Time Complexity Problems

Lecture- 20

Raghav Garg

Let’s jump to some starters first

Ques : Calculate the time complexity for the following code snippet.

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

$i = 2 * i \rightarrow i = i * 2 \rightarrow i *= 2$

$i = i + i$

No. of iterations = No. of values of i

$= n + 1$

$T.C. = O(n+1) = O(n)$

$i = 1, 2, 4, 8, 16 \ldots n$

$i = 1, 2^1, 2^2, 2^3, 2^4, \ldots 2^x$

$x + 1$ terms

$2^x = n$

$T.C. = O(\log_2 n)$

$T.C. = O(\log n)$

$$a^x = b$$

$$\log_a b = x$$

$$\rightarrow \quad 2^x = n$$

$$\Rightarrow \quad \log_2 n = x$$

$$O(\log_3 n) = O\left(\frac{\log_2 n}{\log_2 3}\right)$$

$$= O(\log_3 2 \cdot \log_2 n)$$

$$= O(K \cdot \log_2 n)$$

$$= O(\log_2 n)$$

Ques : Calculate the time complexity for the following code snippet.

```
int c = 0;
for(int i = 1; i <=n;  i += i) {
    for(int j = 0; j < i; j++) {
        c++;
    }
}
```

$i *= 2$

'i' times

$i = 1, 2^1, 2^2, 2^3, \ldots n \rightarrow 2^x$

$\sim x = \log_2 n$

$i = 1 , j = 0 \rightarrow 1$

$i = 2 , j = 0, 1 \rightarrow 2$

$i = 4 , j = 0, 1, 2, 3 \rightarrow 4$

$i = 8 , j = 0, 1, 2, 3, 4, 5, 6, 7 \rightarrow 8$

$\downarrow$

$i = n , j \rightarrow n$

Total No. of Iterations : (M-1)

$2^x = n$

$$= \underbrace{1 + 2^1 + 2^2 + 2^3 + 2^4 \ldots\ldots 2^x}_{x+1 \text{ terms}}$$

$$S_T = \frac{a[r^T - 1]}{r-1}$$

$$= \frac{1[2^{x+1} - 1]}{2-1} = 2^{x+1} - 1$$

$$\Rightarrow T.C. = O(2^{x+1} - 1) = O(2 \cdot 2^x) = O(kn)$$

$T.C. = O(n)$

Total No. of Iterations : (M-2)

$$\Rightarrow \quad 1 + 2 + 4 + 8 + 16 \cdots + \frac{n}{2} + n$$

$$\Rightarrow \quad (2 + 2) + 4 + 8 + 16 \cdots \quad \frac{n}{2} + n - 1$$

$$\Rightarrow \quad 2n - 1$$

$$T.C. = O(2n-1)$$

$$T.C = O(n)$$

$2x \rightarrow \boxed{n = 16}$

$1 + 1 + 2 + 4 + 8 + 16 - 1$

$2 + 2 + 4 + 8 + 16 - 1$

$4 + 4 + 8 + 16 - 1$

$8 + 8 + 16 - 1$

$16 + 16 - 1$

$32 - 1$

$\downarrow$

$2n - 1$

Now see the main course

Ques : Calculate the time complexity for the following code snippet.

```
int c = 0;
for(int i = 1; i <=n;  i += i) {
    for(int j = n; j >=0; j--) {
        c++;
    }
}
```

$i *= 2$

$T.C. = O(\log n)$

$T.C. = O(n)$

'n+1' times

$T.C. = O(n \cdot \log n)$

Ques : Calculate the time complexity for the following code snippet.

Classwork

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

$O(\log n)$

$O(n)$

'i' times → $O(i)$

$i = 1 \rightarrow j = 0 \rightarrow 1$

$i = 2, \rightarrow j = 0, 1 \rightarrow 2$

$i = 4; \rightarrow j = 0, 1, 2, 3 \rightarrow 4$

Ques : Calculate the time complexity for the following code snippet.

