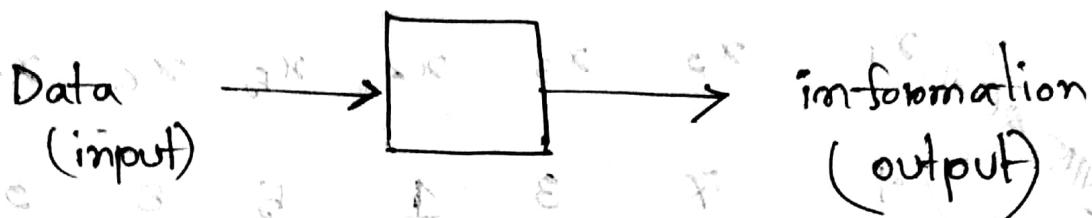


# Chapter 10 - 1

## Lec - 1



~~# Define data structure & with an example:~~

⇒ Formal definition:

- ① A set of function definitions.
- ② A storage structure.
- ③ A set of algorithms.

Lee - 2

## Chapters - 8

Bubble Sort  $\rightarrow O(n) / O(1)$

## ୬ ଆଜାନୋ

ମାର୍କଲେ ୧ ବାଜୁ

	$x_1$	$x_2$	$x_3$	$x_4$	$x_5$	$x_6$	$x_7$	$x_8$
initial	4	7	3	1	5	8	2	6
pass 1:	4	3	1	5	7	8	2	6
pass 2:	4	3	1	5	7	8	2	6
pass 3:	4	3	1	5	7	8	2	6
pass 4:	1	3	4	5	2	5	6	7
pass 5:	1	2	3	4	5	6	7	8
pass 6:	1	2	3	4	5	6	7	8
pass 7:	1	2	3	4	5	6	7	8

## Algorithms

while  $K \neq 0$

(Ex) ER. Wk3 off for  $j=1$  to  $b+k-1$  do  
 (from reverse  $j=0$ )  $k-2$

$x_j > x_{j+1}$  then (comparison)

$x_j \leftrightarrow x_{j+1}$  (arbitrary swap move करते हैं)

十一

$$\begin{aligned} \text{temp}' &= x/x_j \\ x_j/x &= y/x_{j+1} \\ +1/y &= \text{temp} \end{aligned}$$

$n-i$  ( : no আজি অতবার move করবে অতবার  
left -> বড় সাঁতা মাঝে )

$d_{ij}$ : Number of data larger than  $x_{ij}$  and

top sw ①  $\pi_{\text{pp}}^0$  at  $i = i_0$   $\theta = \theta_0$

$$(1-i) \sum_{k=0}^{\infty} e_k = r_0 b + \dots + r_n b + s b + sb$$

## Lec-3

Total no of data movement:

$$d_1 = 0$$

$$d_1 + d_2 \quad (\text{2 no অঞ্চল কর্তৃত পরিষেবা})$$

$$d_1 + d_2 + d_3 \quad (3^{\text{rd}} \text{, } " \text{, } ")$$

$$\dots \rightarrow d_1 + d_2 + d_3 + d_4 + \dots + d_n = 3 \sum_{j=2}^n d_j \quad \text{Eq. 1}$$

$\text{total no of data swap}$

$\therefore d_2 + d_3 + d_4 + \dots + d_n = 3 \sum_{j=3}^n d_j \quad \text{Eq. 1 নাইন কর্তৃত পরিষেবা}$

i) Best Case:  $d_j = 0$

মাত্র কয়েক মুভ  
ক্যালসুল সোর্ট  
হ্যাভেচ

Putting  $d_j = 0$  in eq<sup>n</sup> ① we get;

$$d_2 + d_3 + \dots + d_n = 3 \sum_{j=2}^n 0 \\ = 0$$

ii) Worst Case:  $d_j = j - 1$

Putting  $d_j = j - 1$  in eq<sup>n</sup> ① we get;

$$d_2 + d_3 + d_4 + \dots + d_n = 3 \sum_{j=2}^n (j-1)$$

$$= 3 \{ (\underline{2-1}) + (\underline{3-1}) + (\underline{4-1}) + \dots + (\underline{n-1}) \}$$

$$= 3 \{ 1+2+3+\dots+(n-1) \}$$

$$= 3 \frac{n(n-1)}{2}$$

③ Avg. Case:  $\text{df} = \frac{j-1}{2}$

$$= \sum_{j=2}^n \frac{j-1}{2}$$

$$= 3 \left\{ \frac{(\underline{2-1}) + (\underline{3-1}) + \dots + (\underline{n-1})}{2} \right\}$$

$$= 1 - 3 \frac{\frac{n(n-1)}{2}}{2}$$

$$= \frac{3n(n-1)}{4}$$

$$+ (8-16) + (1-8)$$

Lec-4

(b) Number of comparison's In Bubble Sort:

For loop  $n-1$  (  $k-1$  পর্যন্ত for loop শুরু  
 $k=n$   
so  $n-1$ )

\* Worst Case:  $n - 1 + 1$

largest dj  $\leftarrow$  n

For loop:  $(n-1) + (n-2) + \dots + 1$

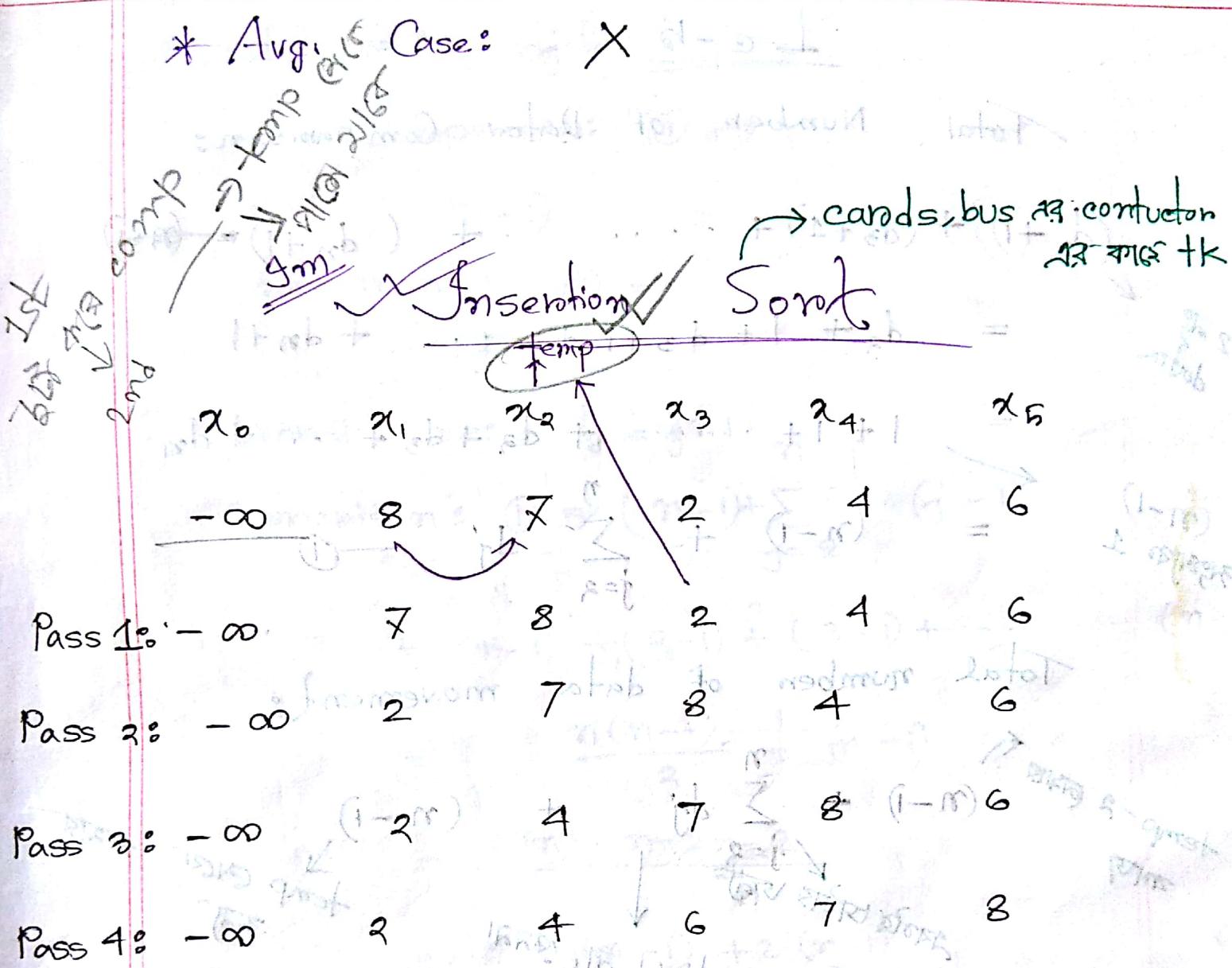
$$= \frac{n(n-1)}{2}$$

Write:  $\frac{n(n-1)}{2}$

Calculate:

କରାନ୍ତିର  
ପରିମାଣ

\* Avg. Case:  $\times$



Lc-15

Total Number of Data Comparisons:

$$\begin{aligned}
 & (d_2 + 1) + (d_3 + 1) + \dots + (d_n + 1) \\
 &= d_2 + 1 + d_3 + 1 + \dots + d_n + 1 \\
 &= 1 + 1 + \dots + d_2 + d_3 + \dots + d_n \\
 &\stackrel{1}{=} (n-1) + \sum_{j=2}^n d_j \quad \text{--- } ①
 \end{aligned}$$

Total number of data movement:

1) Best case:  $d_j = 0 \Rightarrow j = 0$

Comparison: ①  $\Rightarrow$

$$(n-1)$$

Movement: ①  $\Rightarrow$

$$(2n-2)$$

2) Worst Case:  $d_j = j - 1 \quad n$

Comparison: ①  $\Rightarrow (n-1) + \sum_{j=2}^n (j-1)$

$$= (n-1) + (2-1) + (3-1) + \dots + (n-1) \quad (n-1)$$

$$= \frac{n(n-1)}{2} + (n-1)$$

$$= \frac{n^2 - n + 2n - 2}{2}$$

$$= \frac{n(n-1) + 2(n-1)}{2}$$

$$= \frac{(n-1)(n+2)}{2}$$

Movement:  $(2n-2) + \frac{n(n-1)}{2}$

$$= \frac{4n - 4 + n^2 - n}{2}$$

$$= \frac{4(n-1) + n(n-1)}{2}$$

$$= \frac{(n-1)(n+4)}{2} \quad \text{send to } \mathbb{C}$$

$$\textcircled{3) Avg. Case: } \quad d_j = \frac{j-1}{3}$$

$$\text{Comparison: } (n-1) + \sum_{j=2}^n d_j$$

$$= (n-1) + \sum_{j=2}^n \frac{j-1}{2}$$

$$(1-\textcircled{3}) \quad \frac{3+(1-n)}{2} \leq \textcircled{1}$$

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

$$+ (1-\varepsilon) + (1(n-1)) + \frac{n^3 - n}{4}$$

$$(1-\textcircled{3}) + \frac{(1-n)4n-4+n^2-n}{4}$$

$$\frac{4(n-1) + n(n-1)}{4}$$

$$(1-\textcircled{3}) + \frac{(n-1)(n+4)}{4}$$

$$\text{Movement: } (2n-2) + \frac{1}{2} \times \frac{n(n-1)}{2}$$

$$(1-\textcircled{3}) (2n-2) + \frac{n^3 - n}{4}$$

$$= \frac{8n-8+n^3-n}{4}$$

$$(1-\textcircled{3}) \frac{8(n-1) + n(n-1)}{4}$$

(Bubble Sort + Insertion Sort  $\rightarrow$  imp for exam)

$$= \frac{(n-1)(n+8)}{4}$$

Algorithm:

$$x_0 \leftarrow -\infty$$

for  $j = 2$  to  $n$

$$t \leftarrow x_j$$

$$i \leftarrow j-1$$

While  $t < x_i$

$$x_{i+1} \leftarrow x_i$$

$$i \leftarrow i-1$$

$$x_{i+1} \leftarrow t$$

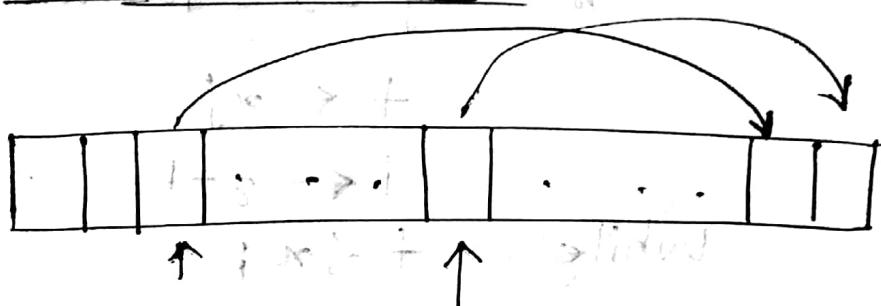
After 1st pass  
After 2nd pass  
After 3rd pass  
 $x_0 \leftarrow -\infty$

