Lec-10

12/07/25 4:49 PM

Time & Space Complexity

asymptotic notation vs Experimental Way

Algo → best / worst

1) code → Time

input size

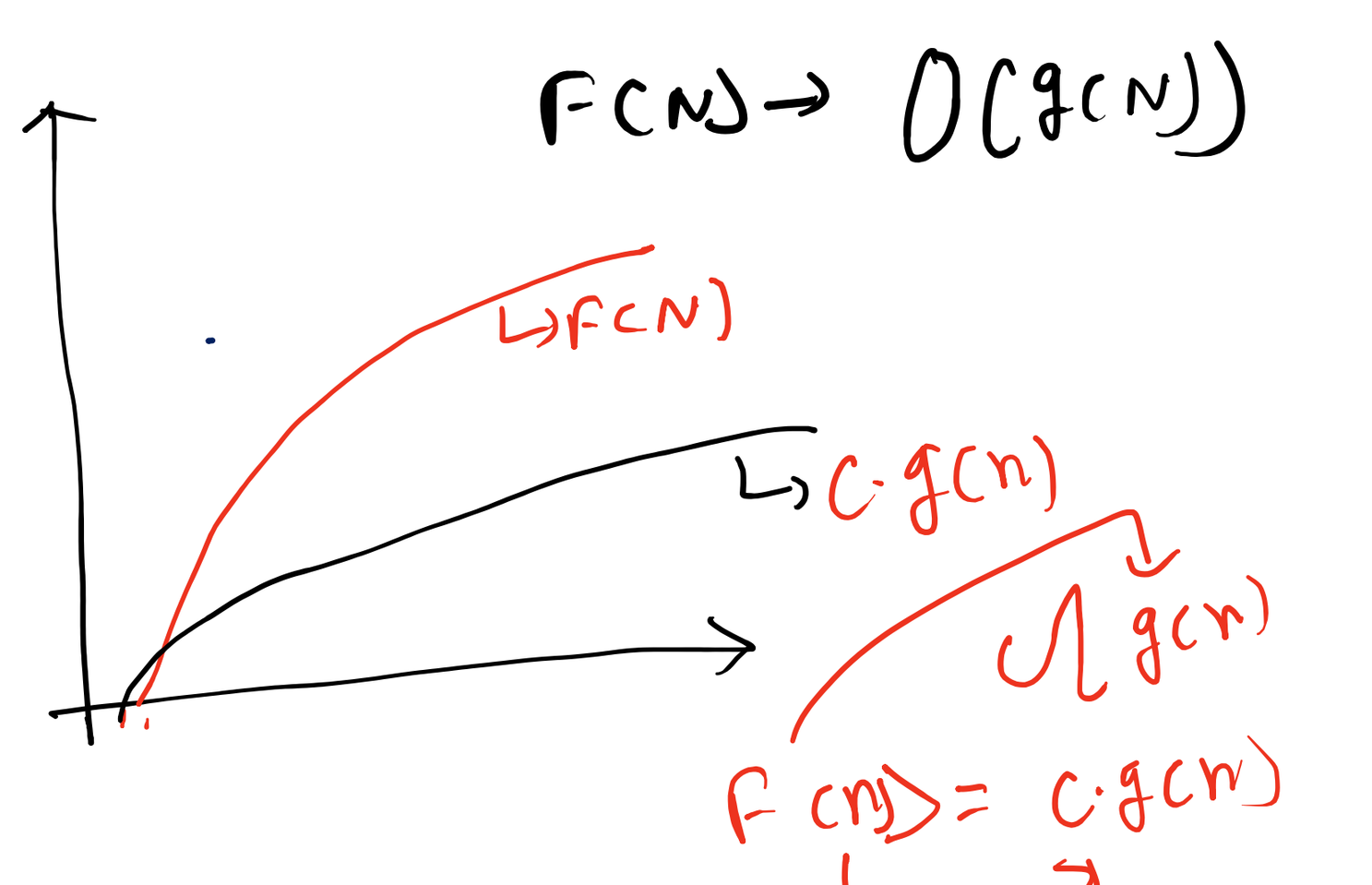
30 min

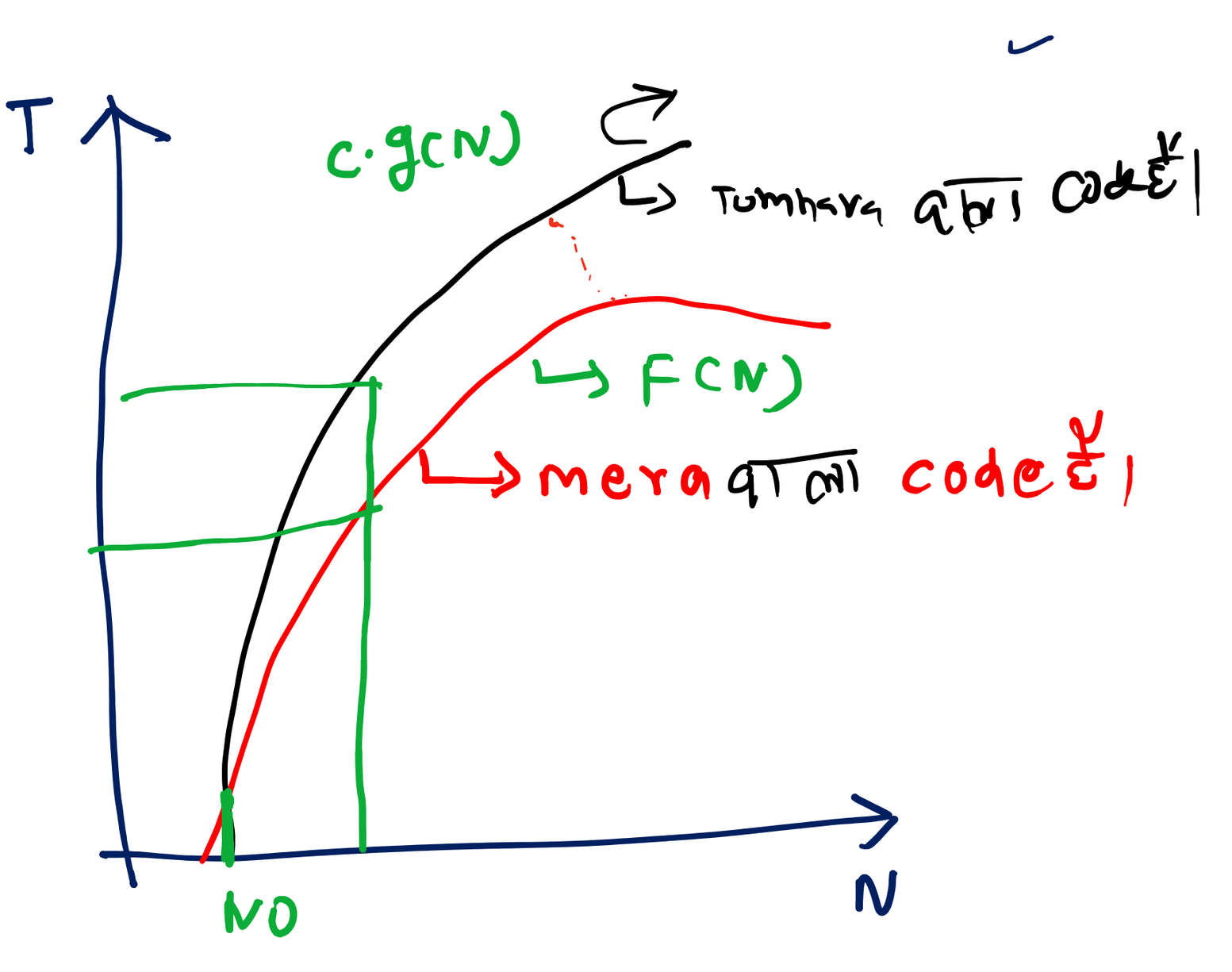
best | Avg | worst

[i] best Avg worst

$F(N) \le c \cdot g(N) \quad N \ge N_0$

$F(N) \to O(g(N))$





1) $T(N) = 3N^2 + 4N + 7$ → $O(n^2)$

$n \to \infty$

$T(n) = 3N^2 + 7(2^N)\log(n) + 3N^3 + 4$

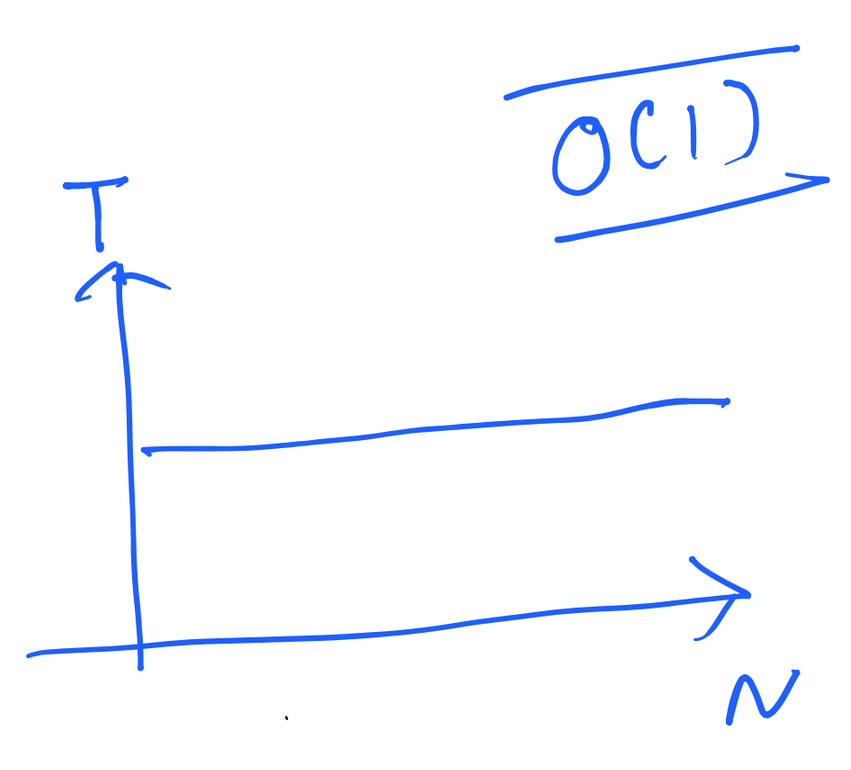
$2^N > N^3$

2^{100} 2) $(100)^3 = 10^6$

1)

```
// linaer
// binary
```

1) Constant Time → $O(1)$




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

i=0 i=1 i=2 i=3 i=4 … i=n-1

1 1 1 1 1 1

$T(N) = N+1 \to O(n)$

```java
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;
}
```

$N \to 1$, $\frac{N}{2} \to 1$, $\frac{N}{2^2} \to 1$, $\frac{N}{2^3} \to 1$, …, $\left[\frac{N}{2^K} = 1\right] \to 1$

$T(N) = K \to ? = \log_2 N$

