Single_NestedLoop_Complexity

Monday, August 2, 2021 3:11 PM

Time Complexity → no of key steps in terms of input size n

```
print("Hello")   # 1
for (i = 1; i < n; i++)
    print(i)     # 2 . n → k
print("Bye")     # 1
```

Iteration	1	2	3	4	...	k	k+1
i	1	2	3	4	...	k	(k+1)

Condition false

1, 2, 3, 4 ... n

n+1

i = n → false

k+1 = n

k = n-1

T(n) = k

T(n) = n-1

$T(n) = 1 + 1 + n + n + n + 1$

$= 3n + 3$ → constant

$= O(n)$

$f(n) = 3n$

$g(n) = n$

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

$g(n) \geq c \cdot f(n)$

$1n \leq 5n \leq 6n \Rightarrow O(n)$

```
i = 1          # 1
While i <= n
DO
    print(i)   # k
    i = i*2    → log2(n)
Done
```

Iteration	1	2	3	4	5	...	k	k+1
Variable i	1 $2^0=1$	2 $2^1=2$	4 $2^2=4$	8 $2^3=8$	16 $2^4=16$	...	2^{k-1}	2^k

(k-1)th

k+1 → loop condition will be false

$i = n$

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

$k \log 2 = \log_2 n$

$k = \log n$

Ajmal

$T(n) = k$ ....(i)

$T(n) = \log(n)$

```
i = n
While i >= 1   Fail
DO
    print(i)   # k
```

k+1

k+1 iterations will make i >= 1 condition false

Anchor can End condition

	1	2	3	4	5	...	k	k+1

Do

print(i) # K

$i = i/2$

Done

iteration

End condition

1	2	3	4	5	...	K	K+1
$i = \frac{n}{2^0}$	$\frac{n}{2} = \frac{n}{2^1}$	$\frac{n}{4} = \frac{n}{2^2}$	$\frac{n}{8} = \frac{n}{2^3}$	$\frac{n}{16} = \frac{n}{2^4}$	...	$\frac{n}{2^{K-1}}$	$\frac{n}{2^K}$

$T(n) = K$

$\boxed{T(n) = \log n}$

$i = 1 \Rightarrow \frac{n}{2^K} = 1 \Rightarrow 2^K = n$

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

$\boxed{K = \log_2(n)}$

```
i = 2
While  i <= n
Do
        print(i) # K
        i = i^2
Done
```

Anchor

iteration

1	2	3	4	5	...	K	K+1
$2 = 2^{2^0}$	$4 = 2^{2^1}$	$16 = 2^{2^2}$	$256 = 2^{2^3}$	$65535 = 2^{2^4}$	...	$2^{2^{K-1}}$	2^{2^K}

i

2^1 2^2 2^4 2^8 2^{16}

2^{2^0} 2^{2^1} 2^{2^2} 2^{2^3} 2^{2^4}

$\boxed{i > n}$ → Anchor

$T(n) = K$

$2^{(2^K)} = n$

$\log_2 2^{(2^K)} = \log_2(n)$

$2^K \log_2(2) = \log_2(n)$

$\log 2^K = \log_2 \log_2 n$

$K \log_2 2 = \log_2 \log_2 n$

$\boxed{K = \log_2 \log_2 n}$

$\boxed{T(n) = O(\log \log n)}$

Question HW

```
i = n
WHILE  i > 1
DO
      print(i)
      i = √i
Done
```

$\log \log n$

Nested loop time Complexity

Independent

```
for (i=1; i<n; i++)
    for (j=1; j<n; j=j*2)
        print(i,j)
```

Dependent

```
for (i=1; i<n; i++)
    for (j=1; j<=i; j++)
        print(i,j)
```

inner loop depends on outer loop variable

```
for (i=1; i<n; i++)
    for (j=1; j<n; j=j*2)      → B
        print(i,j)             Body
```