8	5	3	7	2	4	6	1
3	5	8	7	2	4	6	1
3	5	2	7	4	6	8	1
3	2	5	7	4	6	8	1

13/5 → 1st quiz

## Lec-6

### Selection Sort



১ - ১) আগে select করে মেঘে

in exchange, last - র মাজে

element  
to opposite  
swap

৩) প্রবর্তন করতে

8 7 2 4 6

Pass 1: 6 7 2 4 8

Pass 2: 6 4 2 7 8

Pass 3: 2 4 6 7 8

for  $j = n - 1$  to 2 by -1 do

$+ \leftarrow 1$

for  $k = 2$  to  $j$  do

$x_t \leftarrow x_k$

$+ \leftarrow k$

$x_t \leftrightarrow x_j$

অটো swap

ক্রমান্বয় swap করা

#  $\therefore$  Best / Worst

সার্টিফিল্মেন্ট

$$= 3(n-1)$$

# Why? ( 3 marks )

প্রতিজ্ঞ করে 3 এড়া অঠাই, নিজের- স্থানে পিছে exchange

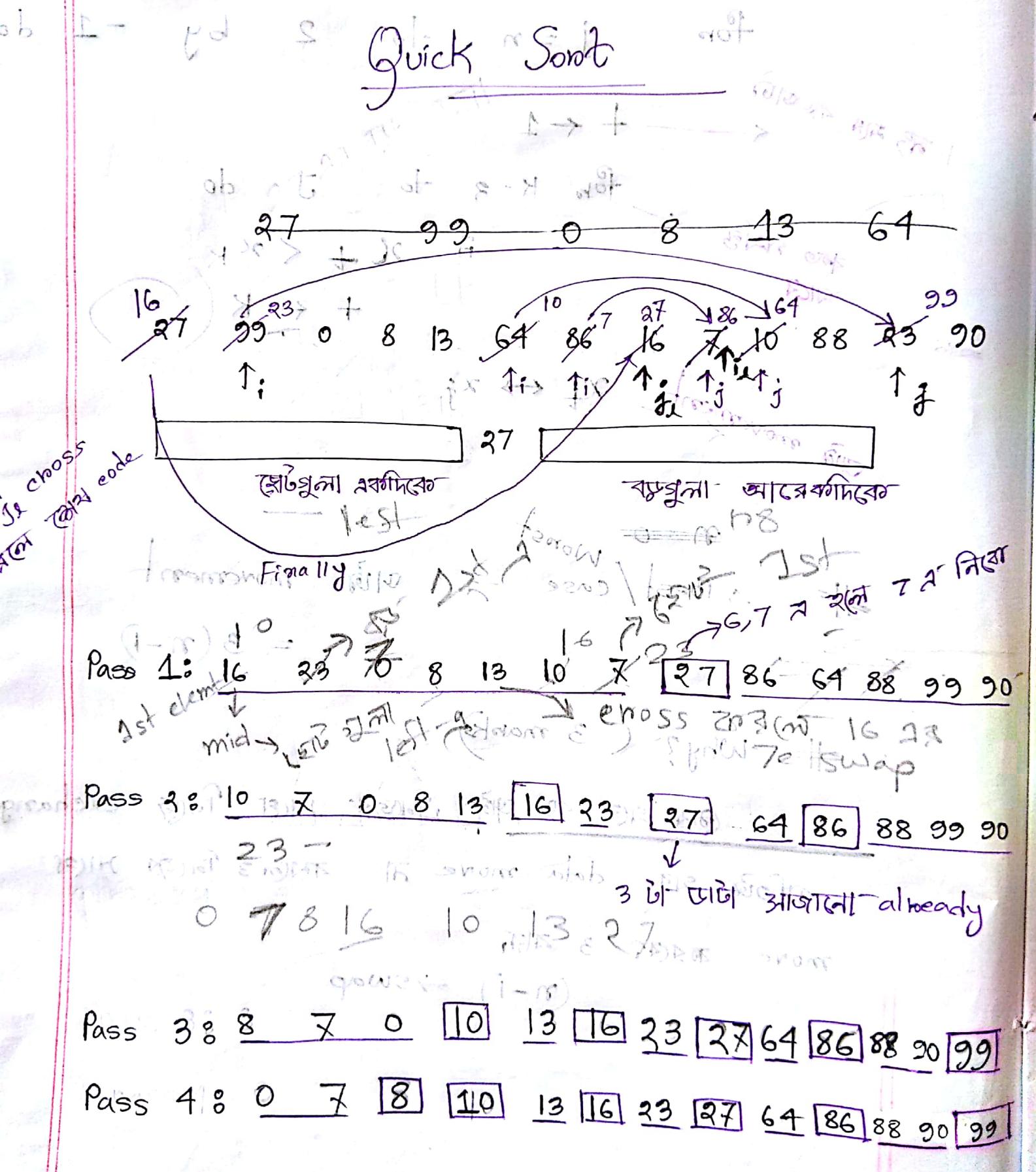
প্রতিজ্ঞ করে data move না বরাবরে নিজে- স্থান

move করবে 3 বার

$(n-1) \rightarrow \text{swap}$

## Lec-7

### Quick Sort



# disadavantg'?

# advantage?

① processor মেমী-কাজে লাগানো আবে.

② parallel processing.

← Quicksort ( $f, l$ )

$f < l$  then

$i \leftarrow f + 1$

while  $x_i < x_f$  do  $i \leftarrow i + 1$

$j \leftarrow l$

while  $x_j > x_f$  do  $j \leftarrow j - 1$

while ( $i < j$ ) do  $\boxed{\quad}$

$x_i \leftrightarrow x_j$

repeat  $i \leftarrow i + 1$  until  $x_i \geq x_f$

repeat  $j \leftarrow j - 1$  until  $x_j \leq x_f$

$x_j \leftrightarrow x_f$

Quicksort ( $f, j-1$ )

" " ( $\underline{j-1}, l$ )

## Lec-8

Merge Sort

Ans:

1	2	3	4	5	6	7	8
---	---	---	---	---	---	---	---

4	7	3	1	2
---	---	---	---	---

5	8	2	6	2
---	---	---	---	---

4	7
3	1

5	8
2	6

4	7	3	1
---	---	---	---

5	8
2	6

ques: 4 7 3 1 5 8 2 6

(2) <sup>2</sup>  
(2) <sup>16</sup>  
24

(i-1+1) Worst Case  
(i-1+1) + 1

Merge - sort ( $A, P, R$ )

- 1) if  $P \leq R$
- 2)  $q = \left\lceil \frac{P+R}{2} \right\rceil$
- 3) Merge ( $A, P, q$ )
- 4) Merge ( $A, q+1, R$ )
- 5) Merge ( $A, P, q, R$ )

Merge ( $A, P, q, R$ )

- 1)  $n_1 = q - P + 1$
- 2)  $n_2 = R - q$
- 3) let  $L [1..n_1 + 1]$  and  $R [1..n_2 + 1]$
- 4) for  $i = 1$  to  $n_1$
- 5)  $L[i] = A[P+i-1]$
- 6) for  $j = 1$  to  $n_2$
- 7)  $R[j] = A[q+j]$
- 8)  $L[n_1 + 1] = \alpha$
- 9)  $R[n_2 + 1] = \alpha$
- 10)  $i = 1$
- 11)  $j = 1$

- 12) for  $K = p$  to  $q$  do  
 13)    if  $L[i] \leq R[j]$  then  $A[k] = L[i]$   
 14)     $A[k] = L[i]$   
 15)     $i = i + 1$  step(A)  
 16)    else  $A[k] = R[j]$  step(A)  
 17)     $j = j + 1$  step(A)

(step(A)  $\rightarrow$  program)

$$l+q-p = 16 \quad (1)$$

$$p-q = 8 \quad (2)$$

2.8 2nd time  $[l+16, l+16]$  lisi

step of  $l=16$  not

$[l+16] A = [l] A$

else of  $l=16$  not

$[l+16] A = [l+1] A$

$l = [l+16] +$

$l = [l+16] A$

$l = i$

Lee - 9

Radicin Sort

207 095

646

198

809

376

917 534 310 209

0 - 310

1 -

2 -

3 - 534

4 - 534

5 - 095

6 - 646 → 376

7 - 207 → 917

8 - 108

9 - 809 → 209

207 → 108 → 809 → 209

310 → 917

534

646

376

095

108

207 → 209

310 → 376

534

646

809

917

Pass 1: 310 534 095 646 376 207 917 108 809 209

Pass 2: 207 108 809 209 310 917 534 646 376 095

Pass 3: 095 108 → 207 209 310 376 534 646 809 917

## Chapter - 4 থেকে

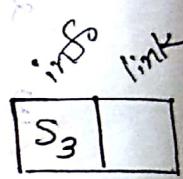
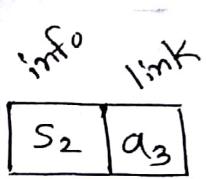
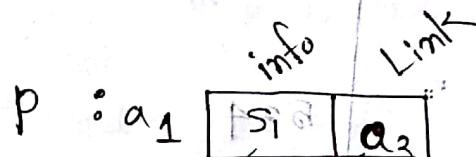
২ ত্রি ques final - ১

L - 10

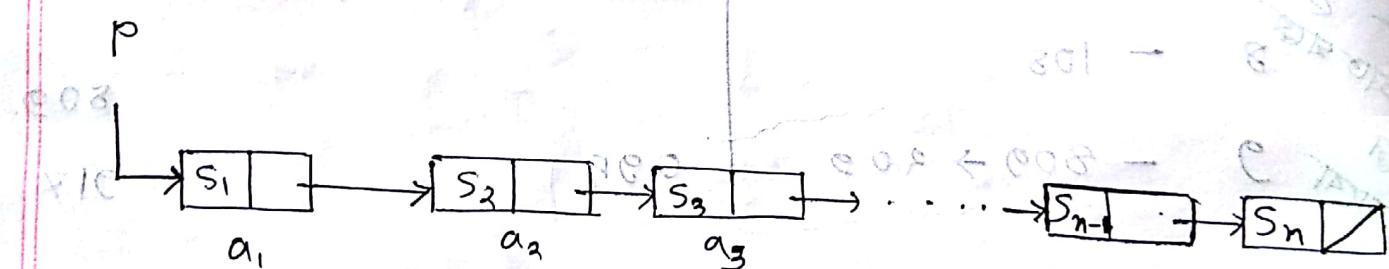
### ~~Imp~~ Chapter - 4

int  
 $x = 10$   
 $x = 10.5$   
 $x = 10.5y$   
float \*p  
int/float \*

~~Diagram~~ ~~Linked List~~



data address



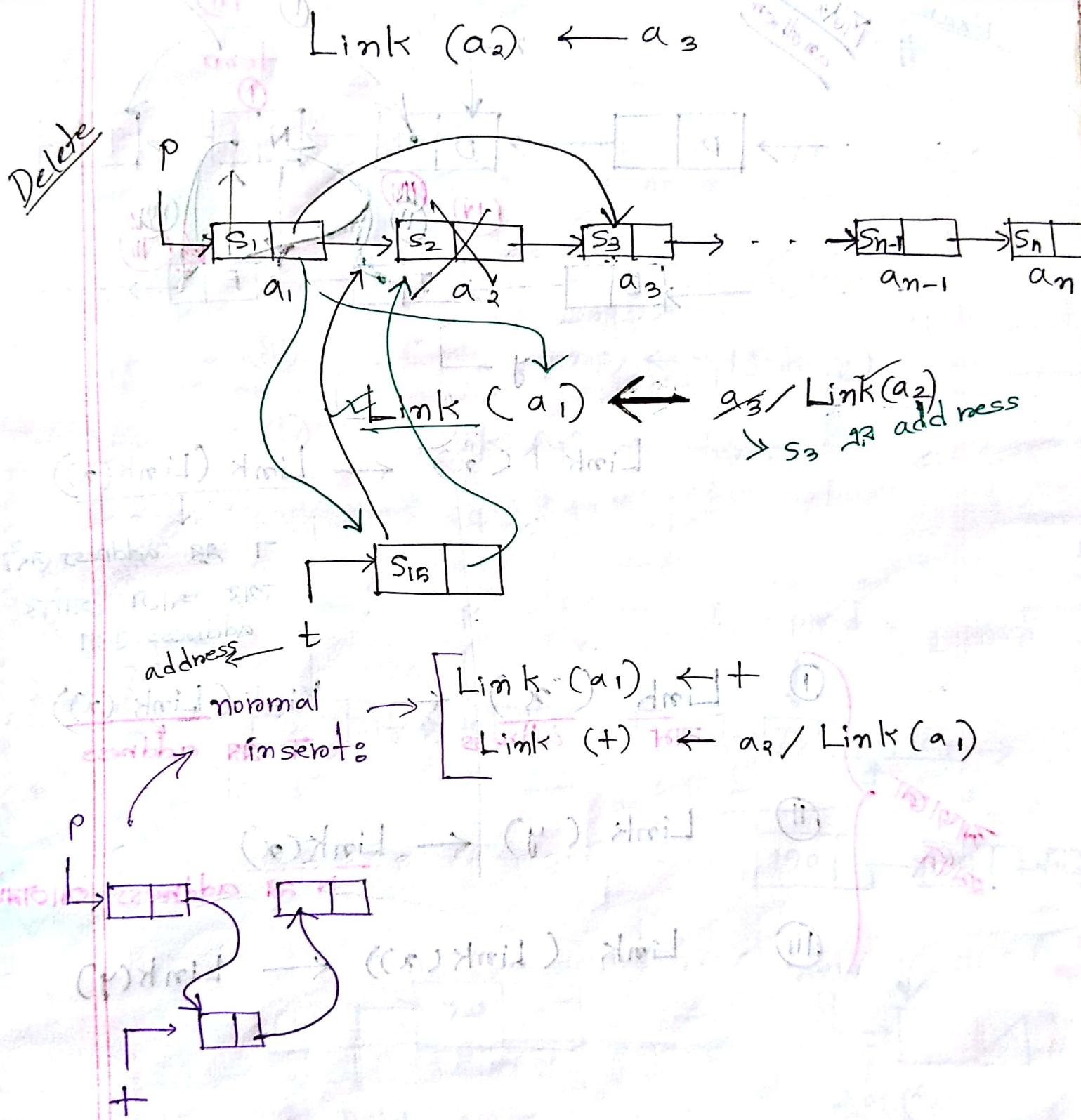
$a_1 \quad a_2 \quad a_3 \quad \dots \quad a_{n-1} \quad a_n$