$N = 2^K$, $2^K = N$, $[K = \log_2 N]$

```java
int i = 0;
    while (i < n) {
        System.out.println("Hey");
        i++;
    }
```

i=0 i=1 i=2 … i=n-1

$T(n) = N \to O(N)$

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

$i=2^0$ $i=2$ $i=2^2$ $i=2^3$ … $i=2^K$

$T(n) = K \to \log_2(n)$

$2^K = N$, $K = \log(n)$

while(n>0){ n=n/k }

while(i<=n){ $\log_k N$ } $i = i*k$

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

        i += 2;
        i += 3;
    }
```

i=0 i=5 i=10 … $i=\frac{N}{5}$

$\frac{N}{5} \to O(N)$

```java
while (i <= n) {

    System.out.println("Hey");
        // O(log(N))base 6
        i *= 2;
        i *= 3;
    }
```

i=1 i=6 $i=6^2$ $i=6^3$ … $i=6^K$

$6^K = N$, $K = \log_6 N$

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

dependent / independent

$n \times n \times 1 = N^2$

i=1 i=2 i=3 … i=N

Σ N N N … N → n^2

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

i=1 i=2 i=3 i×i≤n

$i^2 \le N$, $i \le \sqrt{N}$

```java
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");
        }
    }
}
```

j=1 j=2 j=3 j=4: $\frac{N}{2}$ $\frac{N}{2}$ $\frac{N}{2}$ $\frac{N}{4}$

i=1 i=2 i=3 i=4 … i=N

$\frac{N}{2}$ $2^2 \times \frac{N}{2}$ $3^2\frac{N}{2}$ $4^2\frac{N}{2}$ $N^2 \times \frac{N}{2}$

$\frac{N}{2}[1 + 2^2 + 3^2 + 4^2 + \dots N^2]$

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

$\frac{n \times (n+1)(2n+1)}{6}$

$= N^4 \to T.C$

```java
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

N $\frac{N}{2}$ $\frac{N}{3}$ $\frac{N}{4}$ $\frac{N}{N}$

$N[1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \dots \frac{1}{N}]$ → $\log(N)$

$N \times \log(N)$

6 N=10

N=13 7

3 2 4 2 7 2 2

+ −

e = 3 2 4 7 2

vote = 1 0 1 0 1 2

10

3 2

[5, 7, 3, 7, 2, 7, 7, 3, 7, 4, 7]

e = 5 7 3 7 2 7

v = 1 1 1 1 1 1 2