$B = \log(n)$

$A = n$

A

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

$T(n) = O(A \cdot B)$ → for independent loop

iteration	1	2	3	...	k

$= 2+2+2+2$

$= 2 \times 4$

iteration	1	2	3	...	k
i	1	2	3	...	n → n
j	log(n)	log(n)	log(n)	...	log(n)

$T(n) = \log(n) + \log(n) + \log(n) + \cdots$ (log(n) terms)

$= \log(n)\,(1+1+1+1+\cdots n)$

$= n \log(n)$

```
for (i = 2; i <= n^2; i = i+2)
    for (j = n/2; j <= n; j = j*2)
        for (k = n^2; k >= 2; k = √k)
            print("Hello")
```

$Z = O(n^2)$

$Y = O(1)$

$X = O(\log \log n)$

$T(n) = O(X \cdot Y \cdot Z) \Rightarrow O(n^2 \cdot 1 \cdot \log \log n) \rightarrow O(n^2 \log \log n)$

Iteration

1	2	3	4	5	...	z	z+1
i: 2 (2×1)	4 (2×2)	6 (2×3)	8 (2×4)	10 (2×5)	...	2×z	2×(z+1)

$i = 2*z$ when condition will be False

$2 * z = n^2 \Rightarrow z = \frac{n^2}{2}$

$\rightarrow \boxed{z = O(n^2)}$

→ $z = O(n^2)$

1/2

Iteration j

1	2	3	4	5	...	k	k+1
n/2	n	2n	4n	8n	...		
$2^{-1}n$	$2^0 n$	$2^1 n$	$2^2 n$	$2^3 n$		$2^{k-2} n$	

1 - 2 → -1
2 - 2 → 0
3 - 2 → 1
4 - 2 → 2

$$2^k \cdot n = n$$

$$2^k = 1$$

$$\log_2(2^k) = \log(1)$$

$$k = c$$

$k = n^2$; $k >= 2$; $k = (k)^{1/2}$

Iteration k

1	2	3	4	5	...	k	k+1
n^2	n	$n^{1/2}$	$n^{1/4}$	$n^{1/8}$	...	.	
$n^{\frac{1}{2^{-1}}}$	$n^{\frac{1}{2^0}}$	$n^{1/2^1}$	$n^{\frac{1}{2^2}}$	$n^{\frac{1}{2^3}}$		$n^{\frac{1}{2^{k-2}}}$	

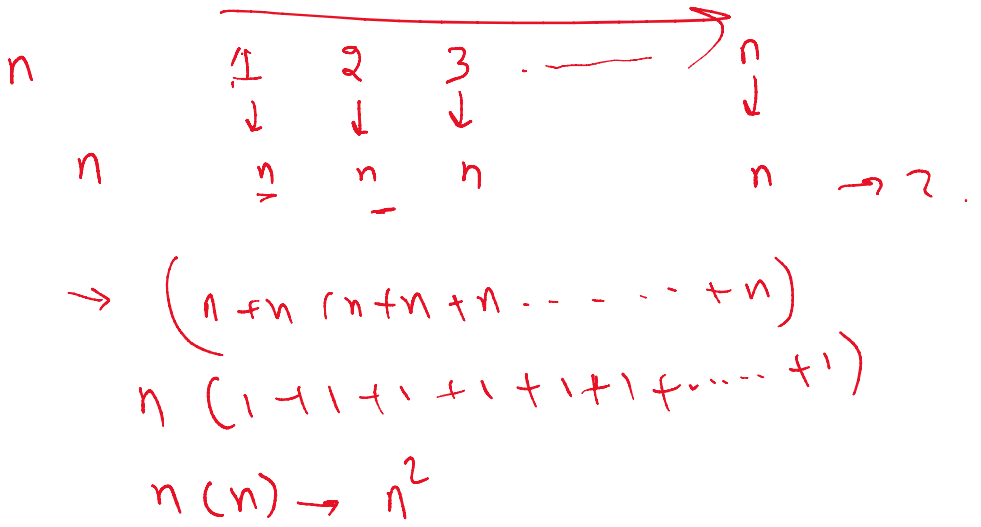
$$n^{\frac{1}{2^k}} = 2 \Rightarrow \frac{1}{2^k}\log(n) = \log(2)$$