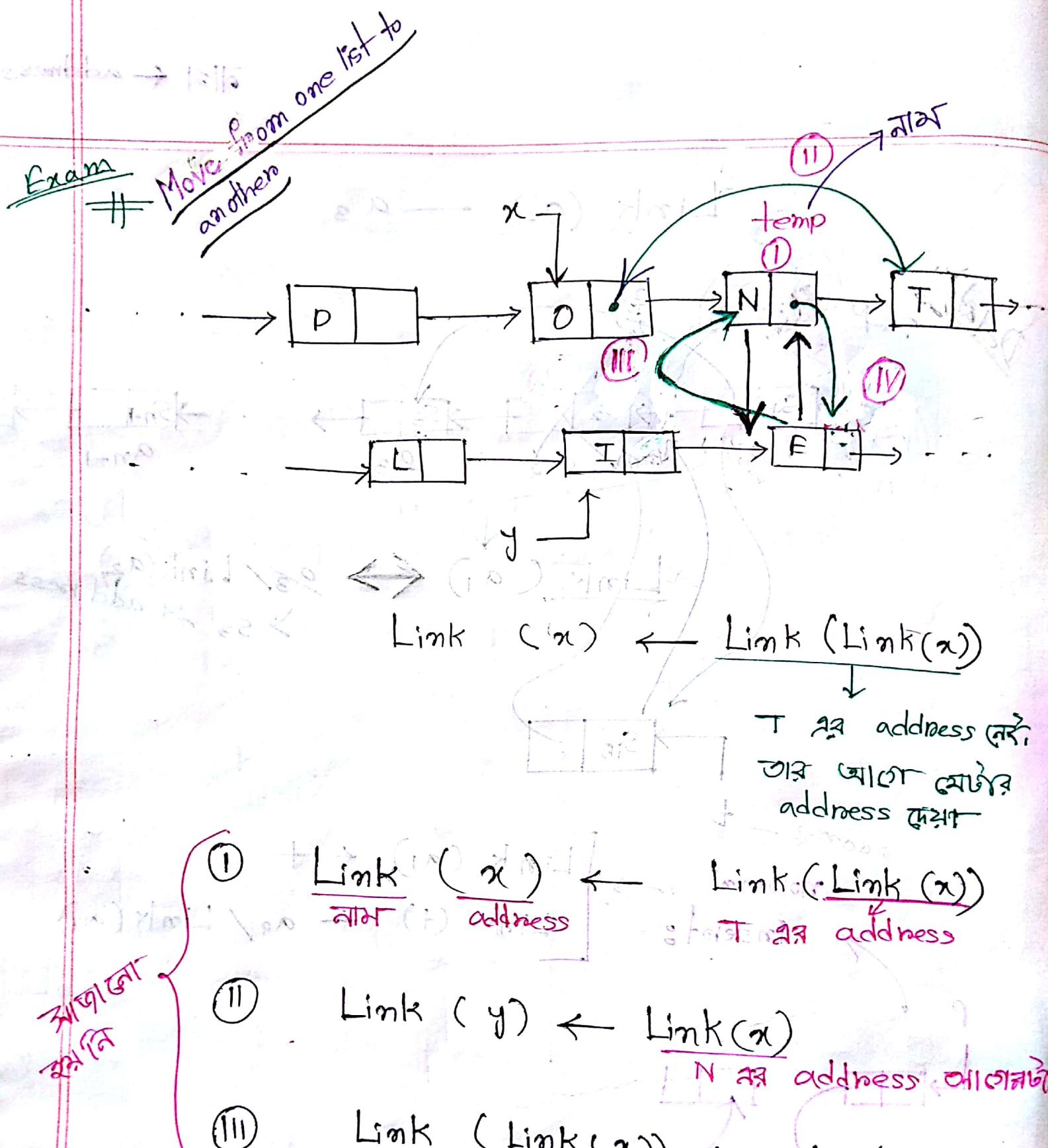
info  $\leftarrow s_2$

info ( $a_2$ )  $\leftarrow s_2$   
address  $\leftarrow$  ~~link~~  $\leftarrow$  ~~info~~

File  $\rightarrow$  ক্লাসগুলির  
ক্লাসগুলির ঠিকানা

पाइ  $\leftarrow$  address





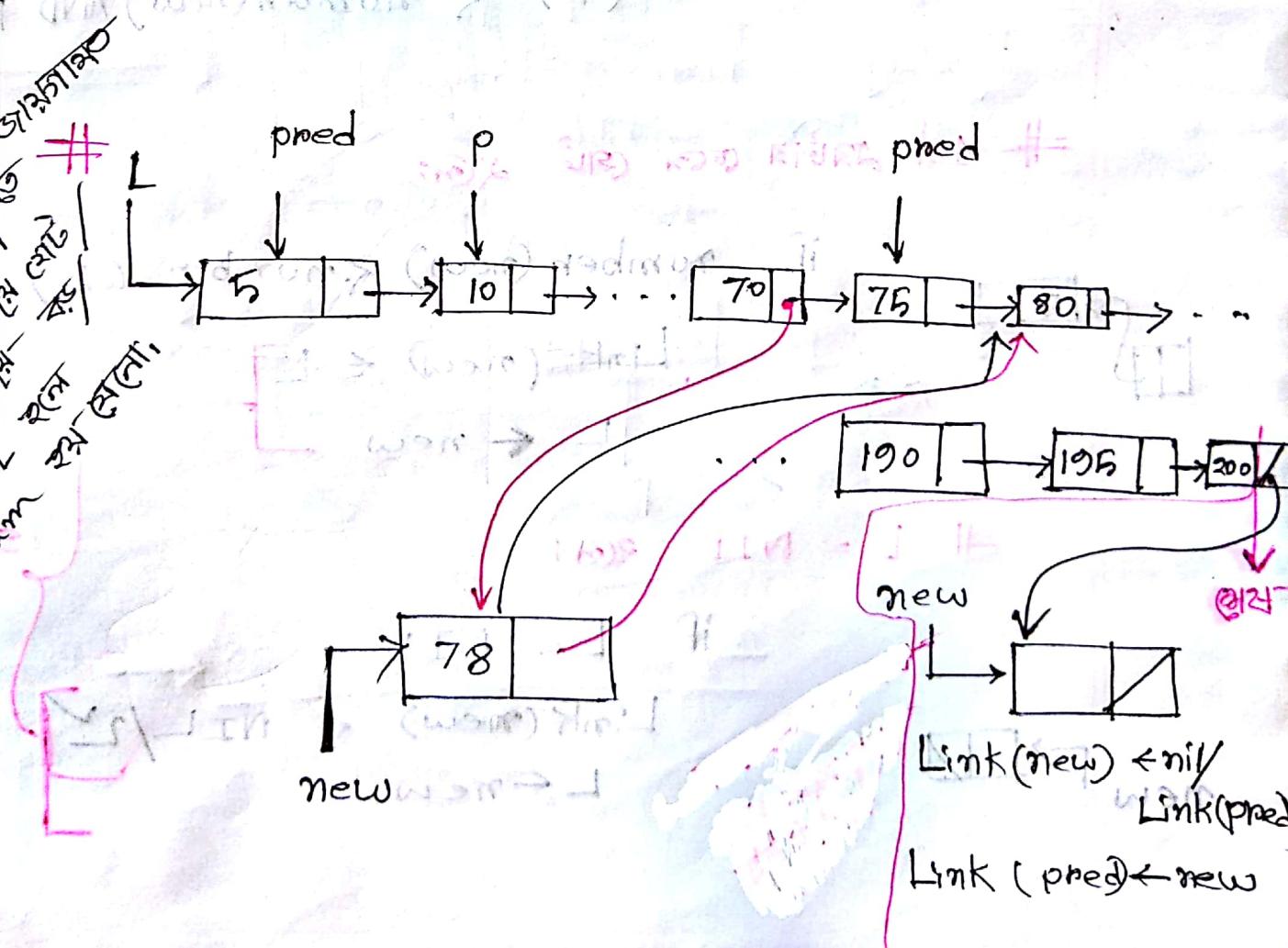
## Lec - 11

①  $\text{temp} \leftarrow \text{Link}(\alpha)$

②  $\text{Link}(\alpha) \leftarrow \text{Link}(\text{temp})$

③  $\text{Link}(\text{temp}) \leftarrow \text{Link}(\gamma)$

④  $\text{Link}(\gamma) \leftarrow \text{temp}$



# If node is advantage

pred

while number (P) < number (new)

pred  $\leftarrow$  P

P  $\leftarrow$  Link (P)

Link (new)  $\leftarrow$  Link (pred)

Link (pred)  $\leftarrow$  new

# 200 টা প্রতি হিসেব

while number (P) < number (new) AND P  $\neq$  NIL

# 1st প্রতির চেয়ে হোট রীত

if number (new) < number (L)

Link (new)  $\leftarrow$  L

L  $\leftarrow$  new

# L = NIL হলে

if L = NIL

Link (new)  $\leftarrow$  NIL / 2 ✓

L  $\leftarrow$  new

Same  
thing

new

$\rightarrow$  [ ]

# Link List এর সূচীবর্ণ / অসূচীবর্ণ  
direct delete / insert, dynamic memory allocation  
array - static

জায়গার প্রয়োজন; searching  $\rightarrow$  prob(binary)

### Full algo:

if number (new) < number (L) or L=NIL

Link (new)  $\leftarrow$  L

L  $\leftarrow$  new

else pred  $\leftarrow$  L

P  $\leftarrow$  Link (L)

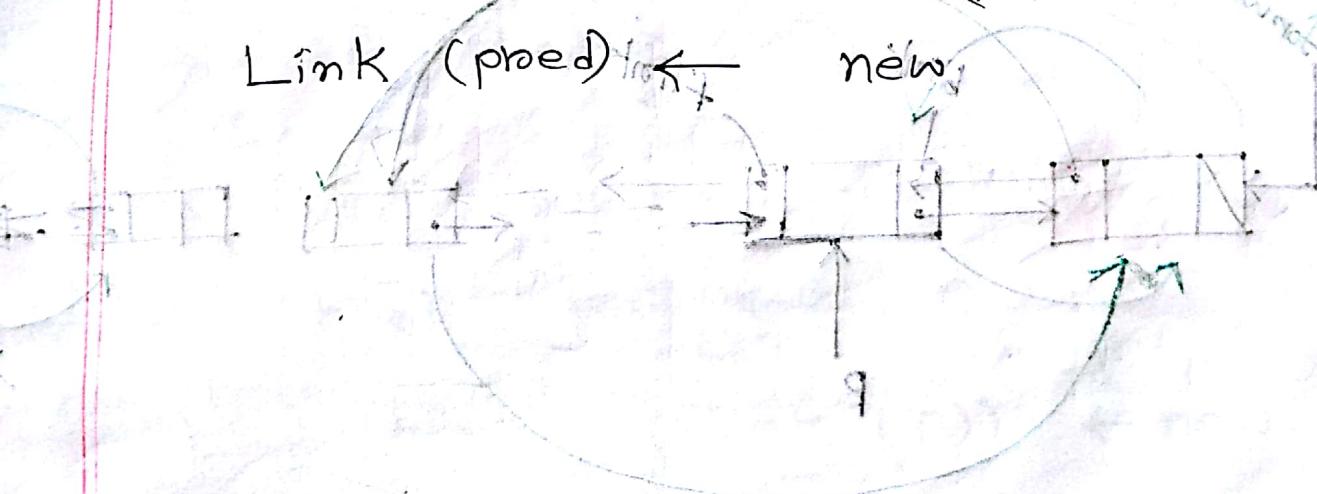
while number (P) < number (new) And P $\neq$ NIL

pred  $\leftarrow$  P

P  $\leftarrow$  Link (P)

Link (new)  $\leftarrow$  Link (pred)

Link (pred)  $\leftarrow$  new



(q) Head  $\rightarrow$  ((q) Head) Head

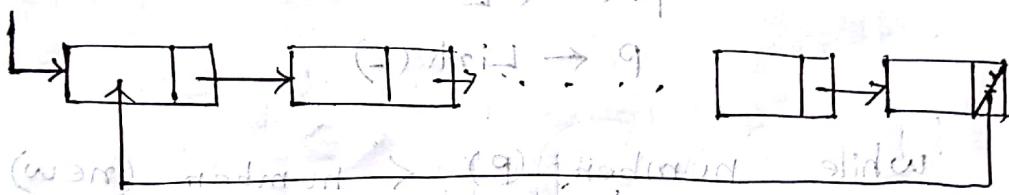
(q) Head  $\rightarrow$  ((q) Head) Head

## Lec - 13

### Common Variants Of Linked List

1. Singly linked list

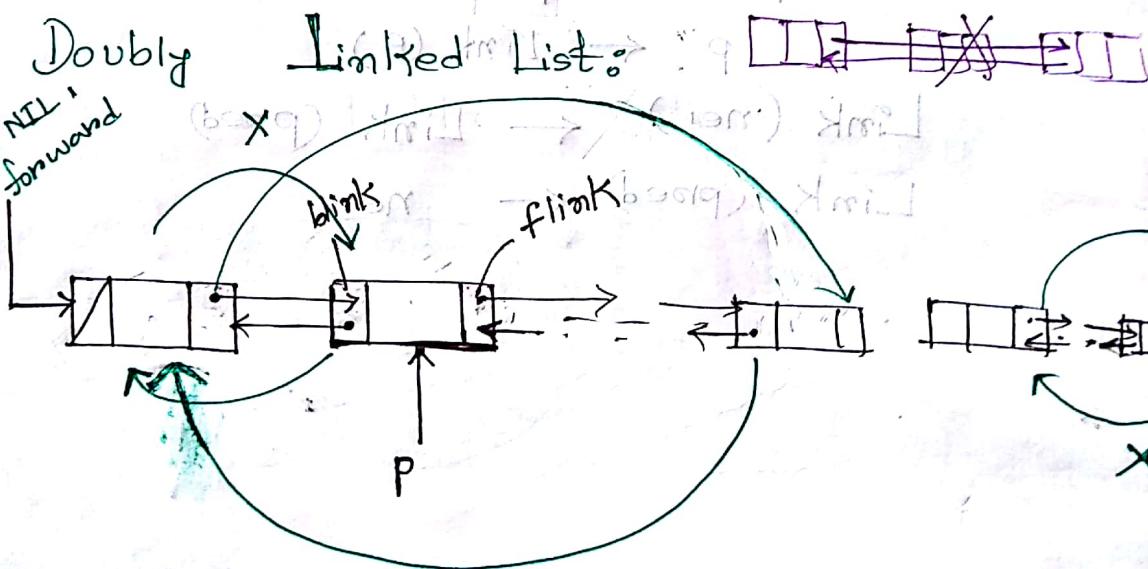
Circular Linked List:



Doubly

Linked List:

1st node  
backward  $\rightarrow$  NIL  
Last node  
 $\rightarrow$  NIL



flink (blink (P))  $\leftarrow$  flink (P)

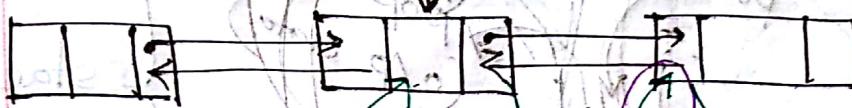
blink (flink (P))  $\leftarrow$  blink (P)

