CSE221 ALGORITHMS

Topic: Tight Bound, Iterative Time Complexity

Prepared by:
Saad Bin Sohan
BRAC University

Email: sohan.academics@gmail.com

GitHub: https://github.com/saad-bin-sohan

(O) Upper bound → (upper + same), $\boxed{c,\ n_0}$;

$$f(n) \le c * g(n)$$

(Ω) Lower bound → (lower + same), $\boxed{c,\ n_0}$;

$$f(n) \ge c * g(n)$$

(θ) Tight bound → (same), $\boxed{c_1,\ c_2,\ n}$;

$$f(n) \le c_1 * g(n) \quad \text{and}$$

$$f(n) \ge c_2\, g(n)$$

(Upper + lower) → both prove করলে that's a tight bound complexity

☆ C_1, C_2 both এর জন্য n_0 same হওয়া mandatory.

($$$ not by sir this page

Tight bound means both the upper bound and lower bound are same for the computational time complexity of an algorithm. Tight bound is usually the average case time complexity of an algorithm.)

Que Verify,

$$2n^2 + 5n + 7 = \Theta(n^2)$$

<u>soln:</u>

(i) n^2 is the tight bound of $2n^2+5n+7$

(ii) True

$\boxed{\begin{array}{l} c_1 = 14 \\ c_2 = 1 \\ n_0 = 1 \end{array}}$

(iii) $2n^2 + 5n + 7 \leq c_1 n^2$

let $c_1 = 14$ $(\ldots \rightarrow 2+5+7)$

so,

$$2n^2 + 5n + 7 \leq 14n^2$$

let $n = 1$

$$2(1)^2 + 5(1) + 7 \leq 14(1)^2$$

$$14 \leq 14 \text{ (True)}$$

let $n=2$

$$2(2)^2+5(2)+7 \leq 14(2)^2$$

$$25 \leq 56 \text{ (also True)}$$

$\vdots$

hence proved that

n^2 is an upper bound of $2n^2+5n+7$

Again,

$$2n^2+5n+7 \geq c_2 n^2$$

let $c_2 = 1$

so,

$$2n^2+5n+7 \geq n^2$$

let $n = 1$

so, $2(1)^2+5(1)+7 \geq 1^2$

$$14 \geq 1 \text{ (True)}$$

let $n = 2$

so $2(2)^2 + 5(2) + 7 \geq 2^2$

$25 \geq 4$ (True)

so n^2 is also a lower bound for $2n^2 + 5n + 7$

since n^2 is both an upper bound and a lower bound for $2n^2 + 5n + 7$, its a tight bound.

Iterative Time Complexity

[Before that we have 3 simplification rules:

① যেকোনো $f(n) = O(\overset{\text{constant}}{k} * \overset{\text{function}}{g(n)})$

এর জন্য we can rewrite

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

যেমন: $f(x) = O(35n^2)$ হলে

$$\Rightarrow \quad f(x) = O(n^2)$$

② $f(n) = O(g_1(n)) + O(g_2(n))$ হলে

$f(n) = O(\max(g_1(n), g_2(n)))$

যেমন: $f(n) = O(n^2) + O(n^3)$

$\Rightarrow f(n) = O(n^3)$

③ $f(n) = O(g_1(n) * g_2(n))$ হলে

$\Rightarrow f(n) = O(g_1(n) * g_2(n))$

যেমন: $f(n) = O(n^2) * O(n^3)$

$\Rightarrow f(n) = O(n^2 * n^3)$

$\Rightarrow f(n) = O(n^5)$

$ Que আমাদের code থেকে time complexity বের করো।

Iterative Time Complexity

(i) $\underbrace{a=b}_{\text{code এর লাইন}} \longrightarrow O(1) \longrightarrow \text{constant}$

order of

(ii) $\left.\begin{array}{l} a=b \\ c=d \\ e=f \end{array}\right\} \longrightarrow O(3) = O(3\times 1)$

order of

$= \underbrace{O(1)}_{\text{constant}}$

[redacted]

