Recursion tree method for solving recurrence relation

- Solve the following recurrence relation using recursion tree.

$T(n) = 2T\left(\frac{n}{2}\right) + n$ → to solve a problem of size n, we must solve two subproblems of size $\frac{n}{2}$ and perform n amount additional work.

Level-0 $\quad n \quad \rightarrow \quad n$

Level-1 $\quad \frac{n}{2} \quad \frac{n}{2} \quad \rightarrow 2 \cdot \frac{n}{2} = n$

Level-2 $\quad \frac{n}{4} \quad \frac{n}{4} \quad \frac{n}{4} \quad \frac{n}{4} \quad \rightarrow 4 \cdot \frac{n}{4} = n$

Level-K $\quad \frac{n}{2^K} \quad \frac{n}{2^K} \quad \cdots \quad \frac{n}{2^K} \quad \frac{n}{2^K} \rightarrow n$

$= (K+1) \cdot n = (\log n + 1) \cdot n$

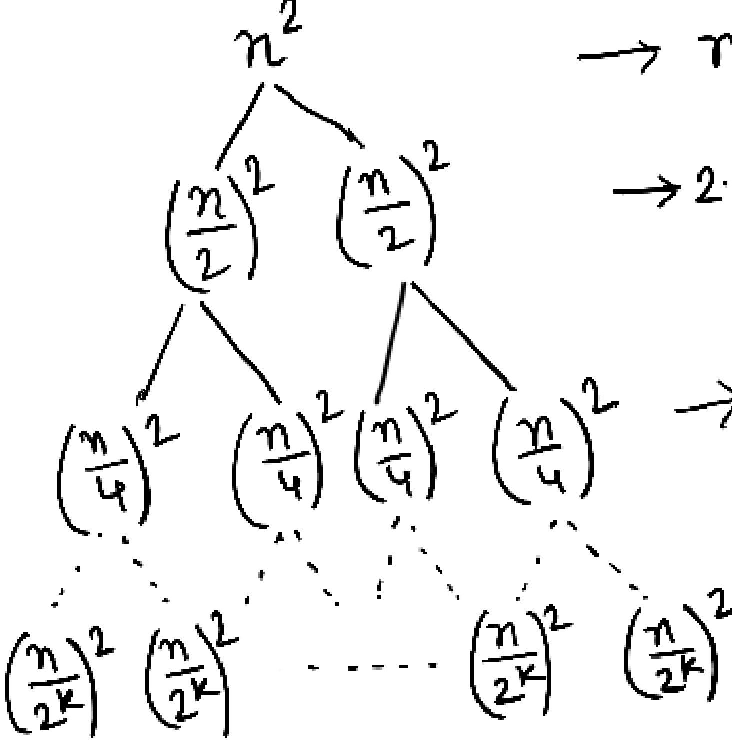
$= O(n \log n)$ [Ans]

Leaves are labelled with base cost $T(0)$

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

• Solve the following recurrence relation using recursion tree.

