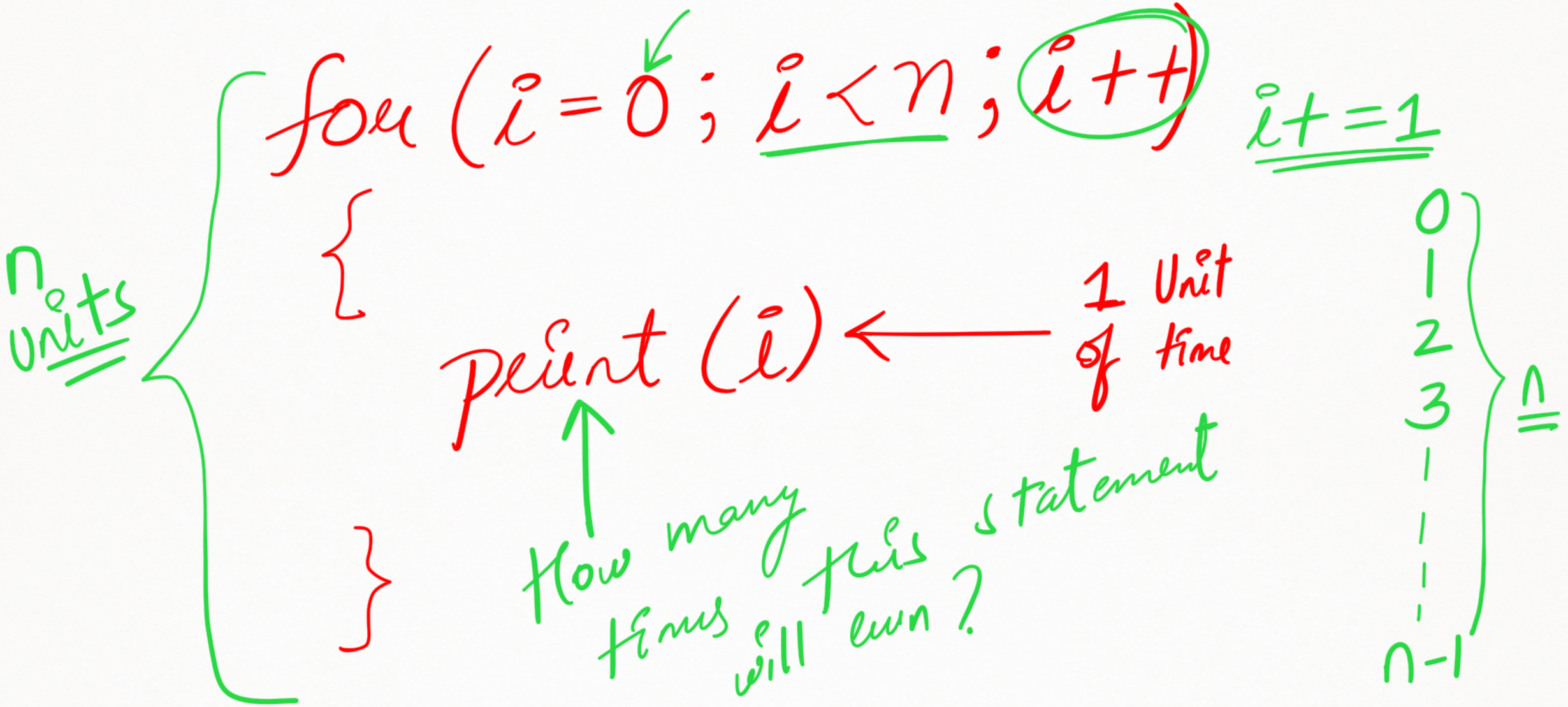
Time Complexity is used to compare various algorithms, and choose the best one for our purpose

We have build our own unit of time here, for various statements and loops in a program.

Time Complexity is Programming language independent.

```
# x = 3
  x = x+1
  print(hello)
```

} Very v basic statements like these takes 1 Unit of time


for (i = 0; i < n; i++)
i += 1
{
print (i)
1 Unit of time
}
How many times this statement will run?
n units
0
1
2
3
n-1
n

```
for (i=0; i<5; i+2)
{
    print (i)   → 1
    print (i+1) → 1
}
```

2

i
0
2
4

3

3 * 2 = 6

```
p = 0
for (i = 1; p <= n; i++)
{
    p = p + i;
}
```

✓

$p > n$

Let assume it runs for k times.

i	p
1	0 + 1 = 1
2	1 + 2 = 3
3	1 + 2 + 3 = 6
4	1 + 2 + 3 + 4 = 10
k	1 + 2 + 3 + 4 + ... + k = k(k+1)/2

~~$k^2 + k^3 + 3$~~

$\frac{k(k+1)}{2}$

$\frac{k(k+1)}{2} > n$

$k^2 > n$

$k > \sqrt{n}$

```
for (i=n; i>=1; i=i/2)
{ print(i) }
```

Example Binary Search

$i < 1$

$\frac{n}{2^k} < 1$

$n < 2^k$

$n \simeq 2^k$

$k = \log_2 n$

i

n

$\frac{n}{2}$

$\frac{n}{2^2}$

$\frac{n}{2^3}$

$\frac{n}{2^k}$

Types of Time Complexity Functions

$1 < \log n < \sqrt{n} < n < n\log n < n^2 < n^3 \ldots$

$< 2^n < 3^n < \ldots\ldots < n^n$

Exponential Time

Big O ⇒ Upper bound

$f(n) = O(g(n))$

iff $f(n) \leq c * g(n)$ $\forall$ $n \geq n_0$

{c & n_0 +ve const}

$O(n)$

$f(n) = 2n+3$

$2n+3 \leq 5n$

$n \geq 1$

$O(n^2)$

$O(n^3)$

Omega ⇒ lower bound

$f(n) = \Omega(g(n))$

$f(n) \geq C * g(n) \quad \forall n \geq n_0$

$f(n) = 2n+3$

$2n+3 \geq n$ ✓

- $\Omega(n)$ ✓
- $\Omega(\log n)$ ✓
- $\Omega(1)$ ✓

#Theta $f(n) = \Theta(g(n))$

$C_1 g(n) \leq f(n) \leq C_2 g(n)$

$f(n) = 2n+3$

$n \leq 2n+3 \leq 5n$

$O(n)$

Best, Worst & Average Case

Example ⇒ Linear Search

Key = E

Key = A

Best Case

Key = G

Worst Case

Key = C

Average Case

0s, 1s, 2s in Ascending Order

⇒

Best case

Worst case ⇒ Execution of swapping + required low ⇒

Parenthesis balancing

{ } ( ) [[ ]]

Best Case = ""

= "}{{{{{{

100

Worst Case => {{{{{

{{{{{{}}}}}}

Best Case ⇒ A case where the algorithm takes least amount of time.

Worst Case ⇒ Most amount

Average ⇒ Average amount.

Space Complexity

=) Max amount of space/memory occupied when running the Algorithm at any point of time.

Unit of Space

$x = 3$ } 1 Unit of Memory.

n elements } n Units of Memory

Homework

1. Hackerrank, leetcode, gfg

Practice
watch youtube dsa
videos.