(iii) <u>in python</u>

```
for i in range(4):
        print → ok
```

<u>in JAVA:</u>

$$\text{for } (\underbrace{i=0}_{\text{initialisation}};\ \underbrace{i<4}_{\text{condition}};\ \underbrace{i++}_{\text{increment / decrement}})\{$$

print → ok

}

condi-tion check	Output
0<4 →	ok
1<4 →	ok
2<4 →	ok
3<4 →	ok
4 ≮ 4	✗

✡ so loop ভেতরে থাকলে time complexity হবে।

$O(1) \longrightarrow$ constant

$O(n) \longrightarrow$ linear

[redacted]

```
for (i=0; i<n; i+3){
    print → ok
}
```

$\rightarrow O\left(\frac{n}{3}\right)$

$\approx O(n)$

step	i
0	$0 = 3\times 0$
1	$3 = 3\times 1$
2	$6 = 3\times 2$
3	$9 = 3\times 3$
$\vdots$	$\vdots$
$k \rightarrow$	$3k$

i এর max value

so $3k = n$

$k = \frac{n}{3}$

loop কতবার চলবে

for (i=0; i<5n; i+3){

print → ok

}

$$O\left(\frac{5n}{3}\right)$$

$$\approx O(n)$$

i < 5n ২3ড়াড়

$$O\left(\frac{5n}{3}\right)$$

$$O(n)$$

```
for (i=1; i<=n; i = i*2){
    print → ok
}
```

$\hookrightarrow O(\log_2 n)$

step	i
0	$1 \rightarrow 2^0$
1	$2 \rightarrow 2^1$
2	$4 \rightarrow 2^2$
3	$8 \rightarrow 2^2$
⋮	
k	$\rightarrow \underbrace{2^k}_{2^{step}}$

so,

max value of $i = 2^k$

and so

$$2^k = n$$

$$\Rightarrow \log_2 2^k = \log_2 n$$

$$\Rightarrow \boxed{k = \log_2 n}$$

$k \rightarrow$ লুপটা কতবার চলবে

```
for (i=1; i<=n; i = i*7){
        print → ok
}
```

উত্তর $O(\log_7 n)$

non-nested loop এর জন্য Order of what?

```
for (i=1; i<=5; i++){
    print -> ok
}

for (j=1; j<=3; j++){
    print -> ok
}
```

nested না, তাই,
total loop চলবে 5+3=8 বার

i j

so এই ক্ষেত্রে

$O(5) + O(3)$

$\rightarrow O(5)$

$\rightarrow O(5 \times 1)$

$\rightarrow O(1)$

[redacted]

nested loop এর জন্য

```
for (i=1; i<=5; i++){
    for (j=1; j<=3; j++){
        print -> ok
    }
}
```

total execution হবে
$5 \times 3 = 15$ বার

i j

এই ক্ষেত্রে জন্য

$O(5) \times O(3)$

$\rightarrow O(5 \times 3)$

$\rightarrow O(15)$

$\rightarrow O(15 \times 1)$

$\rightarrow O(1)$

যদি প্রথম loop এ

. . . $i<=5n$; . . . হলে

এবং

দ্বিতীয় লুপে

. . . $j<=3n$; . . . হলে

তখন,

$\rightarrow O(5n)+O(3n)$

$\rightarrow O(5n)$

$\rightarrow O(5\times n)$

$\rightarrow O(n)$

প্রথম loop এ,

. . $i<=5n$; . . .