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



$\rightarrow n^2$

$\rightarrow 2\cdot\left(\frac{n}{2}\right)^2 = \frac{n^2}{2}$

$\rightarrow 4\cdot\left(\frac{n}{4}\right)^2 = \frac{n^2}{4}$

$\rightarrow 2^k\cdot\left(\frac{n}{2^k}\right)^2 = \frac{n^2}{2^k}$

$$n^2\left[1 + \frac{1}{2} + \frac{1}{4} + \cdots + \frac{1}{2^k}\right]$$

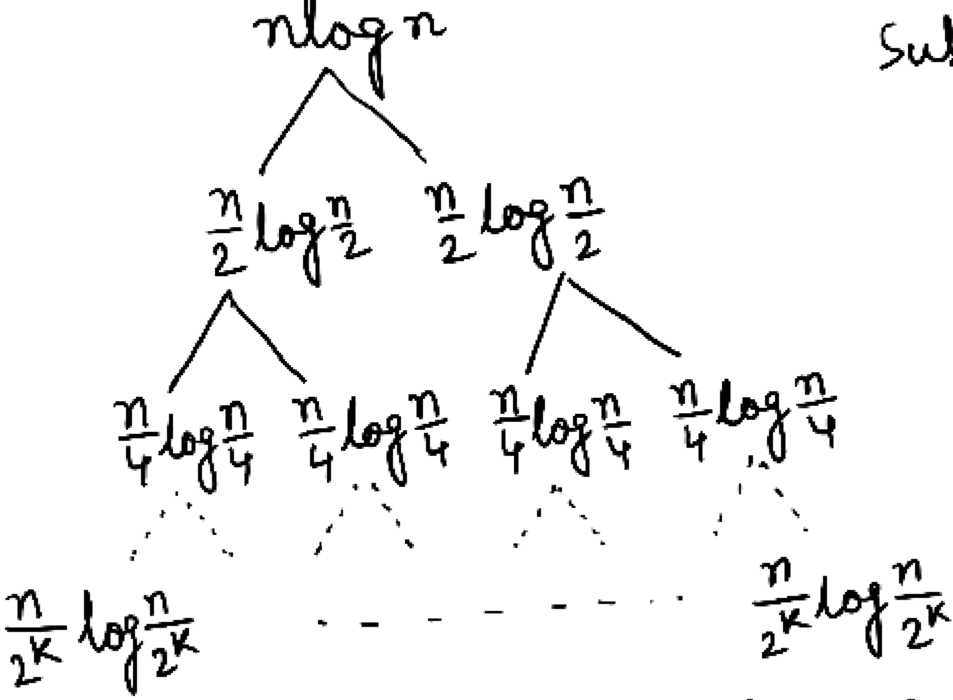
$$\leq n^2 \cdot \sum_{i=0}^{\infty} \frac{1}{2^i}$$

$$\leq n^2 \cdot \frac{1}{1-\frac{1}{2}} \leq 2n^2$$

$$O(n^2)$$

- ## Recursion tree method

$$T(n) = 2T\left(\frac{n}{2}\right) + n\log n$$



Subproblem size $= n \rightarrow n/2 \rightarrow n/4 \rightarrow \cdots \rightarrow \frac{n}{2^K}$

$$\frac{n}{2^K} = 1 \Downarrow K = \log n$$

$$\text{Sum} = n\log n + 2\cdot\frac{n}{2}\log\frac{n}{2} + 4\cdot\frac{n}{4}\log\frac{n}{4} + \cdots + 2^K\cdot\frac{n}{2^K}\log\frac{n}{2^K}$$

$$= n\log n + n\log\frac{n}{2} + n\log\frac{n}{4} + \cdots + n\log\frac{n}{2^K}$$

$$= n\log n + n(\log n - 1) + n(\log n - 2) + \cdots + n(\log n - K)$$

$$= [n\log n + n\log n + \cdots (K+1)\text{ times}] - (n + 2n + 3n + \cdots + Kn)$$

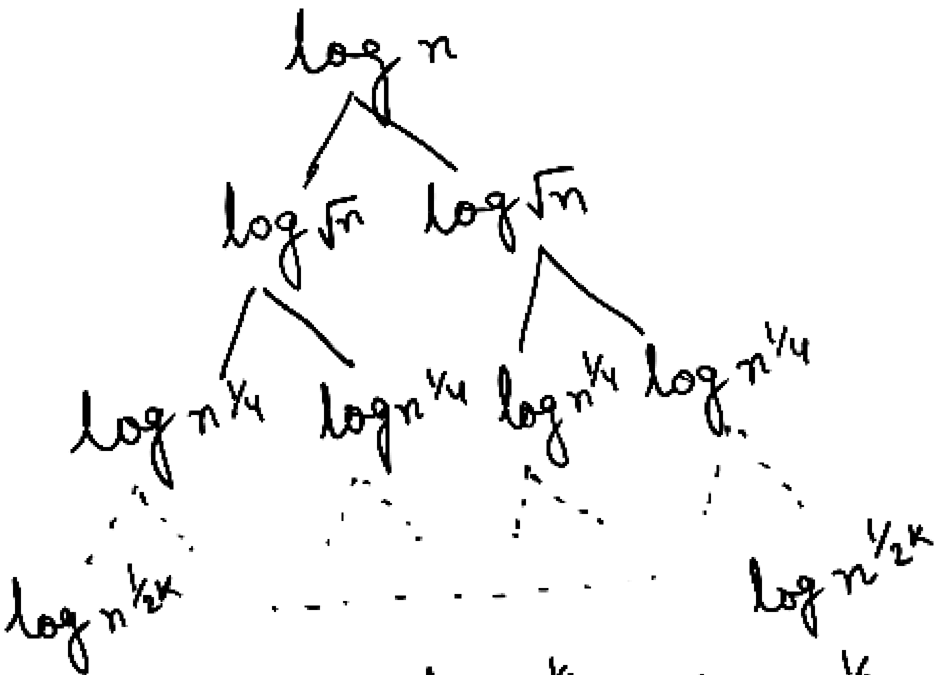
$$= [n\log n + n\log n + \cdots (\log n + 1)\text{ times}] - n(1 + 2 + 3 + \cdots + \log n)$$

$$= n(\log n)^2 - n\cdot\frac{\log n\cdot(\log n + 1)}{2}$$ [ignoring +1 term]

$$O[n(\log n)^2]$$ (Ans.)