বর্ণনা এবং

Deletes { If  $\text{blink}(P) \neq \text{Nil}$  then  $\text{flink}(\text{blink}(P)) \leftarrow \text{flink}(P)$   
If  $\text{flink}(P) \neq \text{Nil}$   $\text{blink}(\text{flink}(P)) \leftarrow \text{blink}(P)$

Note:  $\rightarrow$   $\leftarrow$   $\leftarrow$   $\leftarrow$   $\leftarrow$   $\leftarrow$

# Insert:



$\text{flink}(\text{new}) \leftarrow \text{flink}(P)$

$\text{blink}(\text{new}) \leftarrow P$

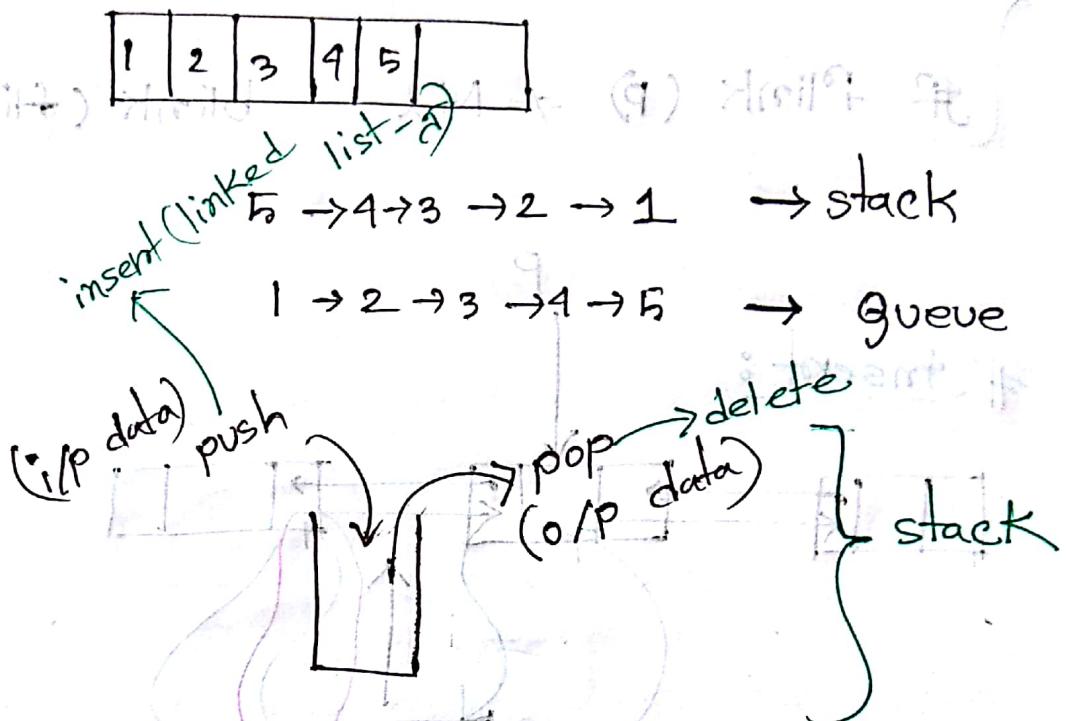
$\text{blink}(\text{flink}(P)) \leftarrow \text{new}$

$\text{flink}(\text{blink}(P)) \rightarrow \text{new}$

## Stacks / Queues

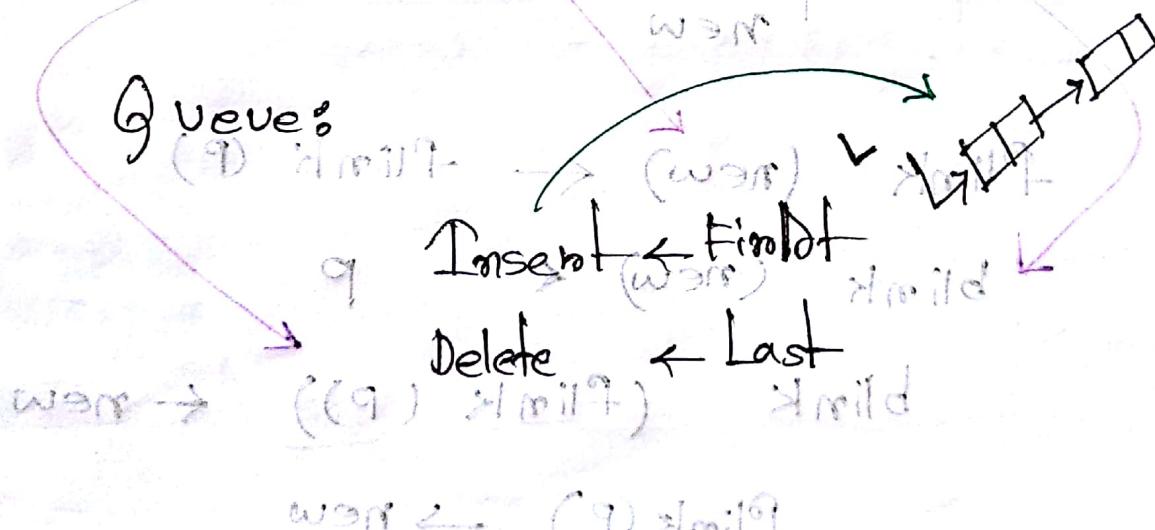
1	2	3	4	5	6
1	2	3	4	5	6

Exampush, pop

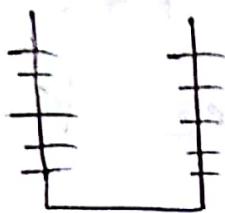


LIFO / FILO

Queues



## Lec-13



$S \leftarrow$  empty stack

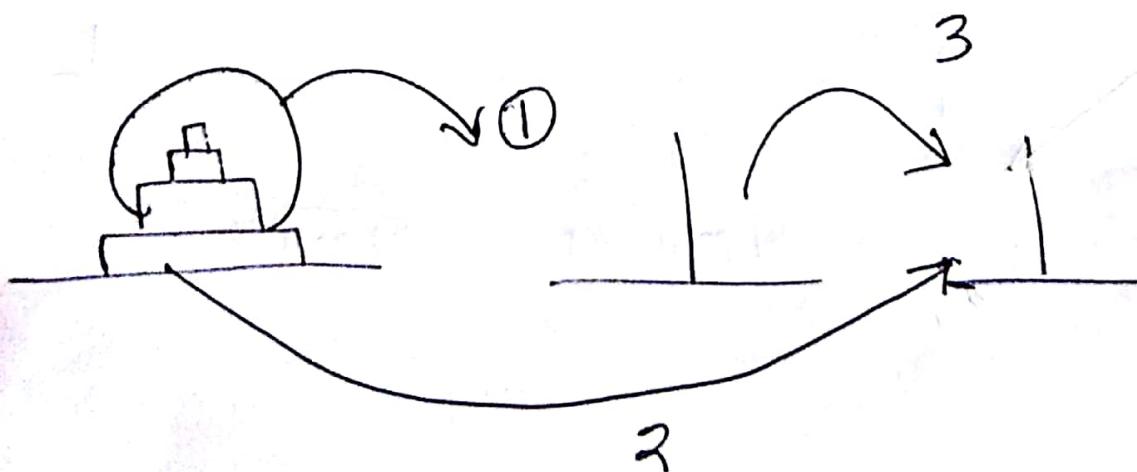
$Q \leftarrow$  empty queue

$D \leftarrow n$

$x \leftarrow D$

top ( $S$ )

## Tower Hanoi



## Recursive

Hanoi ( $n, i, j, k$ )

if  $n = 1$ , then move the top disk  
from pole  $i$  to pole  $k$

Hanoi ( $n-1, i, k, j$ )

move the top disk from pole  
 $i$  to pole  $k$

Hanoi ( $n-1, j, i, k$ )

एक डेटा को लिया जाएगा तो  
जहाँ पर डेटा  
stack की property

move C - shortest to remove vertically

## Lec - 14

Non-rec:  $S \leftarrow$  empty stack

$S \leftarrow (n, 1, 2, 3)$

~~S  $\leftarrow$  data~~  
~~push  $\leftarrow$~~

while  $S \neq$  empty do

$(n, i, j, k) \leftarrow S$

pop /

for value ~~stack~~ if  $n = 1$  then move  $i$  to  $k$   
for variable

else

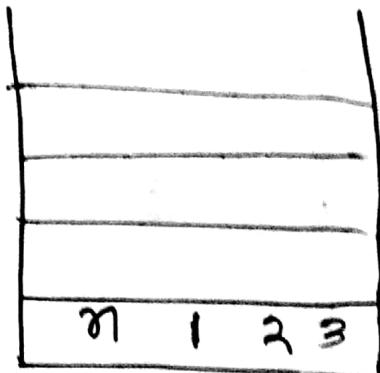
$i \leftarrow (i-1)$

$S \leftarrow (n-1, j, i, k)$

$S \leftarrow (1, i, j, k)$

$S \leftarrow (n-1, i, k, j)$

stack property



Algorithm/ démonstration → exam

Démonstration :

H = 3e-1

start pos → 2

(0, 0, 1, 0) → 2

i j k

1 2 3

n-1 i k j

1 3 2

i j k m

n-1 m

obj pos → 2

2 → (1, 0, 1, 0) j k

2 3 1 2

n-1 i k j

1 3 1 2

1 2 3

i j k m

n-1 m

(0, 0, n-1) → 2

(0, 0, n-1-m) → 2

1 1 2 3

n

i

j

k

3

1

2

3

2

1

3

2

1

1

3

2

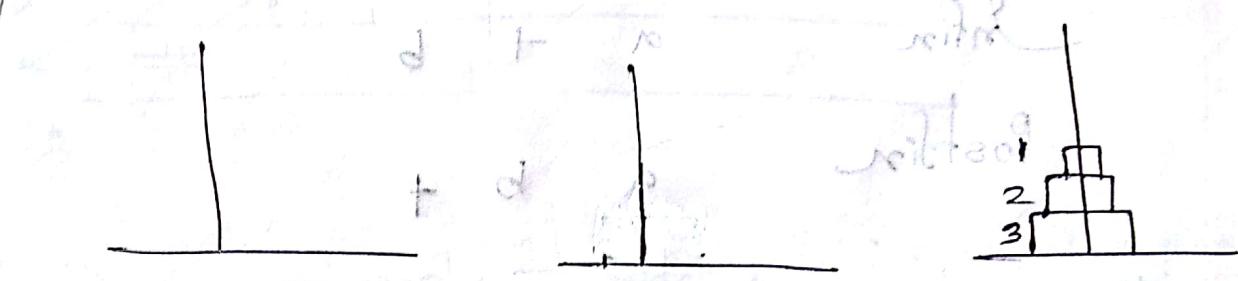
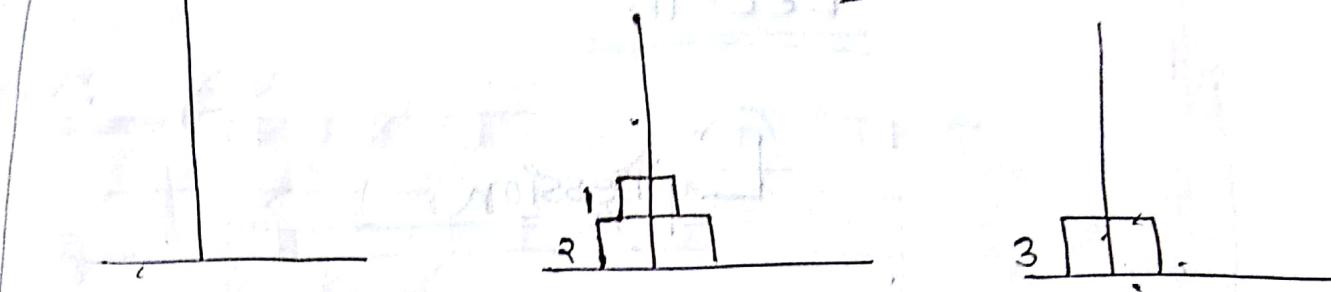
3

1

2

3

<u><math>n</math></u>	<u><math>i</math></u>	<u><math>j</math></u>	<u><math>k</math></u>
2	$\frac{n}{2}$	1	3
1	2	3	1
1	2	1	3
	1	2	3



$\leftarrow k + 7 \times (d-2) + (d+1) = 8d - 9$

Final  
stack

$$k + 7 \times 2 - 13 + 9A = 8d - 9$$

(1) (2)

$$k + 14 - 13 + 9A = 8d - 9$$

$\leftarrow k + 1 - 13 + 9A = 8d - 9$

$$(3) A = \frac{k + 1 - 13 + 9A}{8d - 9}$$

Algorithm / Postfix / ক্ষেত্র ক্ষেত্র <sup>জ্ঞান</sup> → Exam

## Lec-16

### Expression

X Prefix

+ a b

Infix

a + b

Postfix

a b +

# Infix:  $(A+B) * ((C-D) * E + F)$  \$ = 3

opening,  
operand

Post: AB + CD - E \* F + \*

প্রথম direct

push

বর্তমানে আগের operator pop করো  
(3) (-1) recent push

close এর পর যা  
আগে push

# A = 1 B = 2 C = 3 D = 4

E = 5 F = 6

→ 5

⇒ 3 (-1) E \* F + \*

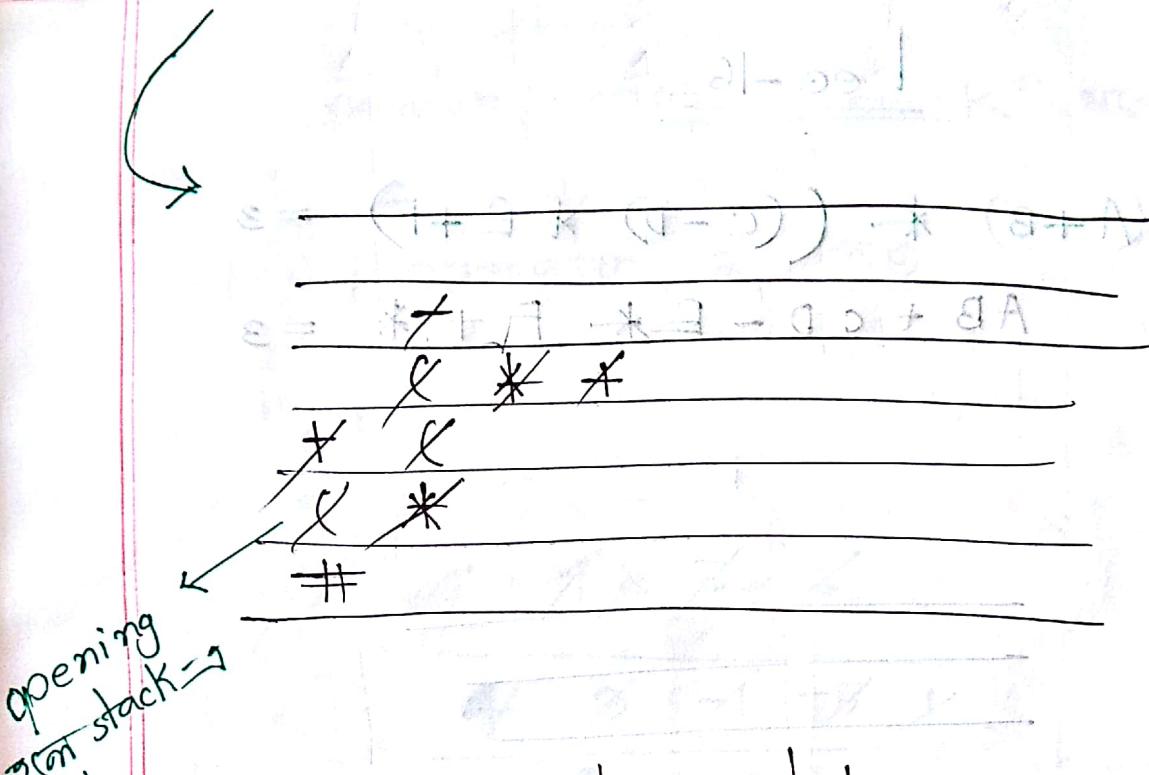
3 - 5 F + \*

3 1 \*

= 3 (Ans)