এবং

দ্বিতীয় লুপে

$j<=3n$ হলে

↳ তখন,

$O(5n)\times O(3n)$

$\rightarrow O(5n * 3n)$

$\rightarrow O(15n^2)$

$\rightarrow O(n^2)$

→ confused

Que: non nested এ $O(n)$ এবং $O(n^2)$ হওয়া সত্ত্বেও both $O(1)$ কীভাবে?

ans: direct value থাকলে তো সেটা $O(1)$. যদি range variable n হয় তবেই সাপেক্ষে comparison

$\Rightarrow$ $O(1)$ হলে বোঝায় constant time time complexity

$\boxed{\text{Que}}$ নিচের code এর time complexity বের করো)

```
for (i=1, i<=4n; i=i+9){

    for (j=1, j<=n; j=j+5){

        for (k=1, k<=40; k=k+5){
            print→ok

        }

        for (m=1, m<=3n; m=m+4){

            print→ok

        }
    }

}
```

[redacted]

soln:

$$O\left(\frac{4n}{9}\right) \times O\left(\frac{n}{5}\right) \times \left\{O\left(\frac{40}{5}\right) + O\left(\frac{3n}{4}\right)\right\}$$

$$O\left(\frac{4}{9} \times n\right) \times O\left(\frac{1}{5} \times n\right) \times \left\{O(8) + O\left(\frac{3}{4} \times n\right)\right\}$$

$$O(n) \times O(n) \times \{O(8 \times 1) + O(n)\}$$

$$O(n) \times O(n) \times \{O(1) + O(n)\}$$

$$O(n) \times O(n) \times \{O(n)\}$$

$\Rightarrow$ finally $O(n \times n \times n)$

$\Rightarrow O(n^3)$

[redacted]

$\boxed{\text{Que}}$ নিচের code এর time complexity বের করো)

```
for (i=1, i<=4n; i=i+9){
    for (j=1, j<=n; j=j*5){
        for (k=1, k<=40; k=k+5){
            print→ok
        }
        for (m=1, m<=3n; m=m+4){
            print→ok
        }
    }
}
```

$$O\left(\frac{4n}{9}\right) \times O(\log_5 n) \times \left\{ O\left(\frac{40}{5}\right) + O\left(\frac{3n}{4}\right) \right\}$$

$$O(n) \times O(\log_5 n) \times \{ O(1) + O(n) \}$$

$$O(n) \times O(\log_5 n) \times O(n)$$

$$\Rightarrow O(n * \log_5 n * n)$$

$$\Rightarrow O(n^2 \log_5 n)$$

Practice question on time complexity

①

```
for (k=n, k>=1; k=k-5){
        print → ok
    }
```

②

```
for (k=n, k>=1; k= k/7){
        print → ok
    }
```

solution of 1:

so,

$n-5k=1$

$5k=n-1$

$k=\frac{n}{5}-\frac{1}{5}$

using simplifica-tion rules

$k=\frac{n}{5}-\frac{1}{5}$

$=\frac{n}{5}$

$k=n$

step	k
0	$n = n-5\times 0$
1	$n-5 = n-5\times 1$
2	$n-10 = n-5\times 2$
3	$n-15 = n-5\times 3$
⋮	⋮
k	$n-5k$

so

$$\boxed{k=n \Rightarrow O(n)}$$

solution to que 2

step	i
0	$n = n/7^0$
1	$n/7 = n/7^1$
2	$n/49 = n/7^2$
3	$n/343 = n/7^3$
$\vdots$	$\vdots$
k	$n/7^k$

so,
$$\frac{n}{7^k} = 1$$

$$n = 7^k$$

$$\log_7 n = \log_7 7^k$$

$$\log_7 n = k$$

$$\Rightarrow \boxed{k = \log_7 n}$$

so iterator কোনো একটা নির্দিষ্ট সংখ্যার গুণিতক হিসেবে বাড়ুক বা কমুক, time complexity same থাকে। যেমন:

```
for (i=0; i<=n; i=i*7){
        print→ok
        }

for (i=n; i>=1; i=i/7){
        print→ok
        }
```

both এর $O(\log_7 n)$

যেই constant আকারে বাড়তেছে বা কমতেছে সেটা log এর base হবে।