- **Recursion tree method:**

$$T(n) = \underline{\underline{2}}T(\sqrt{n}) + \log n$$



$$\text{Sum} = \log n + 2\log n^{1/2} + 4\log n^{1/4} + \dots + 2^k \log n^{1/2^k}$$

$$= \log n + \log n + \log n + \dots + \log n$$

$(k+1)$ terms

$$= (k+1)\log n$$

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

$$= \log n \cdot \log\log n + \log n$$

$$O(\underline{\underline{\log n \cdot \log\log n}})$$

(Ans.)

Subproblem size

$$n \rightarrow n^{1/2} \rightarrow n^{1/4} \rightarrow \dots \rightarrow n^{1/2^k}$$

$$n^{1/2^k} = 2$$

$$\log n^{1/2^k} = \log 2$$

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

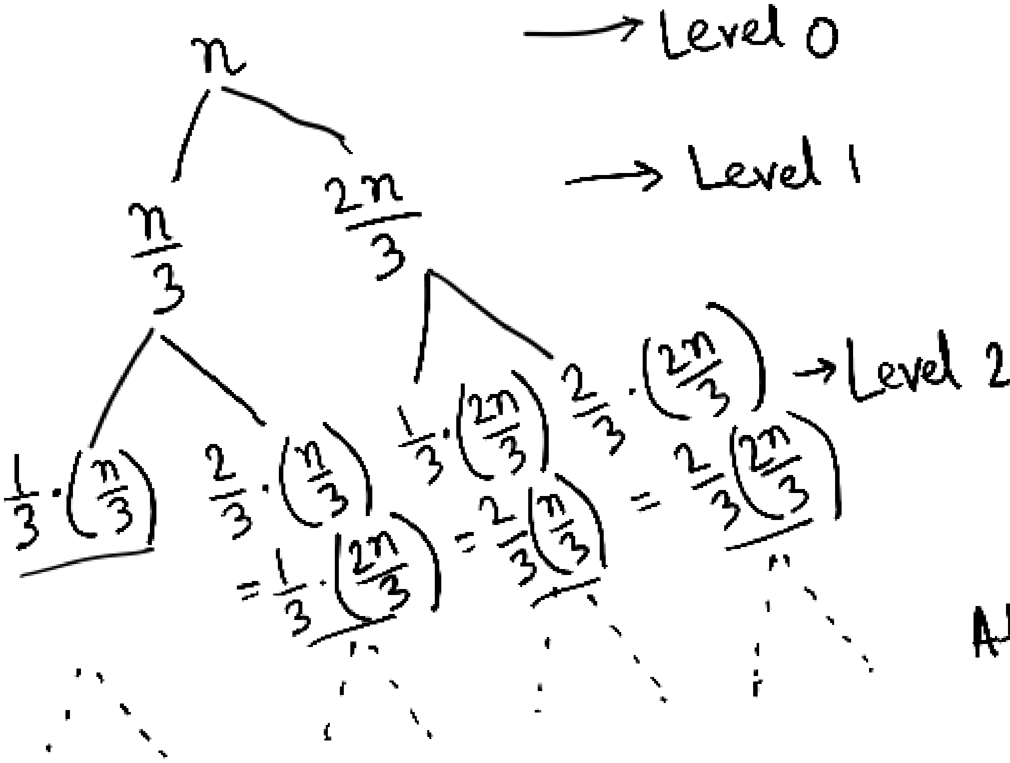
$$\log n = 2^k$$

$$n = 2^{2^k}$$

$$k = \log\log n$$

- <u>Recursion tree method:</u>

$$T(n) = T\left(\frac{n}{3}\right) + T\left(\frac{2n}{3}\right) + \underline{\underline{n}}$$