$i^2 < n \rightarrow i < \sqrt{n}$

$i = 1, 2, 4, 8, 16 \ldots\ldots 2^x$ (x+1 terms, $2^x = \sqrt{n}$)

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

$\rightarrow 2^x \cdot 2^x = n$

$\Rightarrow 2^x = \sqrt{n}$

Total iterations $= 1 + 2 + 3 + 4 + \ldots\ldots 2^x$

$= 2^{x+1} - 1$

$\rightarrow$ T.C. $= O(2^x) \Rightarrow$ **T.C. $= O(\sqrt{n})$**

Ques : Calculate the time complexity for the following code snippet.

```
int c = 0;
for(int i = 1; i * i < n; i += i) {
    for(int j = n; j > i; j--) {
        c++;
    }
}
```

$i < \sqrt{n}$ ↑ ; $i += i$ → $i *= 2$

'n-i'

$i = 1, 2, 4, 8, \ldots 2^x$

→ $2^x = \sqrt{n}$ → $x = \log_2 \sqrt{n}$

$i = 1$, $j = n, n-1, \ldots 3, 2$: $n-1$

$i = 2$, $j = n, n-1, \ldots 4, 3$: $n-2$

$i = 4$, $j = n, n-1, \ldots 6, 5$: $n-4$

⋮ : $n-8$

⋮

$i = \sqrt{n}$ → $n - \sqrt{n}$

Total No of iterations $= (n-1) + (n-2) + (n-4) + (n-8) + (n-16) + \ldots (n-2^x)$

Total No of iterations $= (n-1) + (n-2) + (n-4) + (n-8) + (n-16) + \ldots (n-2^x)$

$= \underbrace{(n + n + n + \ldots)}_{x+1} - \underbrace{(1+2+4+8+\ldots 2^x)}_{x+1}$

$= n(x+1) - [2^{x+1} - 1] = n \cdot x + n - 2 \cdot 2^x + 1$

$T.C. = O(n \cdot \log_2 \sqrt{n} + n - \sqrt{n}) = O(n \cdot \log \sqrt{n})$

$T.C. = O(n \cdot \log n^{1/2}) = O(n \cdot \frac{1}{2} \cdot \log n)$

$T.C. = O(n \cdot \log n)$

Thodi si Maths

$$\log_a b^m = m \cdot \log_a b$$

$$\Rightarrow \log_2 \sqrt{n} = \log_2 n^{1/2} = \frac{1}{2} \log_2 n$$

Ques : Calculate the time complexity for the following code snippet.

$i = 2, 4, 16, 256, (256)^2$

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

T.C. → Total no. of iterations = No. of values i can obtain

→ 2, 4, 16, 256, 256*256 . .

⇒ $2^1, 2^2, 2^4, 2^8, 2^{16}, 2^{32}, 2^{64} \ldots$

i = $2^1, 2^{2^1}, 2^{2^2}, 2^{2^3}, 2^{2^4}, 2^{2^5}, 2^{2^6} \ldots 2^{2^x}$

$x+1$

T.C. = $O(x)$

$\Rightarrow 2^{(2^x)} = n$

$\Rightarrow 2^t = n \quad [t = 2^x]$

$\Rightarrow \log_2 n = t$

$\Rightarrow \log_2 n = 2^x$

$2^x = K \quad [K = \log_2 n]$

$\log_2 K = x$

$\Rightarrow \log_2(\log_2 n) = x$

T.C. = $O(\log(\log n))$

Ques : Calculate the time complexity for the following code snippet.

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

$i <= \sqrt{n}$

$i = 2, 4, 16, 256, 256*256$

$i = 2, 2^2, 2^4, 2^8, 2^{16}$

$i = 2^1, 2^{2^1}, 2^{2^2}, 2^{2^3}, 2^{2^4} \dots 2^{2^x}$

$x+1$

$T.C. = O(x)$

$2^{(2^x)} = \sqrt{n}$

$\Rightarrow x = \log_2(\log_2 \sqrt{n})$

$\Rightarrow x = \log_2(\log_2 n^{1/2})$

$\Rightarrow x = \log_2(\frac{1}{2} \cdot \log_2 n)$

$\Rightarrow x = \log_2 \frac{1}{2} + \log_2(\log_2 n)$

$\Rightarrow T.C. = O(\log_2(\log_2 n) + \cancel{\log_2 \frac{1}{2}})$

$\Rightarrow T.C. = O(\log(\log n))$

Exponential → $O(2^n)$ → recursive calls

Sorting ??

Binary Search → $O(\log_2 n)$

Thank you!

→ 4 lectures