insert next  $\leftarrow$  last

exam-1 using stack বাটে দেখবে



1)  $s \leftarrow$  empty stack

2)  $\$ \leftarrow$  add (SHA)  $\leftarrow$  ~~stack~~

3)  $\# \leftarrow$  add

4) Case

closing লেন্ট  
ঘোলা আছে সেগুলো pop  
ফর্মেলা প্রতিপাদন করে এবং stack-এ opening  
নির্ণয় করে এবং stack-এ push

'(' : Stack-এ push

'operator' : আগে operator একলে সেট  
'operand' : post- $\rightarrow$  push এবং pop করে  
operator-এ stack-এ push

deq enq reverse

Lec-16

$$(A+B) * ((C-D) * E + F) = 3$$

$$\Rightarrow AB + CD - DE * F + F = 3$$

~~$$\begin{array}{r}
 D \\
 B \\
 C \\
 A+B
 \end{array}
 \begin{array}{l}
 E \\
 F \\
 (C-D) \\
 (C-D)*E \\
 (C-D)*E+F \\
 (A+B)*(C-D)*E+F
 \end{array}
 \xrightarrow{3}
 \begin{array}{c}
 \text{postfix} \\
 \text{content}
 \end{array}$$~~

Algorithm

stack



de loop টেরি



ব্রাঞ্চ ব্রাঞ্চ এ স্বামী



operand মেনে stack-এ push

operator পেলে stack থেকে pop

operator এর পার্ট

(e-i) ডাইজিট - অপেক্ষা

$$\begin{array}{r} \text{Digit} \\ \hline 4 5 6 7 8 \\ \hline 4 5 1 - 8 x \\ \hline x 3 - 3 \end{array}$$

Value

পূর্ণ  
মূলচক্র

✓ Quiz (2)

# search is sequence (start-end)  
Algo / no ques

Final

Lec - 17

Breadth

Depth

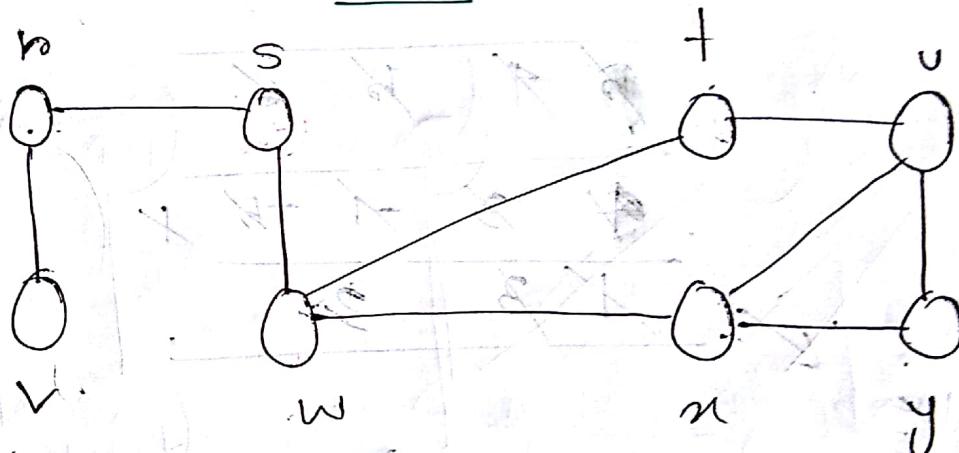
First

Search (BFS)

First

Search (DFS)

BFS



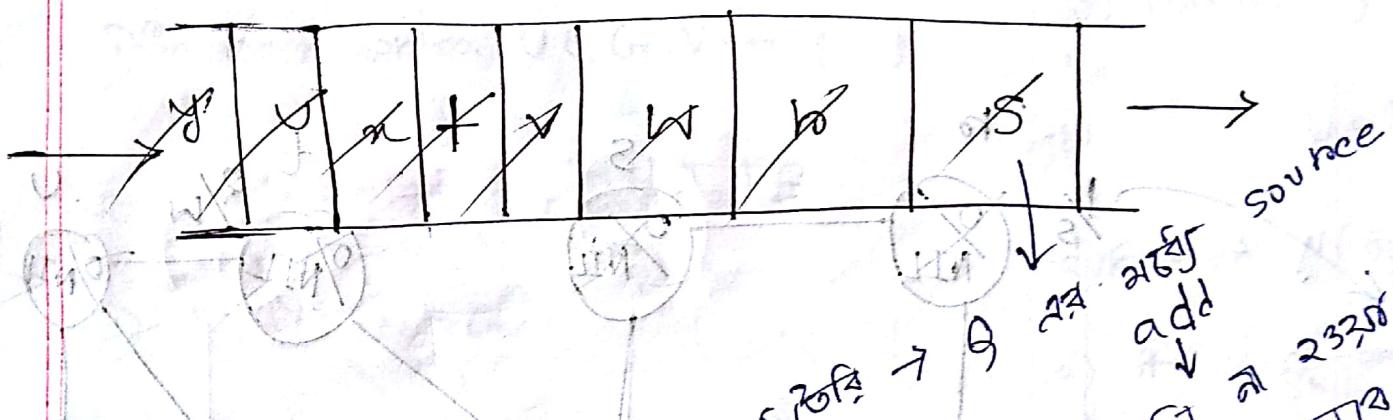
Ways 1 ||  $s \rightarrow t \rightarrow w \rightarrow v \rightarrow + \rightarrow x \rightarrow y$

Ways 2 ||  $s \rightarrow w \rightarrow t \rightarrow + \rightarrow x \rightarrow v$   
 $\rightarrow x \rightarrow v$

ফোর্ম সুপেরিয়ার  
মাত্রার নিয়ন্ত্রণ

Queue

প্রক্রিয়া করা ব্রেক করা হাস্ট



S → h → w → v → + → n → o → y

ব্রেক করা

ক্ষয় করা

add

সুলভ করা

loop করা

pop, sequence

add

a d j a c e n t

হ্যাচ করা

push করা

{... .v. u. t. e. f = V. P}

{... .z. v. s. f = E. P}

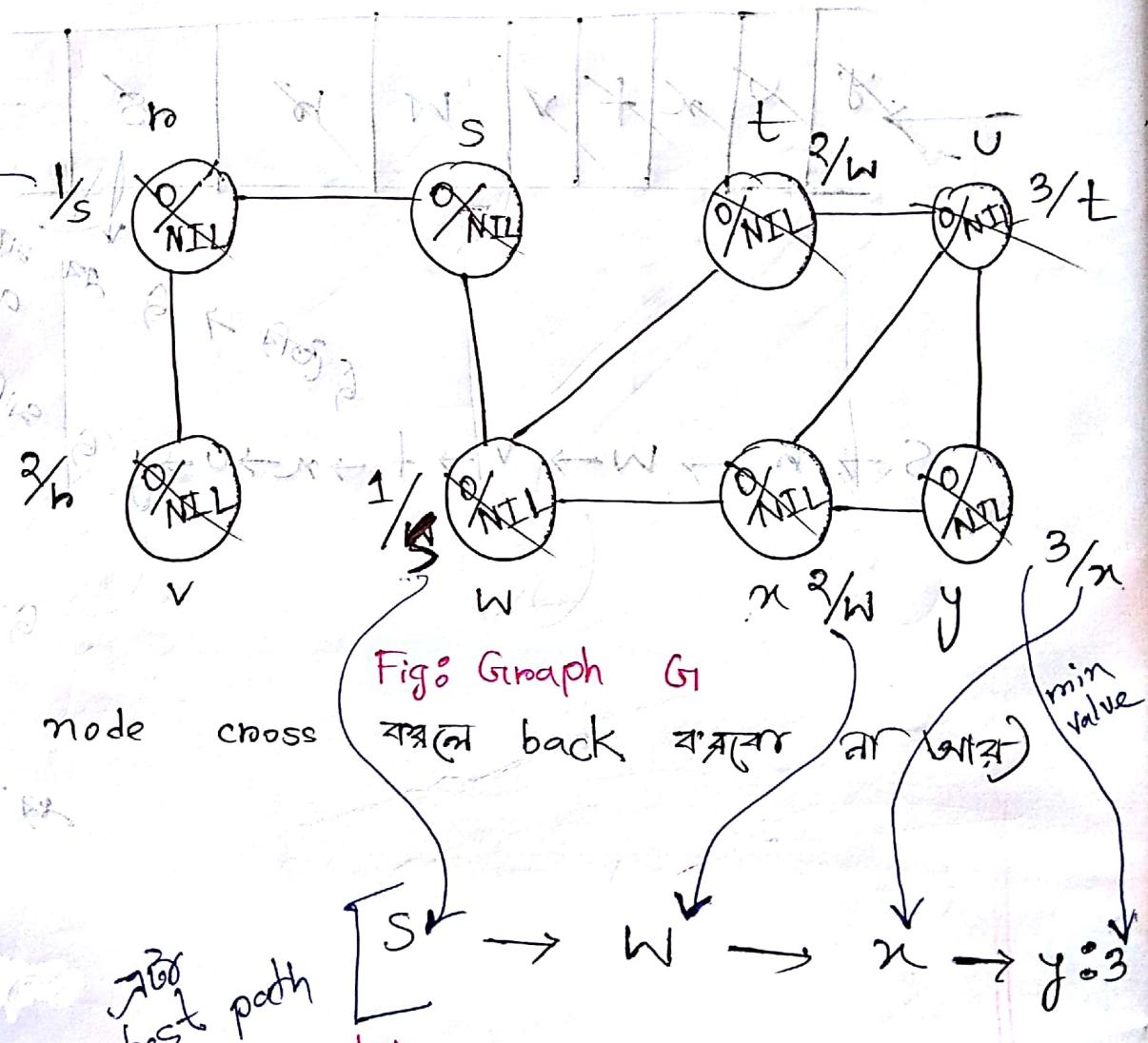
end

ফর্ম তেকনিক অ্যাপ্লি কোর

৩) রোট পথ ফলো কোরা দিন?

start থেকে end -> হাতে?

cost/value  
কাউ এবং ?