$$\Rightarrow 2^k = \log(n)$$

$$\Rightarrow \quad 2^k = \log(n)$$

$$\Rightarrow \quad k \log(2) = \log\log(n)$$

$$= \quad \boxed{y = \log\log}$$




```
for (i=1; i<n; i++) →(n)
    for (j=1; j<n; j++) →(m)
        pri (i,j) # k ] → n
```

2. ⇒

$k = n$

Iteration	1	2	3	4	....	k
i	1	2	3	4	...	n
→ j	n	n	n	n	n	n
	m	m	m	m	m	m

$(n+n+n+\cdots+n) \Rightarrow n(1+1+1+\cdots+1)$ (nk terms)

$(m+m+m+\cdots+m)$ (n terms) $\Rightarrow m(1+1+1+\cdots+1)$ (n terms)

$\Rightarrow n^2$

$\Rightarrow m \cdot n$

→ for ($i = n$; $i > 1$; $i = i - n/2$) → $O(1)$

iteration	1	2	3
i	n	n/2	0 ↑ False

for ($j = 1$; $j <= n^2$; $j = j + 10$) → $O(n^2)$

iteration	1	2	3	4	...	y	n^2
j	1	11	21	31	...		

$1 + 10 \times y = n^2$

$y = n^2$

for ($k = n^{10}$; $k > 1$; $k = k/2$) → $O(\log(n))$

print(i, j, k) # K

iteration	1	2	3	4	5	... k
k	n^{10}	$n^{10}/2$	$n^{10}/4$	$n^{10}/8$	$n^{10}/16$	...

$T(n) = O(n^2 \log(n))$

$\frac{n^{10}}{2^k} = 1 \rightarrow 2^k = n^{10}$

$k \log_2(2) = 10 \log_2 n$

$k = \log_2(n)$

→ Nested Loop

Dependent Loops

for ($i = 1$; $i <= n$; i++)

A ——— B

n n/2

for ($j = 1$; $j <= n$; $j = j + i$) → ($j = 1$; $j <= n$; $j = j + i$)

$j = j + 2$

print(i, j) # K

i	1	2	3	4	...	k
j	1	3	5	7		→

print(i,j) # K

i th	1	2	3	4	...	k
j	1	3	5	7		→

1+2×0 1+2×1 1+2×2 1+2×3 ... 1+2×k

i th	1	2	3	4	...	n	n+1
	n	n/2	n/3	n/4	...	n/n	

1+2k < n
k = n/2

$a_n = a + (n-1)d$
$a_n = 1 + (k-1)3$
$a = 1, d = 3$
$= 1 + 3k - 3$
$= 3k - 2$

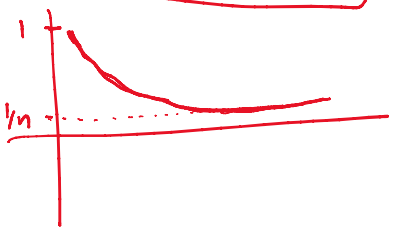
$$T(n) = n + n/2 + n/3 + n/4 + \cdots + n/n$$

$$T(n) = n\left(1 + 1/2 + 1/3 + 1/4 + \cdots + 1/n\right)$$

$\log(n)$

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

$1 + 2 + 3 + 4 + \cdots n \Rightarrow \frac{n(n+1)}{2}$




```
for (i = 1; i <= n; i++)
    for (j = 1; j <= i^2; j++) → 9.
        for (k = 1; k <= n/2; k++)        n/2
            print("Hi")
```

iteration	1	2	3	k
i	1	2	3	n
j	1	4	9	n^2
k	$1 \cdot n/2$	$4 \cdot n/2$	$9 \cdot n/2$	$n^2 \cdot n/2$