$$n = \left(\frac{1}{3}+\frac{2}{3}\right)^0 \cdot n$$

$$\left(\frac{1}{3}+\frac{2}{3}\right)^1 \cdot n$$

$$\left(\frac{1}{3}+\frac{2}{3}\right)^2 \cdot n$$

At Level K, $\left(\frac{1}{3}+\frac{2}{3}\right)^K \cdot n$

$$\text{Sum} = \sum_{i=0}^{K} \left(\frac{1}{3}+\frac{2}{3}\right)^i \cdot n$$

$$= \sum_{i=0}^{K} 1^i \cdot n = \sum_{i=0}^{K} n = (K+1)\cdot n$$

$$= (\log n + 1)\cdot n$$

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

$$O(n \log n) \text{ (Ans.)}$$

Master Theorem - Another method for solving recurrence relation

Consider an equation of the form as

$$T(n) = aT\left(\frac{n}{b}\right) + O(n^k) \text{ where } a \geq 1 \text{ and } b > 1$$

Rewriting it as $T(n) = aT\left(\frac{n}{b}\right) + n^k$

Putting $n = b^m$, $T(b^m) = aT(b^{m-1}) + (b^m)^k$

Dividing both side by a^m,

$$\frac{T(b^m)}{a^m} = \frac{T(b^{m-1})}{a^{m-1}} + \left(\frac{b^k}{a}\right)^m$$

$$\frac{T(b^{m-1})}{a^{m-1}} = \frac{T(b^{m-2})}{a^{m-2}} + \left(\frac{b^k}{a}\right)^{m-1}$$

$$\vdots \qquad \vdots \qquad \vdots$$

$$\frac{T(b)}{a} = \frac{T(b^0)}{a^0} + \left(\frac{b^k}{a}\right)$$

From these equations, we get,

$$\frac{T(b^m)}{a^m} = \frac{T(b^0)}{a^0} + \left(\frac{b^k}{a}\right)^m + \left(\frac{b^k}{a}\right)^{m-1} + \cdots + \left(\frac{b^k}{a}\right)$$

$$\frac{T(b^m)}{a^m} = \sum_{i=0}^{m} \left(\frac{b^k}{a}\right)^i \quad \left[\because \frac{T(b^0)}{a^0} = 1 = \left(\frac{b^k}{a}\right)^0\right]$$

$$\frac{T(n)}{a^m} = \sum_{i=0}^{m} \left(\frac{b^k}{a}\right)^i \quad [\because n = b^m]$$

$$T(n) = a^m \sum_{i=0}^{m} \left(\frac{b^k}{a}\right)^i$$

Now, Case-1 : $\boxed{a=b^k}$

$$T(n) = (b^k)^m \sum_{i=0}^{m} 1 = (b^m)^k \cdot (m+1)$$

$$= (m+1) \cdot n^k \quad [\because n = b^m]$$

$$= (\log_b n + 1) \cdot n^k \quad [\because m = \log_b n]$$

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

Case-2 : $\boxed{a<b^k}$ $\quad T(n) = a^m \sum_{i=0}^{m} \left(\frac{b^k}{a}\right)^i$

Common ratio $\left(\frac{b^k}{a}\right) > 1$, So,

$$T(n) = a^m \cdot \frac{\left(\frac{b^k}{a}\right)^{m+1} - 1}{\left(\frac{b^k}{a} - 1\right)}$$

$$= a^m \cdot \left(\frac{b^k}{a}\right)^m \quad [\text{Neglecting } -1 \text{ terms}]$$

$$= (b^k)^m = (b^m)^k = n^k \quad [\because n = b^m]$$

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

Case-3 : $\boxed{a>b^k}$ $\quad T(n) = a^m \sum_{i=0}^{m} \left(\frac{b^k}{a}\right)^i$

As $\sum_{i=0}^{m} \left(\frac{b^k}{a}\right)^i \leq \sum_{i=0}^{\infty} \left(\frac{b^k}{a}\right)^i$, so, $T(n) \leq a^m \sum_{i=0}^{\infty} \left(\frac{b^k}{a}\right)^i$

As common ratio $\left(\frac{b^k}{a}\right)$ is <1, So $T(n) \leq a^m \cdot \frac{1}{1-\frac{b^k}{a}}$

$$\leq a^m \quad \left[\text{Neglecting } \frac{1}{1-\frac{b^k}{a}}\right]$$

$$T(n) = O(a^m) = O(a^{\log_b n}) = \boxed{O(n^{\log_b a})}$$

$$T(n) = \begin{cases} O(n^K), & \text{if } a < b^K \\ O(n^K \log_b n), & \text{if } a = b^K \\ O(n^{\log_b a}), & \text{if } a > b^K \end{cases}$$

