Lec-11

02/05/25 7:08 PM

TLE

① Time & Space Complexity

Code → logic (Time, Space)

logic: min time, min Space

Time ✓ Space 2 ✓

asymptotic notation vs Experimental Way

① ② SP

4 5 1 0 0 → System

hardware ✗

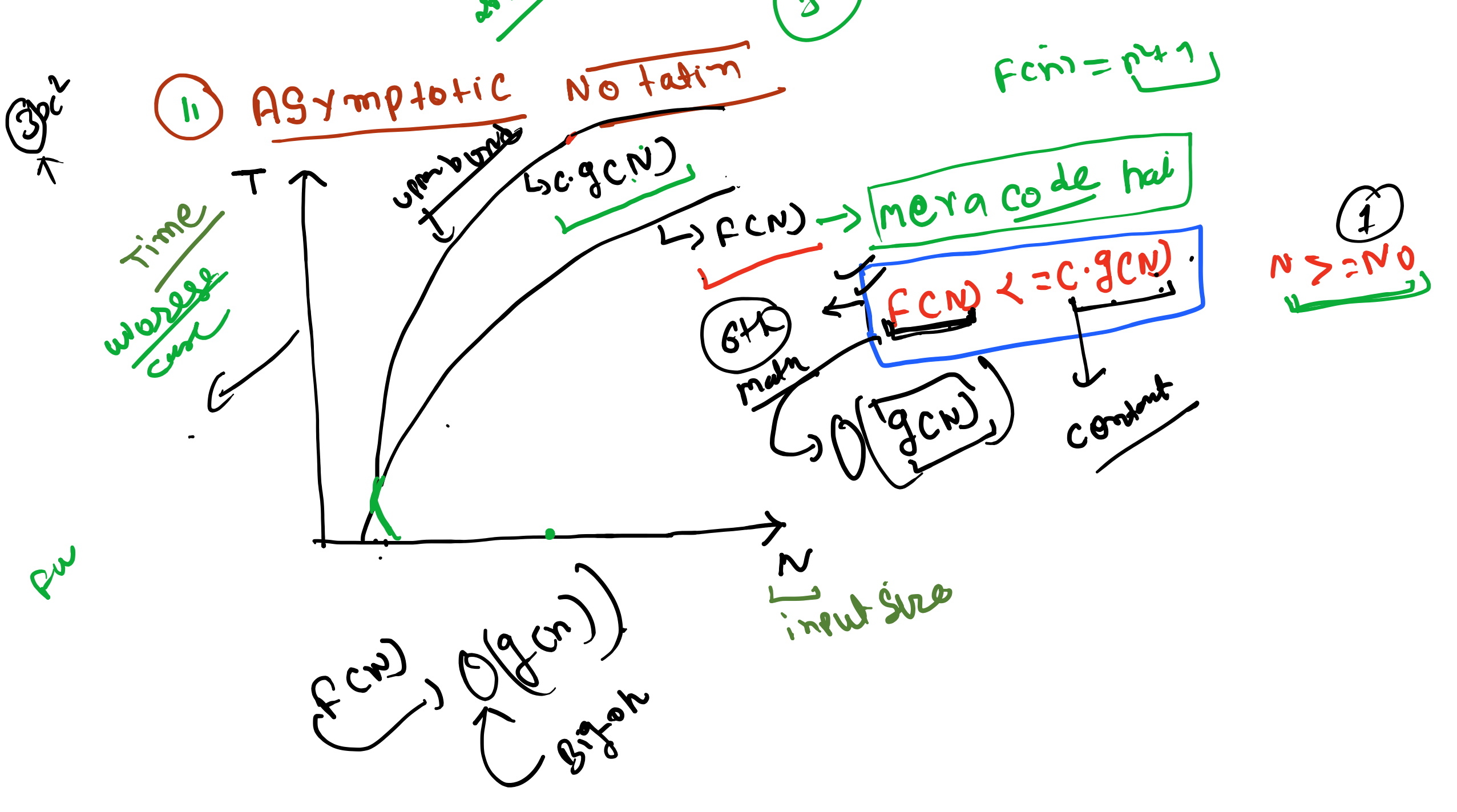
```
public static void main(String[] args) {
    // TODO Auto-generated method stub
    long start = System.currentTimeMillis();
    for (int i = 0; i < 1000_000; i++) {// -->10^6
    }
    long end = System.currentTimeMillis();
    System.out.println(end - start);
}
```