$$T(n) = \frac{n}{2} + 4 \cdot \frac{n}{2} + 9 \cdot \frac{n}{2} + \cdots + n^2 \cdot n/2$$

$$= \frac{n}{2}\left(1 + 4 + 9 + \cdots + n^2\right)$$

$$T(n) = \frac{n}{2}\left(\frac{n \cdot (n+1) \cdot (2n+1)}{6}\right)$$

$$T(n) = \frac{n}{2}\left(\frac{\quad}{6}\right)$$

$$= \frac{n}{12}\left((n^2+n)\cdot(2n+1)\right)$$

$$= \frac{n}{12}(2n^3+n^2+2n^2+n)$$

$$= \frac{n}{12}(2n^3+3n^2+n)$$

$$= \frac{2n^4+3n^3+n^2}{12}$$

$$= n^4 + n^3 + n$$

$n^4 >> n^3 >> n$

n^4 ↑ significant

$$\boxed{T(n) = O(n^4)}$$

Recursive Algorithms →

(*) How write a recursive eqn

→ (1) Back Substitution

→ (2) Recursion Tree Method

→ (3) Masters theorm

→ Questions

→ R ie Function

```
def fact(n):
    if n > 1:
        return n * fact(n-1)
    return 1
```

T(n) — Base Condition: C

T(n) → T(n-1) + C if n > 1
T(n) → C if n = 1

① Back Substitution

problem

$$T(n) \rightarrow \begin{cases} T(n-1)+C & \text{if } n>1 \\ C & \text{if } n=1 \end{cases} \rightarrow \boxed{T(1) = C}$$

$$T(n) = C + T(n-1) \quad \text{--- (1)}$$

$$T(n-1) = C + T(n-2) \quad \text{--- (2)}$$

$$T(n-2) = C + T(n-3) \quad \text{--- (3)}$$

$$T(n-3) = C + T(n-4) \quad \text{--- (4)}$$

(2) in (1)

$$T(n) = C + C + T(n-2) \Rightarrow 2C + T(n-2)$$

(3) in

$$\ldots + T(n-3)$$

③ in

$T(n) = c + c + c + T(n-3) \Rightarrow 3c + T(n-3)$

$$\boxed{T(n) = k \cdot c + T(n-k)}$$

we know $T(1) = c$

So $(n-k) = 1$

$$\boxed{k = n-1}$$

$$T(n) = (n-1) \cdot c + T(n-n+1)$$

$$T(n) = (n-1) \cdot c + T(1)$$

$$T(n) = (n-1) \cdot c + c$$

$$\boxed{T(n) = O(n)}$$

→ Time Complexity for Recursive functions

```
def fact(n):
    if n>1:          # c
        return n*fact(n-1)   # T(n-1)
```

fcn → T(n)

f(n-1) → T(n-1)

T(n) → $\begin{cases} c & \text{if } n = 1 \\ T(n-1) & \text{if } n > 1 \end{cases}$

return n * [cf n-1] ... → T(n-1) if n > 1

return 1

① Back substitution method

$T(n-1) = c + T(n-1-1)$

$T(n-1) = c + T(n-2)$ → (2)

$T(n-2) = c + T(n-3)$ → (3)

...

$T(n) = c + T(n-1)$

(2) in this eqn

$T(n) = c + c + T(n-2) \Rightarrow 2c + T(n-2)$

(3) in this eqn

$T(n) = 2c + c + T(n-3) \Rightarrow 3c + T(n-3)$

$T(n) = 4C + T(n-4)$

After k calling we will get $T(1)$

$$\boxed{T(n) = kC + T(n-(k+1))} \rightarrow (5)$$

if $n=1$ then $T(1) = C$

$$T(n-(k+1)) = T(1)$$

$$n-(k+1) = 1$$

$$\boxed{k = n-2}$$

$\rightarrow T(n-(n-2+1))$

$\rightarrow T(n-(n-1))$

$\rightarrow T(n-n+1)$

$\rightarrow T(1) \Rightarrow C$

Replace k in eq^n (5)