$T(n) = aT\left(\frac{n}{b}\right) + n^K$

General form of Master Theorem:

Let $a \geq 1$ and $b > 1$ be constants and $f(n)$ be a function

$$T(n) = a\,T\left(\frac{n}{b}\right) + f(n)$$

Then we have

i) If $f(n) \in O(n^{\log_b a - \epsilon})$ for some constant $\epsilon > 0$ then $T(n) = \Theta(n^{\log_b a})$

ii) If $f(n) \in \Theta(n^{\log_b a})$ then $T(n) = \Theta(n^{\log_b a} \log n)$

iii) If $f(n) \in \Omega(n^{\log_b a + \epsilon})$ for some constant $\epsilon > 0$ and if $a f\left(\frac{n}{b}\right) \leq c f(n)$ for some constant $c < 1$ and sufficiently large n, $T(n) = \Theta(f(n))$

Replacing $\frac{n}{b}$ with $\left\lfloor \frac{n}{b} \right\rfloor$ or $\left\lceil \frac{n}{b} \right\rceil$ does not change the result.

- Using Master theorem, solve the following recurrence relation.

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

comparing it with $T(n) = aT\left(\frac{n}{b}\right) + O(n^K)$

$a = 1, b = 2, K = 0$, so $a = b^K$

$$T(n) = O(n^K \log_b n) = O(n^0 \log_2 n) = \underline{\underline{O(\log n)}}$$

(Ans)

• Solv

Using Master theorem, solve the following recurrence relation.

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

Comparing it with $T(n) = aT\left(\frac{n}{b}\right) + f(n)$

we get, $a=4$, $b=2$, $f(n) = n$

Checking 1^{st} condition, $f(n) \in O\left(n^{\log_b a - \epsilon}\right)$

$$n \in O\left(n^{\log_2 4 - \epsilon}\right)$$

$$n \in O\left(n^{2-\epsilon}\right), \text{true}$$

So, $T(n) = \Theta\left(n^{\log_b a}\right) = \Theta\left(n^{\log_2 4}\right) = \Theta\left(n^2\right)$ (Ans.)

• Solve $T(n) = 2T\left(\frac{n}{2}\right) + n$ using Master theorem.

Comparing it with $T(n) = aT\left(\frac{n}{b}\right) + f(n)$

we get, $a=2$, $b=2$, $f(n) = n$

Checking 2^{nd} condition, $f(n) \in \Theta\left(n^{\log_b a}\right)$

$$n \in \Theta\left(n^{\log_2 2}\right)$$

$$n \in \Theta(n), \text{true}$$

$T(n) = \Theta\left(n^{\log_b a} \log n\right) = \Theta\left(n^{\log_2 2} \log n\right)$

$= \Theta(n \log n)$ [Ans]

- Solve $T(n) = 3T(\frac{n}{2}) + n^2$ using Master theorem.

Comparing it with $T(n) = aT(\frac{n}{b}) + f(n)$ we get, $a=3, b=2, f(n) = n^2$

Checking 1st condition, $f(n) \in O(n^{\log_b a - \epsilon})$

$$n^2 \in O(n^{\log_2 3 - \epsilon})$$

As $\log_2 3 < 2$ so this is false

Checking 2nd condition, $f(n) \in \Theta(n^{\log_b a})$

$$n^2 \in \Theta(n^{\log_2 3}), \text{false}$$

Checking 3rd condition, $f(n) \in \Omega(n^{\log_b a + \epsilon})$

$$n^2 \in \Omega(n^{\log_2 3 + \epsilon}), \text{true}$$

Checking $af(\frac{n}{b}) \leq cf(n)$

$$3f(\frac{n}{2}) \leq cf(n)$$

$$3 \cdot \frac{n^2}{4} \leq cn^2, \; c<1, \text{ true}$$

$T(n) = \Theta(f(n)) = \Theta(n^2)$ [Ans.]