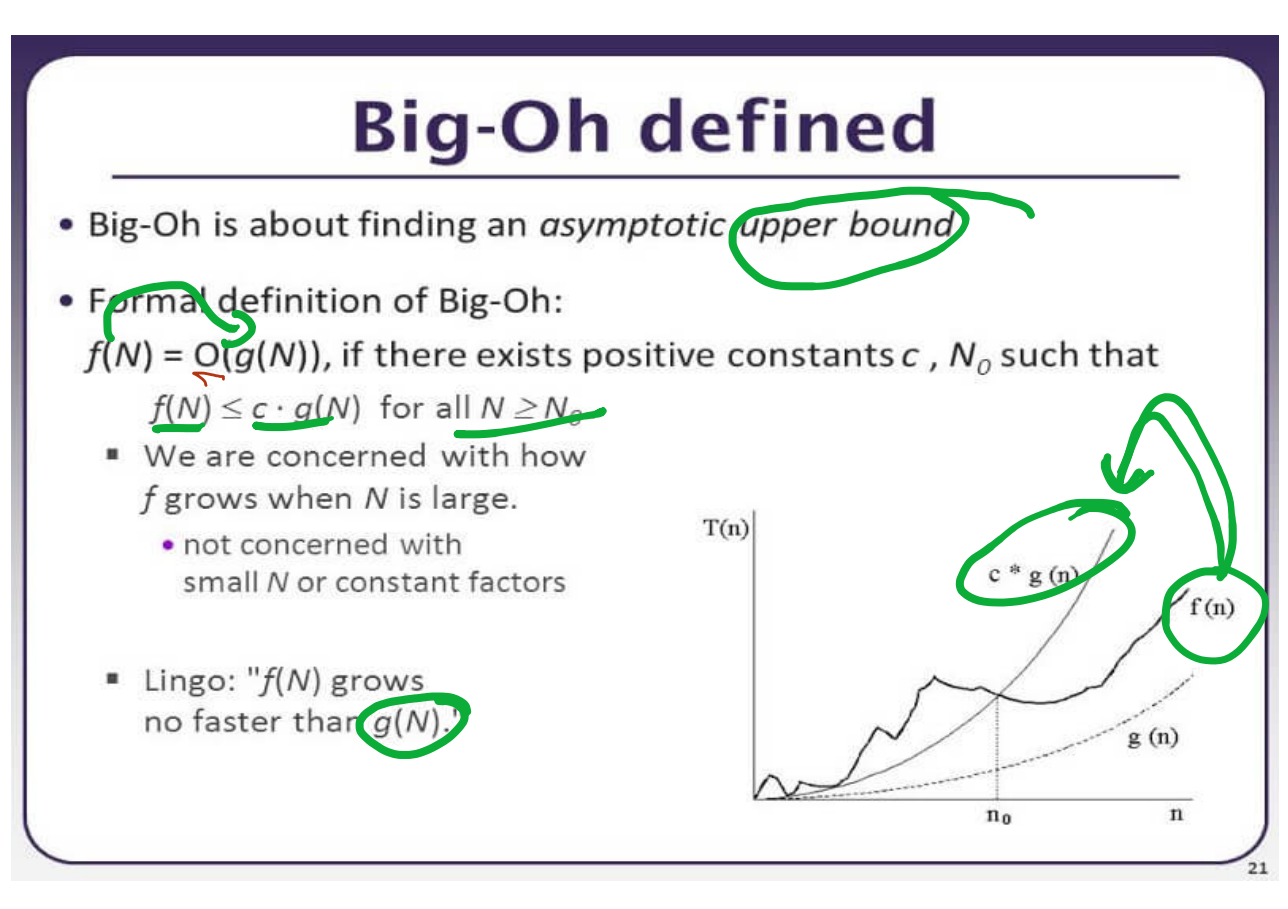
① 20to25 Time complexity

Asymptotic Notation =

① ⇒ worst case ② Best case ③ Avg case

meaning 1RS 20to25 2101 100+201





Big-Oh defined

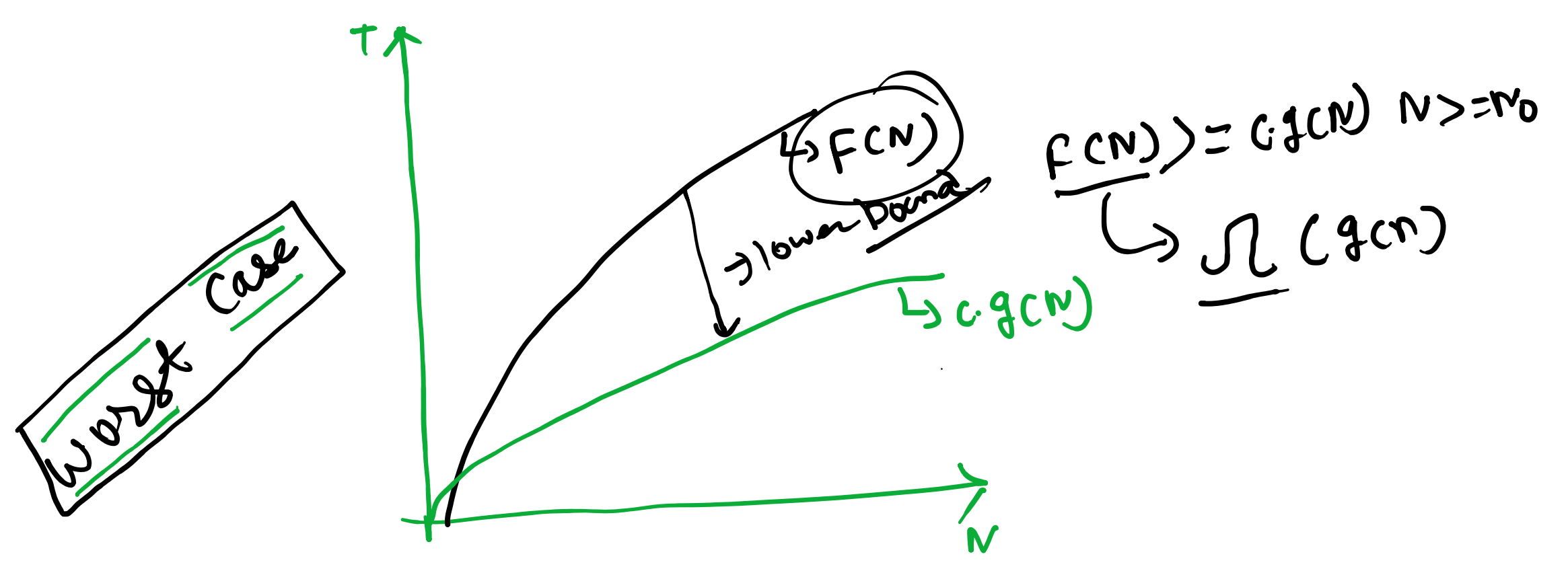
- Big-Oh is about finding an *asymptotic upper bound*.
- Formal definition of Big-Oh:
  $f(N) = O(g(N))$, if there exists positive constants c, N_0 such that $f(N) \le c \cdot g(N)$ for all $N \ge N_0$
  - We are concerned with how f grows when N is large.
    - not concerned with small N or constant factors
  - Lingo: "$f(N)$ grows no faster than $g(N)$."