Sequence print করতে বললে  
 ৩, ১, ৬, ৭, ১৫, ১৬ লাইন  
 পিছে না

## Lec - 18

### BFS (G)

edge এজেন্সি  
 সব vertex  
 পর গ্রাফ

1) for each vertex  $U \in G, V = \{S\}$

2)  $U.\text{color} = \text{WHITE}$

3)  $U.d = \infty$

4)  $U.\pi = \text{NIL}$

5)  $S.\text{color} = \text{GRAY}$

6)  $S.d = 0$

7)  $S.\pi = \text{NIL}$

8)  $Q = \emptyset$

9) Enqueue  $(Q, S)/ Q \leftarrow S$

queue - পূর্ব মধ্যে S push করতে হবে

S.P.F. property  
 ৩টি

start  
 পূর্ব দূরত্ব  
 = 0

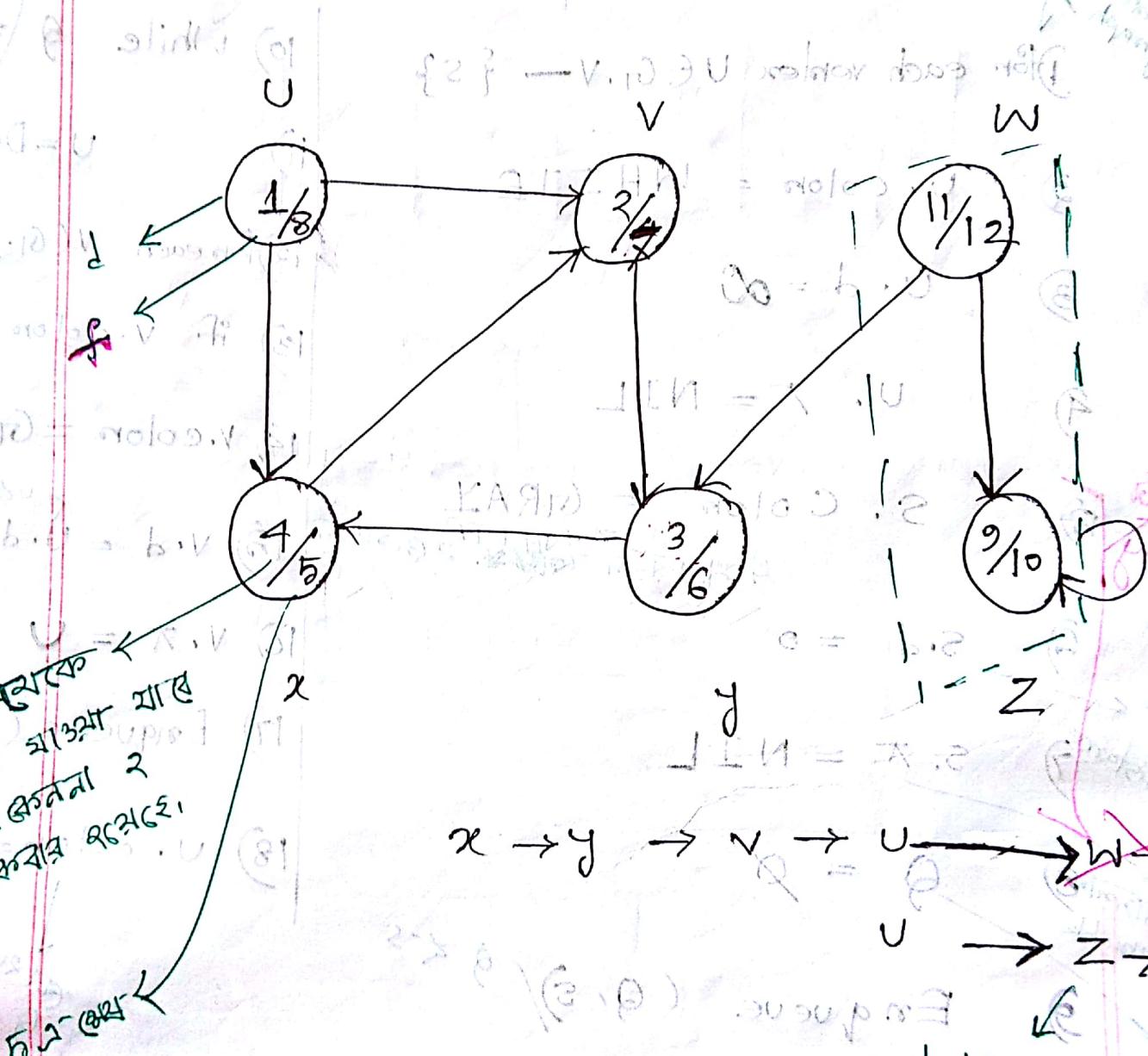
দোয়ান  
 প্রেক্ষণ অবস্থা  
 প্রক্রিয়া

প্রক্রিয়া  
 প্রক্রিয়া  
 প্রক্রিয়া

- পুরো চোক
- pop
- 10) while  $Q \neq \emptyset$   
 $U = \text{Dequeue}(Q)$
  - 11)
  - 12) for each  $V \in G.\text{Adj}[U]$
  - 13) if  $V.\text{color} = \text{WHITE}$
  - 14)  $V.\text{color} = \text{GRAY}$
  - 15)  $V.d = U.d + 1$
  - 16)  $V.\pi = U$
  - 17) Enqueue  $(Q, V)$
  - 18)  $U.\text{color} = \text{BLACK}$

পূর্ব মধ্যে যাবে  
 Q মধ্যে যাবে

## DFS



depth এর কাজ  
আগে রেখে DFS -এ

## Lec - 19

### DFS ( $G$ )

1) for each vertex  $U \in G.V$

2)  $U.\text{color} = \text{WHITE}$

3)  $U.\pi = \text{NIL}$

time = 0

4) for each vertex  $U \in G.V$

if  $U.\text{color} = \text{WHITE}$

DFS - visit( $G, U$ )

graph টা গুরু

### DFS visit( $G, U$ )

1) time = time + 1

2)  $U.d = \text{time}$

3)  $U.\text{color} = \text{GRAY}$   
বাহ্য ঘূর্ণ

4) for each  $V \in G.\text{Adj}[U]$

5) if  $V.\text{color} = \text{WHITE}$

6)  $V.\pi = U$

7) DFS - visit ( $G, V$ )

8)  $U.\text{color} = \text{BLACK}$

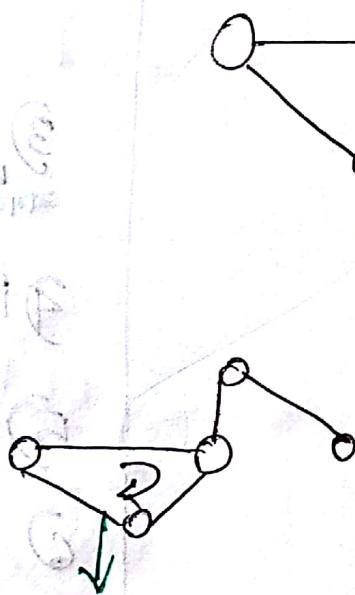
9) time = time + 1

10)  $U.f = \text{time}$

টাওজ নাড়ি  
time কোরি  
কে দ্রুত

## Chapters - 5

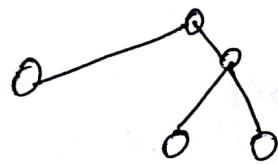
Tree



cyclic graph এ কোন কোণ জায়গাম cycle

Definition with Sig

Forest & Acyclic graph.

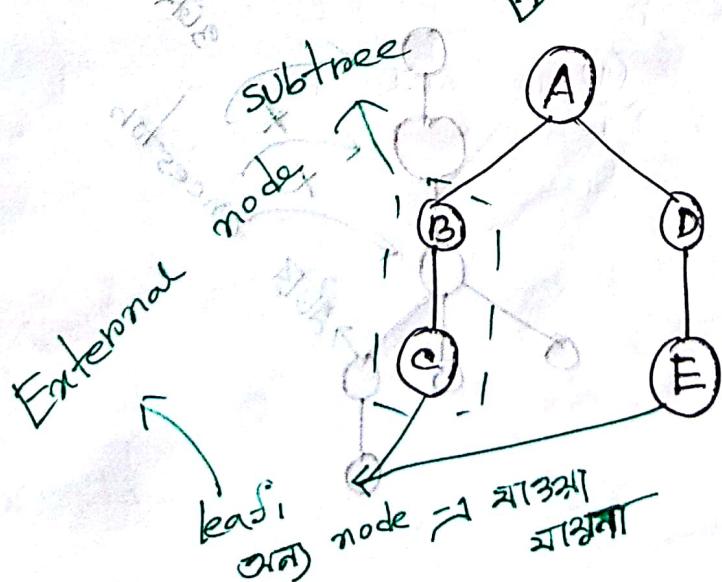
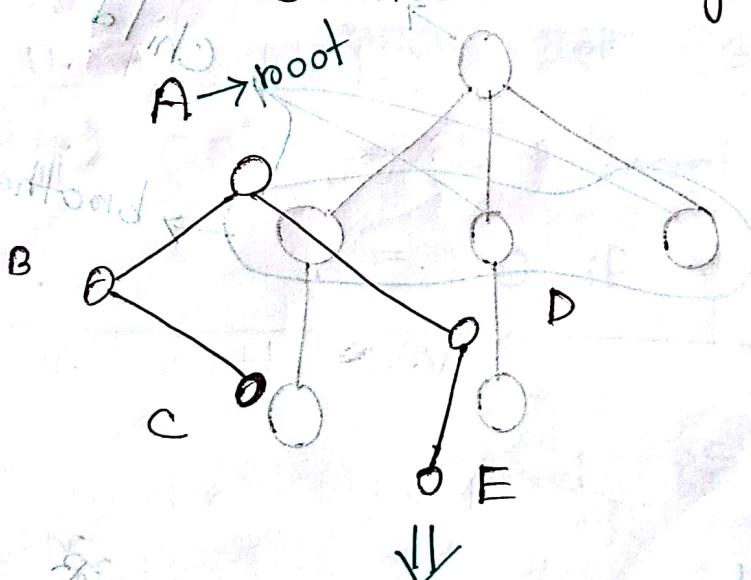


Tree:

Subtree:

tree এর  
একটি অংশ

Connected acyclic graph.



root

degroup silofoA

: head

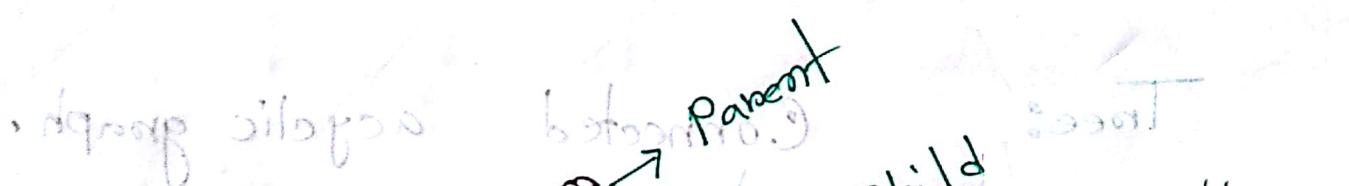
leaf / external node

internal node

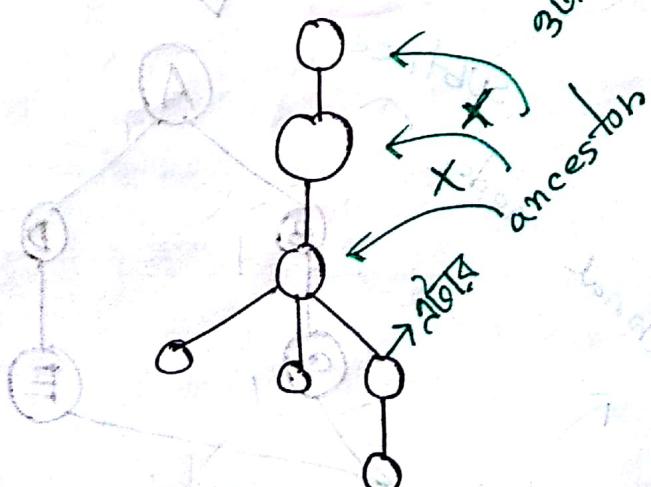
Parent / Father

Child / Son

Sibling / brother



Ancestor



Descendant :

tree টুর বীকে সন্ত

level :

root এবং level 0

এরপরে

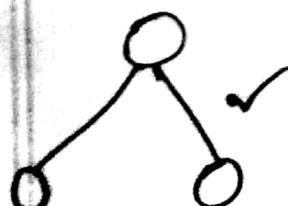


এসময়ের গুলা 3

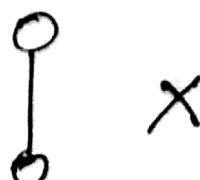
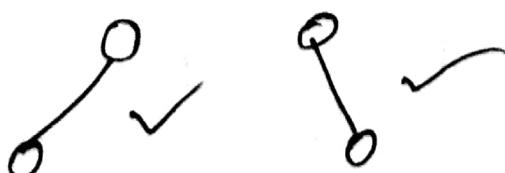
$$\begin{aligned} \text{level } (P) &= 1 + \text{level } (\text{Father } (P)) \\ &= 0 \text{ if } P \text{ is root} \end{aligned}$$

heights মাঝেটি level টোরি height.

binary trees মাত্রি- mode এবং max  
বাঁকা side.



left child right child

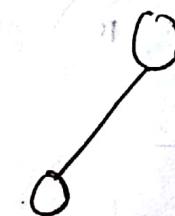


## Complete binary trees

O

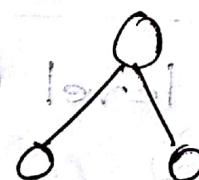
(একটি node নিয়ে  
আরুল)

o level RR focus



(২টি node নিয়ে  
আরুল)

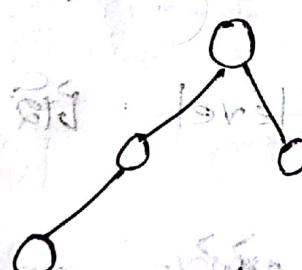
o ত্বরিত RR focus



(৩টি node (১) নিয়ে আরুল)

complete binary  
tree (১০৪)

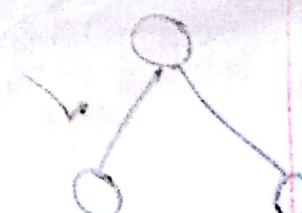
আরুল কারেনা  
অস্টা



(৪টি node নিয়ে  
আরুল)