- Check whether Master theorem is applicable in $T(n) = 4T(\frac{n}{2}) + \frac{n^2}{\log n}$

Comparing it with $T(n) = aT(\frac{n}{b}) + f(n)$, we get, $a=4, b=2, f(n) = \frac{n^2}{\log n}$

Checking 1^{st} condition, $f(n) \in O(n^{\log_b a - \epsilon})$

$$\frac{n^2}{\log n} \in O(n^{\log_2 4 - \epsilon})$$

$$\frac{n^2}{\log n} \in O(n^{2-\epsilon})$$

For this to hold we must have,

$$\frac{n^2}{\log n} \leq cn^{2-\epsilon}$$

$$\frac{n^2}{\log n} \leq c \cdot \frac{n^2}{n^\epsilon} \Rightarrow \frac{1}{\log n} \leq c \cdot \frac{1}{n^\epsilon}$$

$$\log n \geq \frac{1}{c} \cdot n^\epsilon \text{, false}$$

so, $f(n) \notin O(n^{\log_b a - \epsilon})$

Checking 2^{nd} condition, $f(n) \in \Theta(n^{\log_b a})$

$$\frac{n^2}{\log n} \in \Theta(n^2) \text{, false}$$

Checking 3^{rd} condition, $f(n) \in \Omega(n^{\log_b a + \epsilon})$

$$\frac{n^2}{\log n} \in \Omega(n^{2+\epsilon})$$

For this to hold, we must have

$$\frac{n^2}{\log n} \geq c \cdot n^{2+\epsilon}$$

$$\frac{n^2}{\log n} \geq c \cdot n^2 \cdot n^\epsilon \Rightarrow \frac{1}{\log n} \geq c \cdot n^\epsilon \text{, false}$$

So, $f(n) \notin \Omega(n^{\log_b a + \epsilon})$

So, Master theorem will not be applicable here.

Amortized Analysis

Algorithm 1 $\longrightarrow$ Best case $O(1)$
Worst case $O(n)$

100 operations $\longrightarrow \begin{Bmatrix} 99 \rightarrow \text{Best case} \\ 1 \rightarrow \text{Worst case} \end{Bmatrix}$

$$\text{Average} = [99 * O(1) + 1 * O(n)] / 100$$
$$= \frac{99}{100} * O(1) + \frac{1}{100} * O(n) = O(n)$$

Stack : S

Push(S), Pop(S), Multipop(S, k) → Remove k no. of elements from S

Multipop(S, k)
1. While non-empty(S) and K != 0
2. Pop(S)
3. K := K - 1

min(l, k) → l = No. of elements in S

Binary Counter:

Increment(A) → Array
1. i := 0
2. While i < A.length and A[i] = 1
3. A[i] := 0
4. i := i+1
5. if i < A.length
6. A[i] := 1

1 → 0 (Reset)
0 → 1 (Set)

	5	4	3	2	1	0 → Position
	1	0	1	1	1	1 → A
After 1st increment →	1	1	0	0	0	0

Amortized Analysis:

1. Aggregate method
2. Accounting method
3. Potential method

1. **Aggregate**: Total no. of operations are considered and their average is calculated.

Stack: $n \rightarrow$ total no. of operations (Push, Pop and Multipop)

$$O(n)/n = O(1)$$

Binary counter:

10000
10001
10010
10011
10100
10101
10110
10111
⋮

$$n + \frac{n}{2} + \frac{n}{4} + \cdots = n\left[1 + \frac{1}{2} + \frac{1}{4} + \cdots\right]$$

$$= n \cdot 2$$

$$\text{Average} = \frac{2n}{n} = 2$$

For binary counter Increment operation, on an average, two bits are changing.

2) **Accounting method**: We assign different charges to different operations.

Stack: Amortized cost of Push $\rightarrow 2$

Pop $\rightarrow 0$

Multipop $\rightarrow 0$

Binary counter: $1 \rightarrow 0$ (Reset)

$0 \rightarrow 1$ (Set)

Amortized cost of increment $\rightarrow 2$

③ **Potential method**: Potential Function (PF) is considered

Stack: PF: No. of elements in the stack

Amortized cost of Push (A·C) = Actual cost + Effect of the operation

$= 1 + (K+1-K)$

$= 2$

Amortized cost of Pop $= 1 + (K-1-K)$ A·C

$= 1 - 1 = 0$

Amortized cost of Multipop $= \min(l,K) - \min(l,K)$

$= 0$

Binary counter:

PF: No. of 1's present in the counter

Amortized cost of increment

$$= A \cdot C + \text{Effect of the operation}$$

$$= \underbrace{t_i + 1}_{\text{No. of digits that are changing}} + (-t_i + 1) = 2$$

t_i

$1 \to 0$ (Reset)

$0 \to 1$ (Set)

$101111 \to$

$110000 \to$