T(n)
c * g (n)
f (n)
g (n)
n₀
n
21

Ω

O

O(g(n))



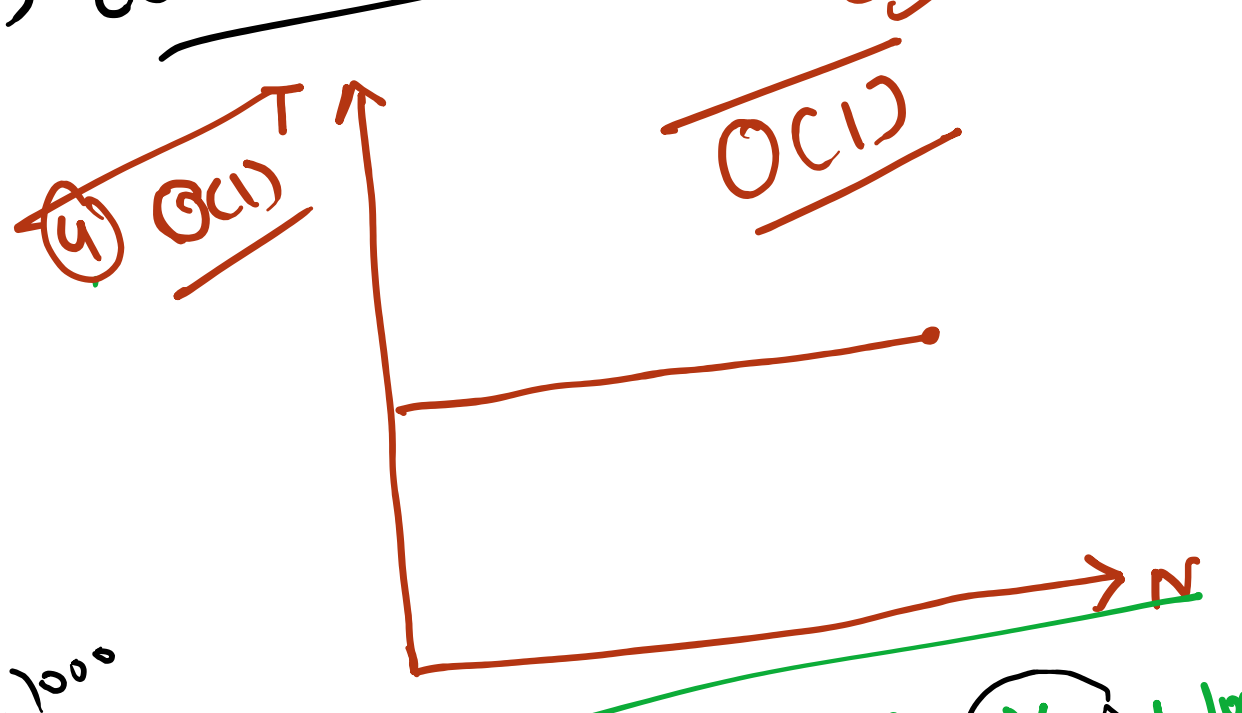
$F(N) \ge c.g(N)$ $N \ge N_0$ → $\Omega(g(n))$

⑪

→ {} mathematical

+ − * / % ++ -- → Constant Time

④ O(1)



O(1)

logic

$10^3 = 1000$

① $F(N) = 2N^2 + 5N^4 + 6N + 7$

N^4

$g(N) =$

$F(n) = 2N^3 + 2^N n + \log(N)$

$2^N \times N$

```
public static int Search(int[] arr, int item) {
    for (int i = 0; i < arr.length; i++) {
        if (arr[i] == item) {
            // System.out.println(i);
            return i;
        }
    }
    // System.out.println(-1);
    return -1;
}
```

compile

$F(N) \le c.g(N)$

i=0 i=1 i=2 i=3 --- i=n-1

1 1 1 1 1

$F(N) = N+1$ → O(N)

1+ i=0 i=1 i=2 --- i=n-1 ①

1 1 1 1

n+2

```
public static int Maximum_Value2(int[] arr) {
    int max = Integer.MIN_VALUE;// -2^31
    for (int i = 0; i < arr.length; i++) {
        max=Math.max(max, arr[i]);
    }
    return max;
}
```

→ O(1)