$$\boxed{k = n-2}$$

$$T(n) = k \cdot C + T(n-(k+1))$$

$$T(n) = (n-2) \cdot C + T(n-(n-2+1))$$

$$T(n) = nC - 2C + T(1)$$

$$T(n) = nc - 2c + T(1)$$

$$T(n) = nc - 2c + c$$

$$\boxed{T(n) = O(n)}$$

```
def func(n):
    if n>1:
        for i=1 to n:
            print(i)
    func(n-1)
    return 1
```

n → for i=1 to n: print(i)

T(n-1) → func(n-1)

C → return 1

$$T(n) \rightarrow \begin{cases} n + T(n-1) & ; n > 1 \\ C & ; n = 1 \end{cases}$$

$$T(n) = n + T(n-1) \quad \rightarrow (1)$$

$$T(n-1) = (n-1) + T(n-2) \quad \rightarrow (2)$$

$$T(n-2) = (n-2) + T(n-3) \quad \rightarrow (3)$$

eqn (2) subst in eqn (1)

$$T(n) = n + (n-1) + T(n-2) \quad \cdots (4)$$

eqn (3) subst in eqn (4)

$$T(n) = n + (n-1) + (n-2) + T(n-3) \quad \cdots (5)$$

$T \;) = n \quad n \quad -) \; (\; 3)+ \cdots n \;) - (\quad 2+1)) \rightarrow (6)$

we know $T(n) = C$ if $n = 1$

$$T(\underline{n-(k+1)}) = T(\underline{1})$$

$$n-(k+1) = 1 \Rightarrow \boxed{k = n-2} \rightarrow (7)$$

Substitute value of k from eq^n (7) to eq^n (6) we get

$$T(n) = n + (n-1) + (n-2) + (n-3) + \cdots + (n-k) + T(n-(k+1))$$

$$T(n) = n + (n-1) + (n-2) + (n-3) + \cdots + (\cancel{n}-(\cancel{n}-2)) + T(n-(n-2+1))$$

$$T(n) = \underbrace{n + (n-1) + (n-2) + (n-3) + \cdots + 2}_{} + \cancel{T(1)}^{\,C}$$

$$T(n) = \frac{n \cdot (n+1)}{2} + C$$

$$\boxed{T(n) = O(n^2)}$$

---

def $f(n)$:

$f(n) = T(n)$

$f(n/2) = T(n/2)$

def T(n):

if n >= 1:

return f(n/2) + f(n/2)

v_1 v_2

return 1

$T(n) = f(n/2)$

$f(n/2) = T(n/2)$

$$T(n) = \begin{cases} c & \text{if } n = \\ c + 2T(\quad 2) \end{cases}$$

$$T(n) = c + 2T(n/2) \quad \text{---} \quad (1)$$

$$T(n - c + 2T(\tfrac{n}{4})) \quad \text{---} \quad 2$$

$$T(\quad) = c\ 2T(\tfrac{n}{8}) \qquad 3$$

$T(n) = c + 2T(n/2) \rightarrow (1) \Rightarrow 2^0 c + 2^1 T(\frac{n}{2^1})$

Substitute value of T(n/2) from eqn (2) to eqn (1)

$T(n) = c + (c + 2T(n/4))$

$T(n \quad c + c - 4 \quad) \Rightarrow 2^0 c + 2^1 c + 2^2 T(\frac{n}{2^2})$

Substitute value of T(n/4) from eqn (3) to eqn 4

$T(n) = c + 2c + 4 \cdot (c + 2 \cdot T(n/8))$

$= c + 2c + 4c + 8T(n/8) \Rightarrow 2^0 c + 2^1 c + 2^2 c + 2^3 T(\frac{n}{2^3})$

$$T\left(\frac{n}{8}\right) = C + 2T\left(\frac{n}{16}\right)$$

$$= C + 2C + 4C + 8\cdot\left(C + 2T\left(\frac{n}{16}\right)\right)$$

$$= C + 2C + 4C + 8C - 16T\left(\frac{n}{?}\right)$$

$$- {}^{\circ}C + 2C - 2^3C + 2^4 T\left(\frac{n}{2^4}\right)$$

Let a we we 'll k th r n-1 —)

$$\Rightarrow 2^0C + 2^1C + 2^2C + 2^3C - \cdots + 2^{k-1}C + 2^k T\left(\frac{n}{2^k}\right)$$

