V.Imp.

FREEMIND

Date

Page

Lecture - 22

Time And Space Complexity

Time Complexity

Q→ You're given a no. x and y $(1 \le x \le 10^5)$, $x \le y \le 10^8$). Calc. sum of all the no. in the range $[x, y]$.

Sol→ $x = 5$, $y = 8$

Sum → $5+6+7+8 = 11+7+8 = 18+8 = 26$

Method 1→ (Basic Method)

→ Iterate from $x \rightarrow y$ and sum up all the no.s.

```
int ans = 0
    for (int i = x; i <= y; i++){
        ans += i;
    }
    Cout << ans;
```

Method 2→ $x = 2$, $y = 6$

$\{2, 3, 4, 5, 6\}$

- Sum of n terms in AP → $\frac{n}{2}(2a + (n-1)d)$

a → First term

n → no. of term

d → common difference

n = no. of terms = $(y - x + 1)$

$= 6 - 2 + 1 = 5$

Code concept

```
int n = (y - x + 1);
int a = x;
int result = (n * (2*a + (n-1) * k)) / 2 ;

cout << result;
```

~~# instru~~

Asymptotic Analysis

→ No. of operations w.r.t. change in input (no. of instruct)

{ Input size ko change karne pe tym taken by program kaise change ho rha hai. }

$$\boxed{\text{growth} = \frac{\Delta \text{ No. of instruct}}{\Delta \text{ Input size}}}$$

→ Const
log n
$\sqrt{n}$
n
n log n
n^2
n^3

Types of Time complexity Analysis and their notations:

- Worst case time complexity (O) → n instruction | O(n) — Big O
- Best case time complexity (Ω) → 1 instruction | Ω(1) — Big omega
- Average case time complexity (Θ) → n instruction | Θ(n) — Big theta

Date
Page

(A) Time Complexity for a traversing an array of length N.

```
int main(){
    int arr[]={1,2,3,4,5,6,7,8,9};
    int n=9;
    for(int i=0; i<n; i++){
        cout<<arr[i]<<"\n";
    }
    return 0;
}
```

$\boxed{O(n)}$

(A) Time complexity when traversing 2 individual arrays of length M and N respectively.

```
int main(){
    int arr1[] = {1,2,3,4,5,6,7,8,9};
    int n=9;
    int arr2[] = {1,2,11,4,6,7,9,12,8}
    int m=8;
    int sum1 = 0;
    int sum2 = 0;
    for(int i=0; i<n; i++){
        sum1 += arr1[i];
    }
    for(int i=0; i<m; i++){
        sum2 += arr[i];
    }
    cout<<sum1<<" "<<sum2;
    return 0;
}
```

3n

3m

$O(n+m)$

FREEMIND

Date

Page

Time Complexity for nested loops

```
for (i=0; i<n; i++){
    for(j=0; j<n; j++){  }  O(n)
    }
}
```

→ $\boxed{O(n^2)}$

(n^2 instructions)

Space Complexity :-

↳ It is the extra memory space requirement of an algorithm, using asymptotic analysis.

1130

Q3. Print n^{th} number of fibonacci series :-

0, 1, 1, 2, 3, 5, 8, 13, 21 ----

M1 ↳ Simple method

{Here size of array will inc. acc. to the input (n) no. of elements

```
a[0] = 0
a[1] = 0
for (i=2; i≤n; i++){
    a[i] = a[i-1] + a[i-2]
}
cout<<a[n]<<endl;
```

M2 →

```
a=0, b=1, c=1
for (i=2; i≤n; i++){
    c=a+b
    a=b
    b=c
} return c;
```

Date
Page

Lecture - 23

Problems on Time Complexity Calculation

Q1)

```
int value = 0;
for (int i=1; i<=n; i*=2){
    value++;
}
```

$i \to$ 1 2 4 8 16 ---- 2^k 2^{k+1}

2^0 2^1 2^2 2^3 2^4 ---- 2^k

1st 2nd 3rd 4th ---- $(k+1)^{th}$ term

$$2^k \le n$$
$$\log_2 2^k \le \log_2 n$$
$$k \log_2 2 \le \log_2 n$$
$$k \le \log_2 n$$

total iteration $= \log_2 n + 1$

$O(\log n)$

Q2)

```
int val = 0
for (int i=1; i<=N; i+=i){
    val++;
}
```

$\underbrace{i+=i}_{i = 2 \times i}$

{Same as Previous}

$O(\log n)$

1 2 4 8 16 ---- 2^k

Q3)

```
int val = 0;
for (int i=1; i<=N; i*=2){
    for (int j=1; j<=i; j++){
        value++;
    }
}
```