$2^{10} = 1024$

$\log_2 1024 = \log_2 2^{10} = 10 \times \log_2 2 = 10 \times 1 = 10$

$\log_a a^b = b \times \log_a a$

```
public static int Search(int[] arr, int item) {
    int lo = 0;
    int hi = arr.length - 1;
    while (lo <= hi) {
        int mid = (lo + hi) / 2;
        if (arr[mid] == item) {
            return mid;
        } else if (arr[mid] > item) {
            hi = mid - 1;
        } else {
            lo = mid + 1;
        }
    }
    return -1;
}
```

O(1)

(k+1)=k

$\log_2 N$

$\frac{N}{2^k} = 1$

$2^k = N$

$k = \log_2 N$

$\frac{N}{2^0}$ → 1

$N/2$ → 1

$\frac{N}{2^2}$ → 1

$\frac{N}{2^3}$ → 1

$\frac{N}{2^4}$

$\frac{N}{2^k}$ → 1

n

$\log_2 1024 = 10$

m_L

O(1L)

$\log_2 N$

10

9:07

time

N = 1024 → 512 → 256 → 128 → 64 → 32 → 16 → 8 → 4 → 2 → 1

1024

1<=n

$2 \times 2 = 2^{2+1} = 2^3$

i=1 i=2 i=2² i=2³ i=2⁴ ---- i=2^k → n

1 1 1 1 1

1 2 4 8 16 32 64 128 256 512 1024

```
while (i <= n) {
    System.out.println("Hey");
    i *= 2;
}
```

$\log_2(n)$

$\log n$

while(n>0){

n=n/2

O(1L)

$2^k = n$

$k \to \log(n)$

n/5 O(n)

i×5 1+5 100 20

i=0 i=5 i=10 i=15 --- i=100

1 1 10 1

```
while (i <= n) {
System.out.println("Hey");
        i += 2;
        i += 3;
    }
```

```
for (i = 1; i <= n; i++) {
            for (int j = 1; j <= n; j++) {
                System.out.println("hey");
            }
        }
```

n n n^2

① dependent

② independent

i=1 i=2 i=3 i=4 i=5 ---i=N

N N N N N N

$n \times n = n^2$

$i^2 <= N$

i=1 $i * i <= N$

```
for (i = 1; i * i <= n; i++) {
        System.out.println("hey");
    }
```

$i^2 <= N$

$i <= \sqrt{N}$

O(n⁴)

```
for (i = 1; i <= n; i++) {
            for (int j = 1; j <= i * i; j++) {
                for ( k = 1; k <= n / 2; k++) {
                    System.out.println("hey");
                }
            }
        }
```

i=1 i=2 i=3 i=4 i=5 ----i=N

$\left[\frac{N}{2} \quad 2^2 \times \frac{N}{2} \quad 3^2 \times \frac{N}{2} \quad 4^2 \times \frac{N}{2} \quad 5^2 \times \frac{N}{2} \quad N^2 \times \frac{N}{2}\right]$

$\frac{N}{2}\left[1 + 2^2 + 3^2 + 4^2 + 5^2 --- N^2\right]$

$\left[\frac{N}{2} \times \left[\frac{n \times (n+1)(2n+1)}{6}\right]\right]$

n^4

cos(x) sin(x) Tan(x) log(n) e^n

9.45

1 2 3 4 --- N

8.5

```
for (i = 1; i <= n; i++) {
            for (int j = 1; j <= n; j += i) {
                System.out.println("hey");
            }
        }
```

i=1 i=2 i=3 i=4 --- i=N

$\left[N \quad \frac{N}{2} \quad \frac{N}{3} \quad \frac{N}{4} \quad \frac{N}{N}\right]$

$N\left[1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + --- \frac{1}{N}\right]$

$\log(N) = 1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + --- + \frac{1}{N}$

$N \times \log(N)$

1 sec → 10^8

$N < 10^3$

n^3 → $(10^3)^3$ = 10^9

n^2, $n^2 \log n$

n^2

$(10^5)^2 = 10^{10}$

$N <= 10^5$