$\rightarrow T\left(\frac{n}{2^k}\right) = T(1)$ then $\frac{n}{2^k} = 1 \Rightarrow k = \log(n)$

eq^n will be $\Rightarrow C\left[2^0 + 2^1 + 2^2 + 2^3 + \cdots \cdots 2^{k-1}\right] + T(1)$

$\Rightarrow C\left[1+2+4+8+16+\cdots\cdots 2^{k-1}\right] + C$

$(2n-1)$ so $T(n) = C\cdot(2n-1) + C =$

Assignment → $\Rightarrow$ $\boxed{T(n) = O(n)}$

** def fab(n):

if n == 1:

return 0

if n == 2:

return 1

return fab(n-1) + fab(n-2)

$T(n-1)$ $T(n-2)$

$$T(n) = \begin{cases} C & \text{if } n = 1 \text{ or } n = 2 \\ C + T(n-1) + T(n-2) \end{cases}$$

$T(n) = C + T(n-1) + T(n-2)$

Ass $(n-1) \approx 1(\_\ )$

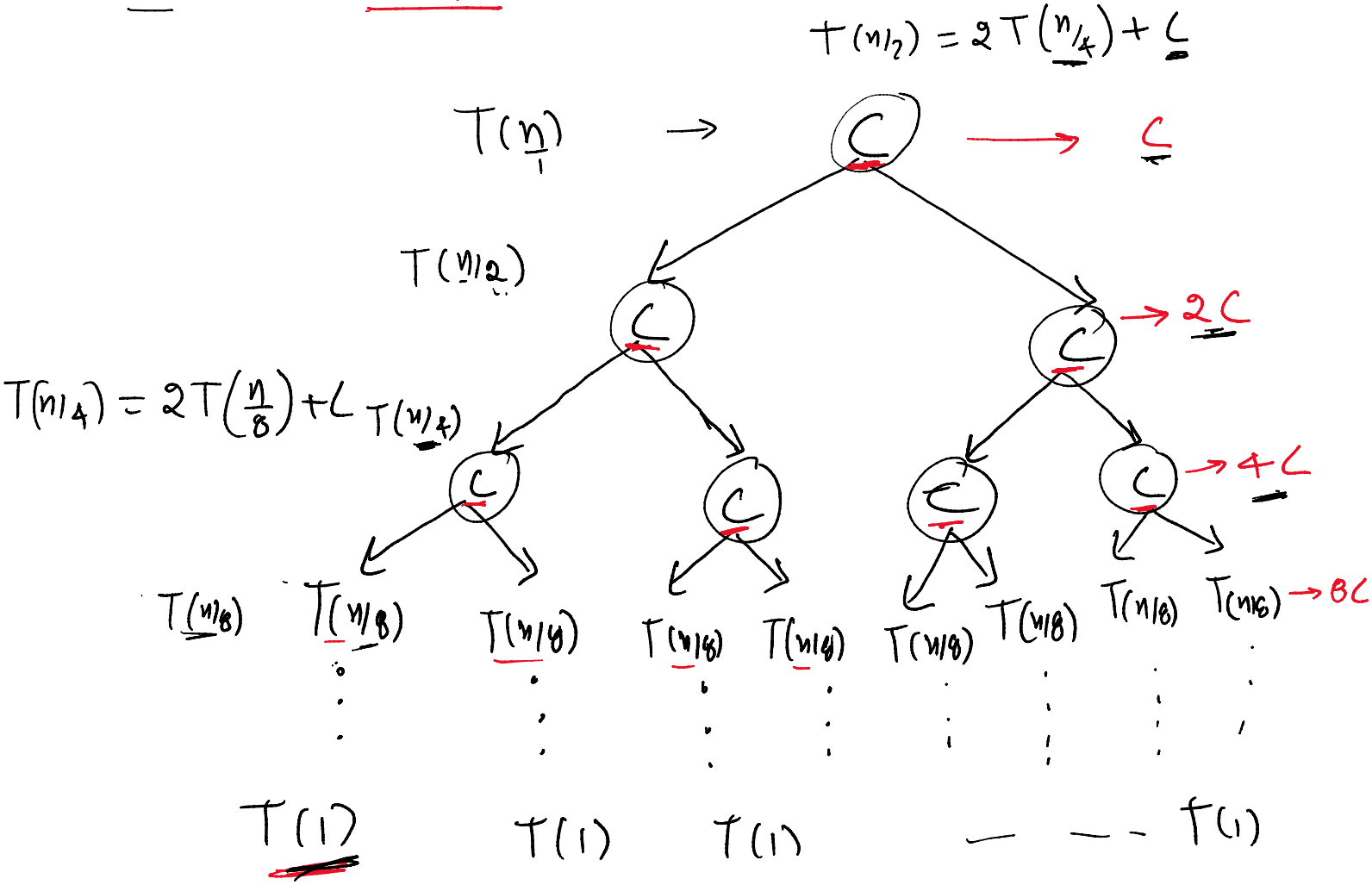
$T(n) = C + T(n-2) + T(n-2)$

$\boxed{T(n) = C + 2T(n-2)}$ →

② Recursion Tree method

$$T(n) = \begin{cases} 2T(n/2) + c & ; \ n > 1 \\ c & ; \ n = 1 \end{cases}$$

$$T(n) = 2T(n/2) + c$$



$$T\left(\frac{n}{n}\right) = T(1)$$

$$n = 2^k$$

$$T\left(\frac{n}{2^k}\right) = T(1)$$

$C + 2C + 4C + 8C + \cdots \longrightarrow + nC$

$C + 2C + 4C + 8C + \cdots \longrightarrow + 2^k \cdot C$

$C\left(1 + 2 + 4 + 8 + \cdots \longrightarrow + 2^k\right) \rightarrow$ G.P

$$\frac{a \cdot (r^n - 1)}{r - 1} \quad ; \; a = 1, \; r = 2, \; n = k$$