$\boxed{i \to 2^k}$

$j \to$ 1 + 2 + 4 + 8 + ---- 2^k $r=2$

$2^0 + 2^1 + 2^2 + 2^3 +$ ---- 2^k $\to$ gp

$\boxed{2^k - 1} \leftarrow 1 \times (2^k - 1) \leftarrow \dfrac{a(2^k-1)}{(2-1)}$

$2^k \leq N$ (taking log both side)

(i) $\Rightarrow k \leq \log_2 N$

(j) Total iterations $\Rightarrow 2^k \rightarrow 2^{\log_2 N} \longrightarrow O(N)$

Ans

Q 4)

```
int val = 0
for(int i = 1; i <= N ; i *= 2) {
    for (int j = N ; j > i ; j--) {
        val++
    }
}
```

(i)	(j)
i = 1	[N, 2]
i = 2	[N, 3]
i = 4	[N, 5]
i = 8	[N, 9]
⋮	⋮
$i = 2^{k-1}$	$[N, 2^{k-1}+1]$

Total iterations $\Rightarrow (N-1) + (N-2) + (N-4) + (N-8) + \text{------} (N - 2^{k-1})$

$$= \Big(\underbrace{N + N + N + N + \text{--} + N}_{k \text{ times}}\Big) - \left(2^0 + 2^1 + 2^2 + 2^3 + \text{---} 2^{k-1}\right)$$

$\Rightarrow NK + 2^{k} - 1$

$\Rightarrow \left(N \log_2 N + 2^{\log_2 N} - 1\right)$

$\Rightarrow N \log_2 N + N - 1$

$\Rightarrow O(N \log N)$ Ans

$2^{k-1} \leq N$

$k-1 \leq \log_2 N$

$\boxed{k \leq \log_2 N}$

Q 5)

```
int Val = 0;
for (int i = N ; i > 0 ; i /= 2) {
    for (int j = 0; j < i ; j++) {
        Val++;
    }
}
```

$i = N \longrightarrow j : \underbrace{[0, N-1]}_{N\text{-iterations}}$

$i = \frac{N}{2} \longrightarrow \underbrace{\left[0, \frac{N}{2} - 1\right]}_{\frac{N}{2}\text{ iterations}}$

$i = \frac{N}{2^2} \longrightarrow j : \underbrace{\left[0, \frac{N}{4} - 1\right]}_{\frac{N}{4}\text{ iterations}}$

$i = \frac{N}{2^{k-1}} = 1 \quad j : \underbrace{[0, 0]}_{1\text{ iteration}}$

$2^{\log a} = a$

$$\text{Total} = N + \frac{N}{2} + \frac{N}{4} + \frac{N}{8} + \cdots \frac{N}{2^{k-1}} \quad 1 \quad \frac{N}{2^{k-1}}$$

$$N\left(1 + \frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \cdots \frac{1}{N}\right)$$

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

$N = 2^{k-1}$

$2N = 2^k$

$\log 2N = k$

$\boxed{k \approx \log N}$

$\boxed{\text{it is a G.P.}}$

$$\frac{a \times (r^n - 1)}{(r-1)} \qquad \frac{a \times (1 - r^n)}{(1-r)} \quad \text{if } (r < 1)$$

$$\Rightarrow \frac{N\left(1 - \left(\frac{1}{2}\right)^{\log N}\right)}{1 - \frac{1}{2}}$$

$$\Rightarrow \frac{N\left(1 - \frac{1}{N}\right)}{\frac{1}{2}} = N\frac{(N-1)}{N}$$

$$\Rightarrow 2(N-1) \approx O(N) \quad \text{Ans}$$

Q → int val = 0;
for (int i = 2; i < N; i *= i){
    val++;
}

$2^k \leq N$

$k \leq \log_2 N \quad (1)$

$\boxed{k \approx \log_2 N}$

$2 \longrightarrow 4 \longrightarrow 16 \longrightarrow 256 \text{ ------ } k \Rightarrow$ final value of i

$2^1 \longrightarrow 2^2 \longrightarrow 2^4 \longrightarrow 2^8 \text{ ------ } 2^k$

Powers of 2 $\quad 2^0 \longrightarrow 2^1 \longrightarrow 2^2 \longrightarrow 2^3 \text{ ------ } 2^{T-1} \approx 2^T$

$2^T \leq k$

$T \leq \log k$

Use (1)

$T \leq \log(\log N)$

$O(\log(\log N)) \longleftarrow T \approx \log(\log N)$