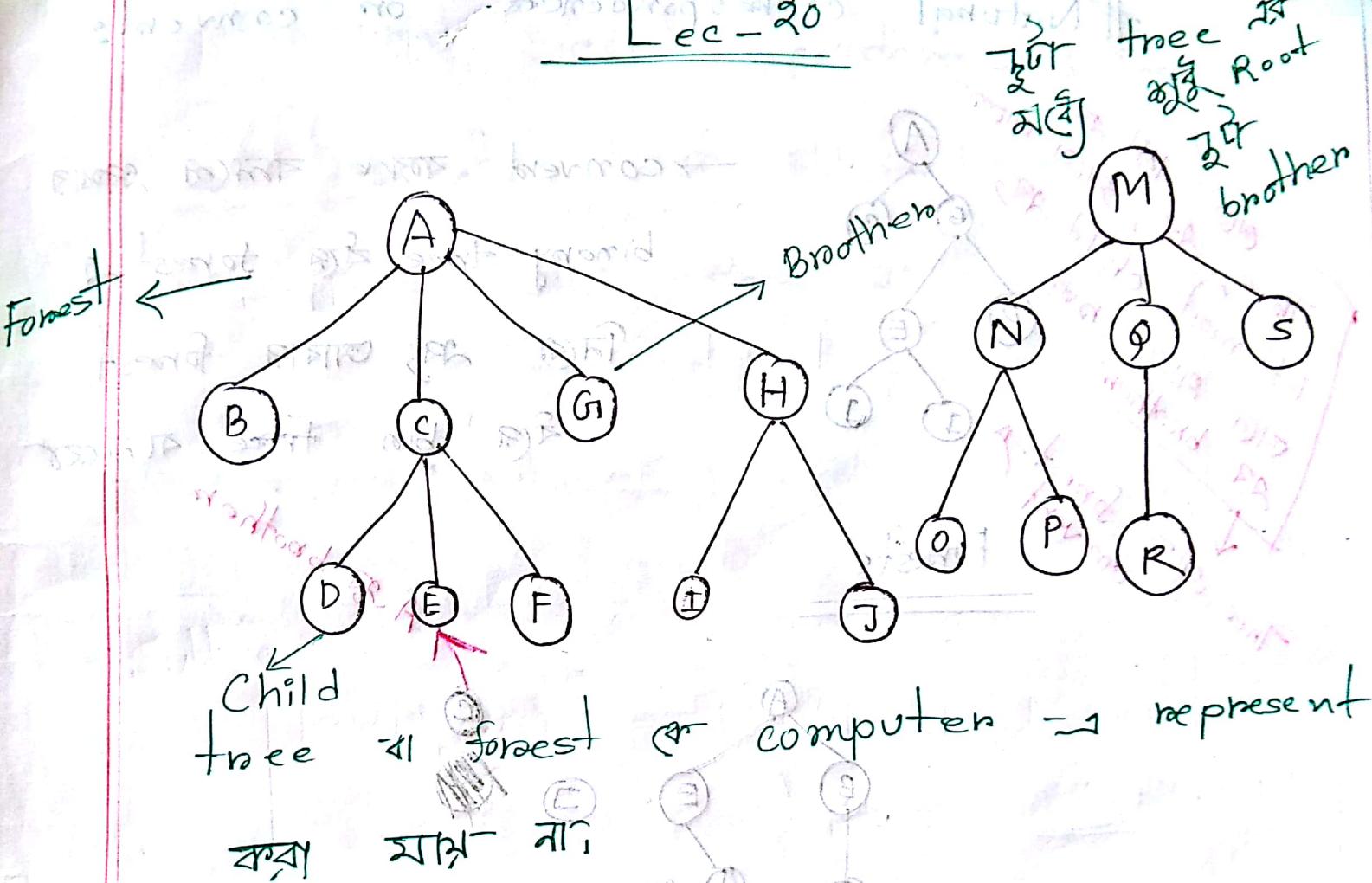
আরুল একটি level

ক্ষেত্র করে যেতে হবে

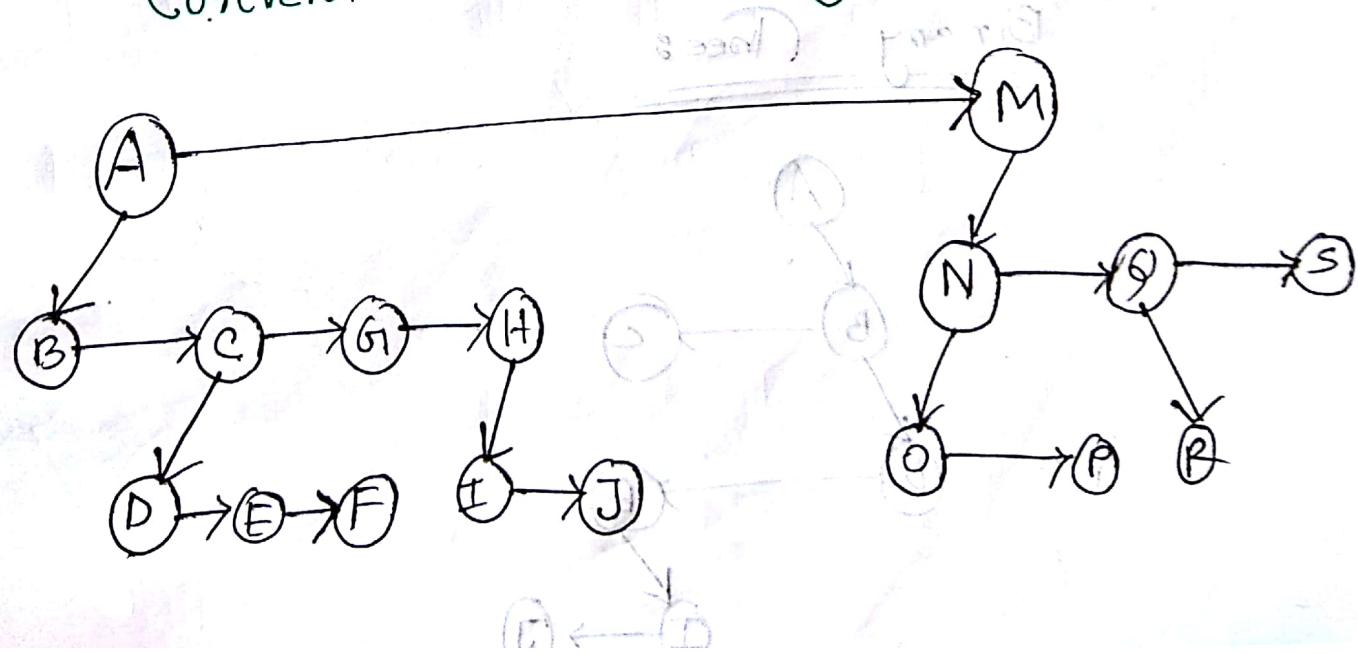


bliflo ১০৮ bliflo ১০৯

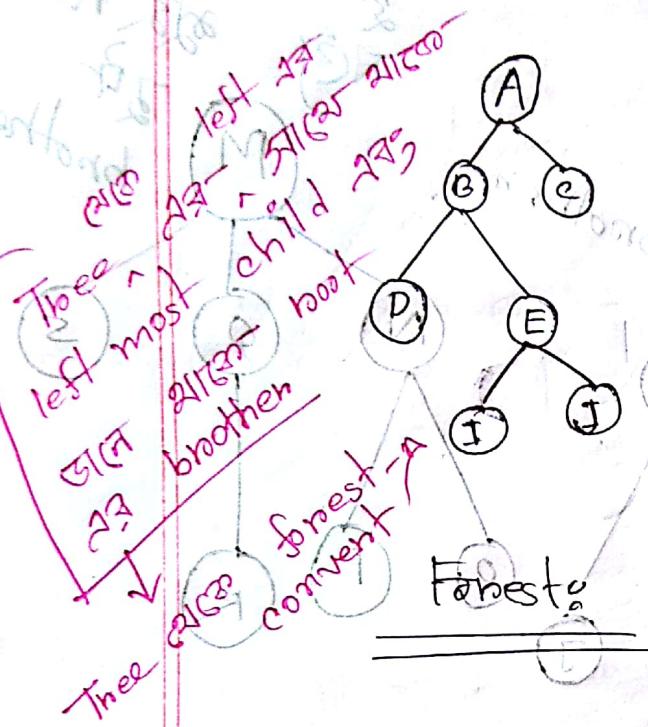
## Lec - 20



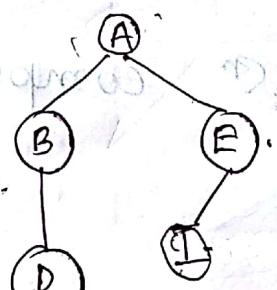
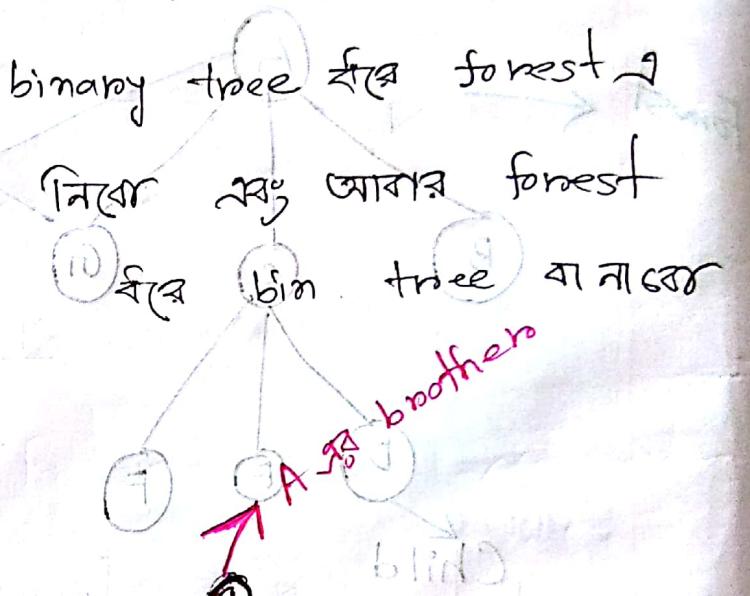
Convert into Binary trees:



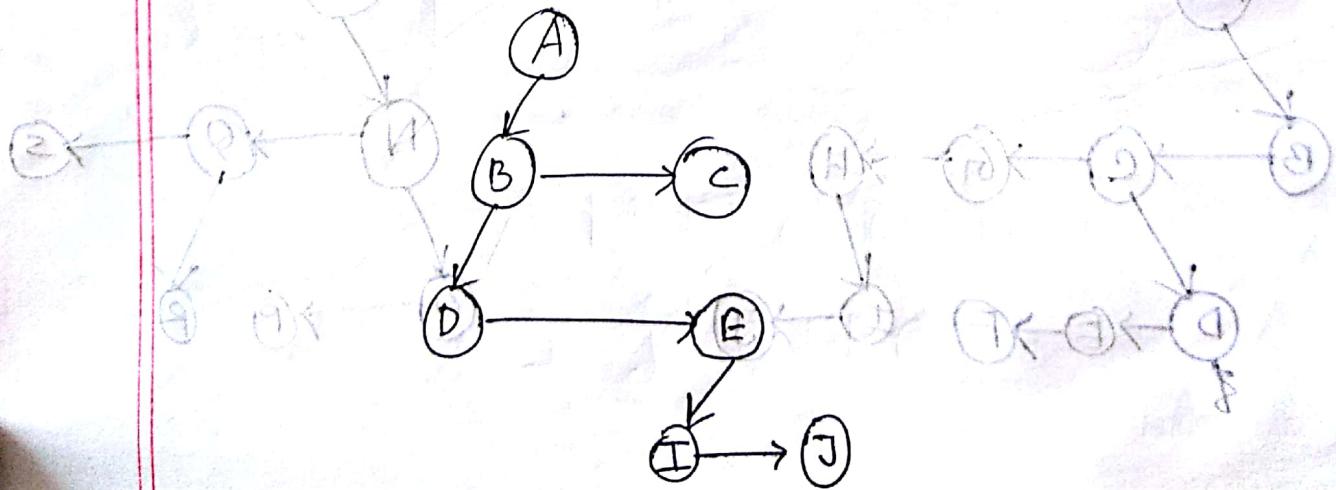
# Natural correspondence or convert.



→ convert বায়েছে বলপে প্রিষ্ঠার



## Binary Trees



Natural correspondence & Tree

বন forest মেল bin tree বাঁজ

bin tree মেল tree at forest

forest - root - বনের মুখ

convert

বন to বিন

বিন সংক্ষিপ্ত

বিন

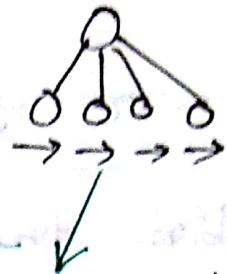
30 ১০ ৩০ ৩০ M A

১০ ৫ I

Lec - 21

Exam / ↪  
algo / traverse

## Traversal



- 1) Preorder : root → left → right
- 2) Inorder : left → root → right
- 3) Postorder : left → right → root

## 3) Level order

### 3) Level Order:

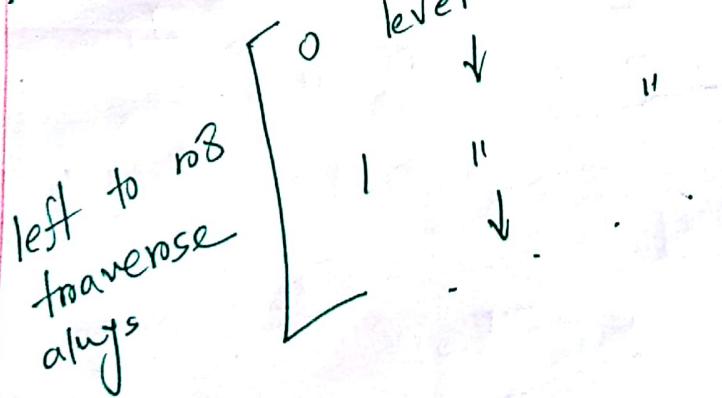
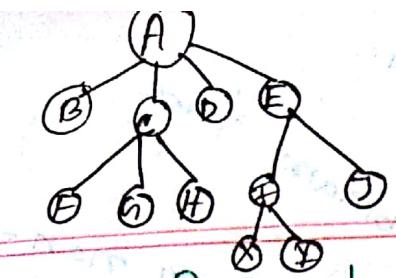


Figure ← A M B C G H N S O P R D E F  
I J

2016/2



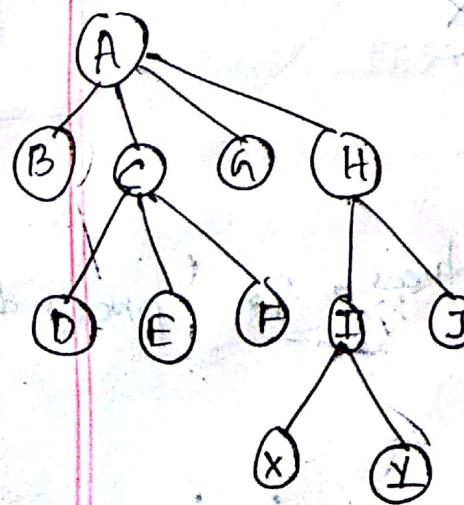
1) Preorder: প্রিওরেড: *(A B C D E F G H I J K)*

B এর নীচে  
নাই কিমু

2) Inorder: অনেক বাবলি traverse এর মত

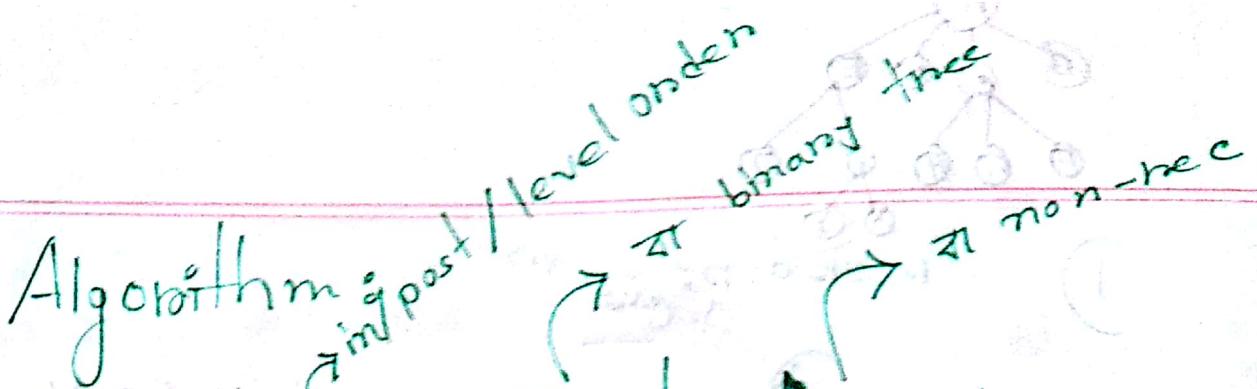
forest/tree " " "  
but binary tree @ স্থানে  
left একটি  
right দ্বিতীয়