$$\frac{1 \cdot (2^k - 1)}{1} \rightarrow 2^k - 1 \rightarrow 2^{\log(n)}$$

$n = 2^k \rightarrow k = \log(n) \longrightarrow \Rightarrow n \rightarrow O(n)$

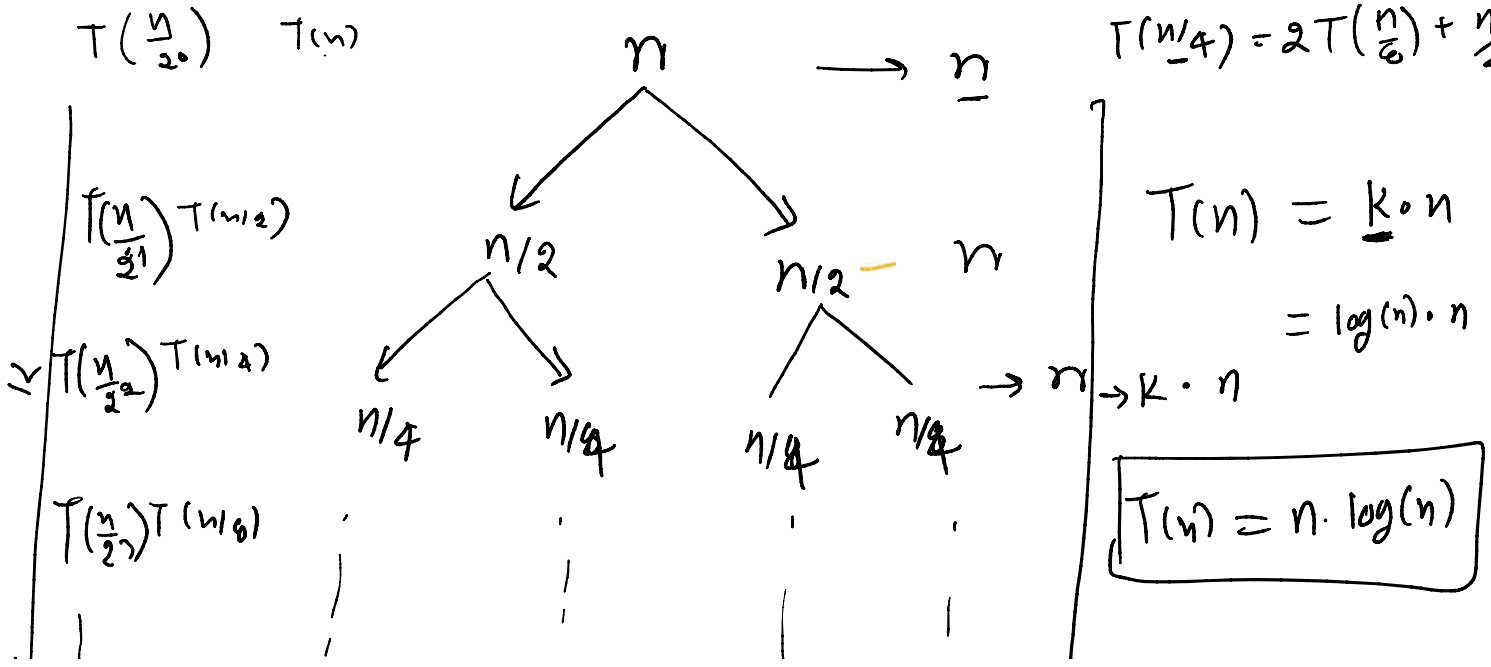
$$\rightarrow T(n) = 2T(n/2) + n \quad ; \; n > 1$$

$$C \quad ; \; n = 1$$

$T(n) = 2T(n/2) + n$

$T(n/2) = 2T(n/4) + \frac{n}{2}$

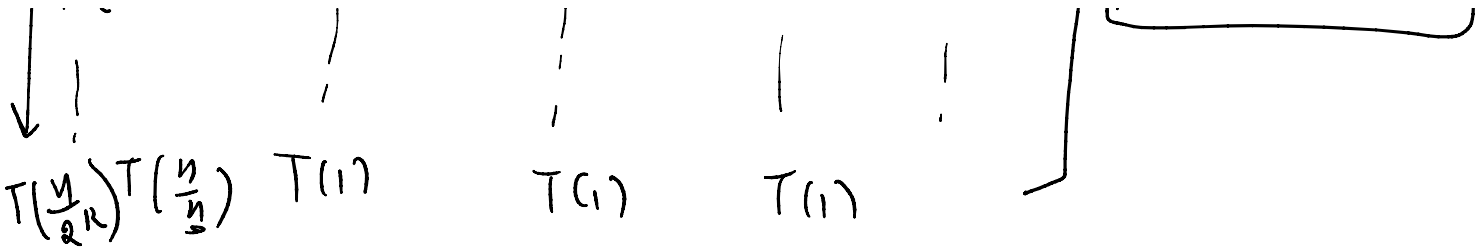
$T(n/4) = 2T(\frac{n}{8}) + \frac{n}{4}$



$T(n) = k \cdot n$

$= \log(n) \cdot n$

$T(n) = n \cdot \log(n)$



$$T\left(\frac{n}{2^k}\right) = T(1)$$

$$\frac{n}{2^k} = 1 \Rightarrow n = 2^k$$

$$k = \log(n)$$