3) Postorder:



B D E F C G X Y I J H A ... M

open rigs, M



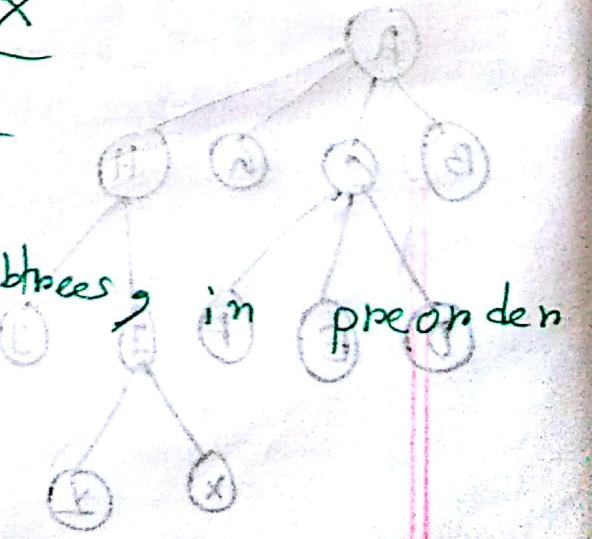
Preorder of Forest: Recursion

- ① Visit the root of the first tree every forest
- ② Traverse the subtrees, if any of the first tree in preorder. rec process
- ③ Traverse the remaining trees if any, in preorder.

tree root 2 3 X

1) Visit the root

2) Traverse the subtrees in preorder



Postorder + Inorder + level

Forecast:

2 at line 1 - 1

1 " " 2 "

3 → line 3 →

Preorder      binary tree      subtree तर्क  
इसे left/right

7) Visit the root

3) Traverse the subtrees of  
in preoder ~~in~~

3 ③ have ~~at~~ ~~line~~ ~~in~~

## Lec - 29

Preorder

Binary Trees

Non-Rec

stack / quev

dn loop

Algo:

$S \leftarrow$  empty stack

$S \leftarrow$  root

while  $S \neq$  empty do

$P \leftarrow S$   
if  $P \neq NIL$   
visit node  $P$

stack  $\rightarrow$  root  
or other  $S \leftarrow$  Right ( $P$ )  
 $S \leftarrow$  Left ( $P$ )

## Level Order Binary Tree

$Q \leftarrow$  empty queue

$Q \leftarrow$  root

while  $Q \neq$  empty do

$P_{gf} \leftarrow Q$   
 $P_{gf} \neq NIL$

visit node  $P$

$Q \leftarrow$  left ( $P$ )

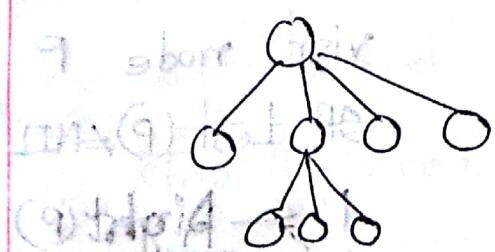
$Q \leftarrow$  right ( $P$ )

## ~~From~~ Level order traversal of a forest

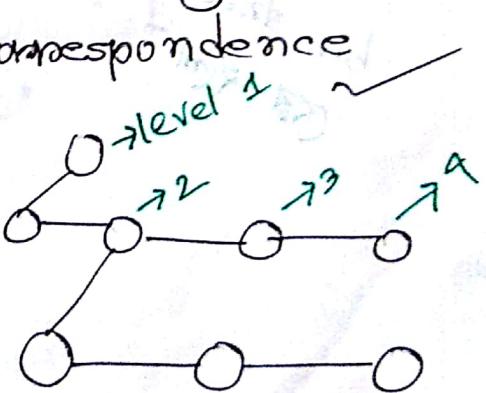
ফুট লেভেল পার্ট করে  
 represented as a binary tree

via natural

Correspondence

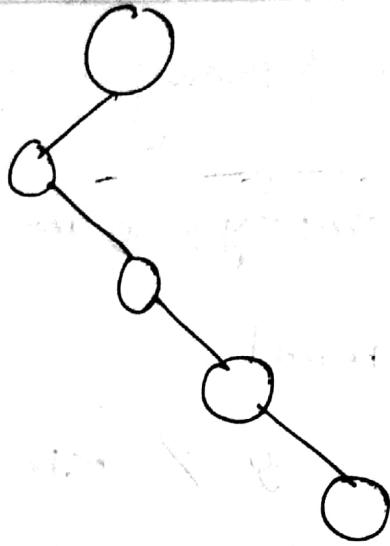


$\Rightarrow$



forest রিমেক নাই

bin tree রিমেক  
আহো



Algo:

```

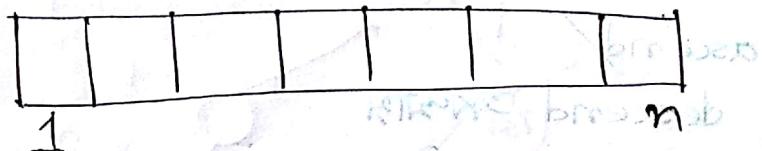
empty Q ←
root P ←
while Q ≠ empty do
    P ← Q
    while P ≠ NIL do
        visit node P
        if Left(P) ≠ NIL then
            P ← Right(P)
            Link
    
```

✓ Quiz 3

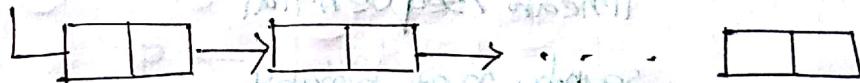
## Chapters - 7

### (Searching)

1)



2)



①

array  
in memory  
data

$i \leftarrow 1$

while  $i \leq n$  and  $z \neq x_i$  do  
 $i \leftarrow i + 1$

if  $i \leq n$  then "Found"  
else "Not Found"

Unsorted

② For Linked Lists

$p \leftarrow L$

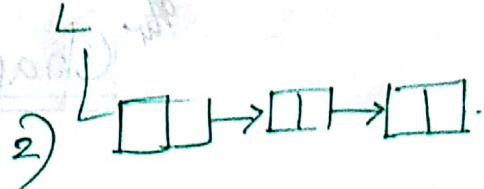
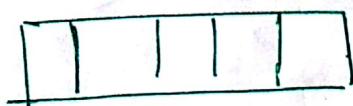
while  $p \neq \text{NIL}$  and  $z \neq \text{key}(p)$

$p \leftarrow \text{Link}(p)$

if  $p \neq \text{nil}$  then "Found"

else "Not Found"

info (of  
linked list)



ascend /  
descend অবস্থান  
স্থানে, স্থির

linear / sequential search: not binary.

Sonated থার্মে

## ମେହି Key ବଳାଇ search

করতে সেটি মেঘানে

ଆଏ ତାର ପର ସେଇଁ break

କାହିଁ ଦିଯୋ, ଆଗେ କୋଡ୍

modify

i ← 1

while  $z > x_i$  do

- 1 -

if  $x_i = z$  then "found"

else "Not Found"

$p \leftarrow L$

$P \leftarrow \text{Link}(P)$

if  $\text{key}(P) = z$  then "Found"

else "NOT FOUND"

## Lec - 23

Sequential search using sorted Linked

List:

$P \leftarrow L$

while  $Z > \text{key}(P)$  do

$P \leftarrow \text{Link}(P)$

if  $\text{key}(P) = Z$  then "Found"

else "NOT Found"

Search in sorted linked lists

Binary Search

$l \leftarrow 1$

$h \leftarrow n$

$\text{found} \leftarrow \text{False}$

while  $l < h$  and not found do

$m \leftarrow \left[ \frac{l+h}{2} \right]$

case :

$Z < x_m : h \leftarrow m-1$

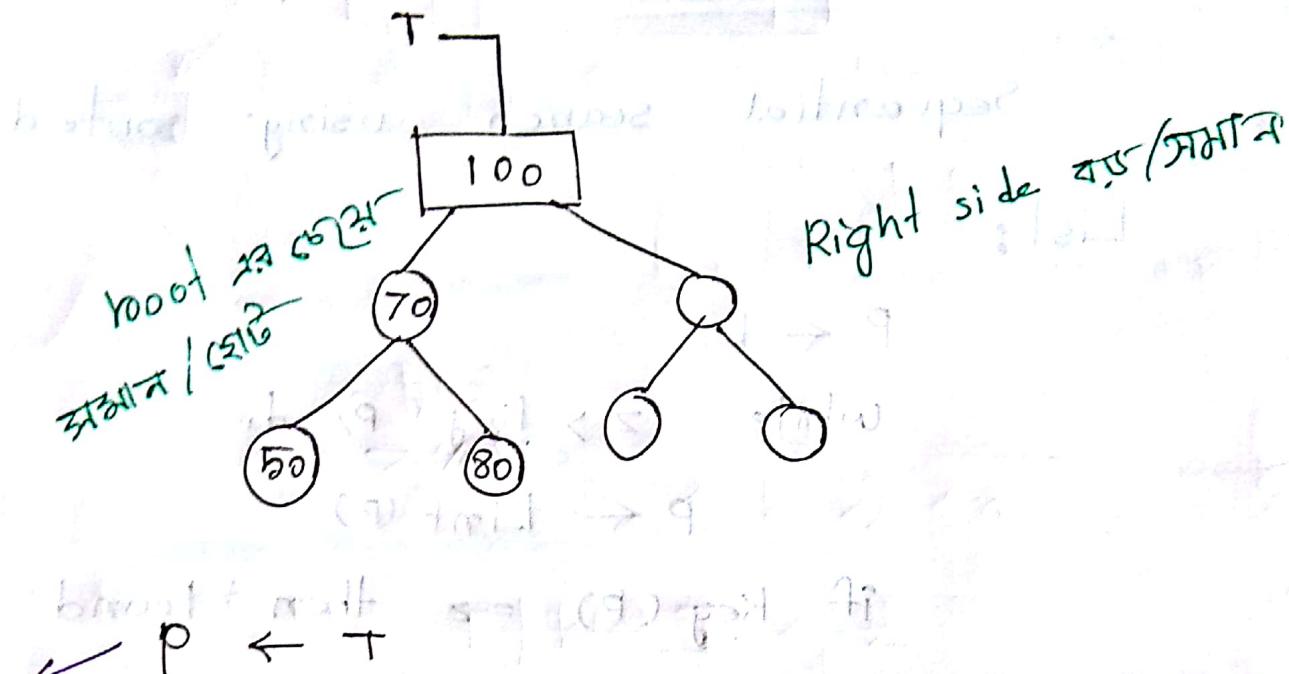
$Z > x_m : l \leftarrow m+1$

$Z = x_m : \text{found} \leftarrow \text{true}$

if  $\text{found} = \text{true}$  then "Found"

else "NOT Found"

## Binary Search Tree:



$p \leftarrow T$

found  $\leftarrow$  false

while  $p \neq \text{nil}$  and not found

case :

$z < \text{Key}(p)$  :  $p \leftarrow \text{left}(p)$

$z > \text{Key}(p)$  :  $p \leftarrow \text{right}(p)$

$z = \text{Key}(p)$  : found  $\leftarrow$  true

if found = true then "Found"

else "Not Found"

আর্গারি search?

Binary search এর number of searches =  $\log_2 n$

(but sorted algo আৰু (০ ১৬৩)

# Binary search & linear search এৱে চেনে কতো time

faster?

⇒ 1 যাৰ search কৰিবলৈ 500

2 " 250

3 " 125

" 62.5

" 31.25

" 15.625

" 7.8125

" 3.90625

" 1.953125

" 0.9765625

" 0.48828125

" 0.244140625

" 0.1220703125

" 0.06103515625

" 0.030517578125

" 0.0152587890625

" 0.00762939453125

" 0.003814697265625

" 0.0019073486328125

" 0.00095367431640625

" 0.000476837158203125

" 0.0002384185791015625

" 0.00011920928955078125

" 0.000059604644775390625

" 0.0000298023223876953125

" 0.00001490116119384765625

" 0.000007450580596923828125

" 0.0000037252902984619140625

" 0.00000186264514923095703125

" 0.000000931322574615478515625

" 0.0000004656612873077392578125

" 0.00000023283064365386962890625

" 0.000000116415321826934814453125

" 0.0000000582076609132674072234375

" 0.00000002910383045633370361171875

" 0.000000014551915228166851805859375

" 0.0000000072759576140834259029296875

" 0.00000000363797880704171295146484375

" 0.000000001818989403520856475732421875

" 0.0000000009094947017604282378662109375

" 0.0000000004547473508802141189331054875

" 0.00000000022737367544010705946652774375

" 0.000000000113686837720053529733263871875

" 0.0000000000568434188600267648866319359375

" 0.00000000002842170943001338244331596796875

" 0.000000000014210854715006691221657983984375

" 0.0000000000071054273575033456110899974921875

" 0.0000000000035527136787516728054499987490625

" 0.00000000000177635683937583640272499997450390625

" 0.00000000000088817841968791820136249999872501953125

" 0.00000000000044408920984395910068124999993625009375

" 0.0000000000002220446049219795503406249999968750048828125

" 0.000000000000111022302460989775170312499999937500244140625

" 0.000000000000055511151230494887585156249999996875001220703125

" 0.0000000000000277555756152474437925812499999993750006109375

" 0.0000000000000138777878076237218962906249999999687500030546875

" 0.000000000000006938893903811860948145312499999999375000152734375

" 0.000000000000003469446951905930474072656249999999968750000763671875

" 0.000000000000001734723475952965237036328124999999999375000038184375

" 0.000000000000000867361737976482618518140624999999999968750000190921875

" 0.000000000000000433680868988241309259070312499999999993750000095464375

" 0.000000000000000216840434494420654629535156249999999999687500000477328125

" 0.000000000000000108420217247210327312767812499999999999375000002386640625

" 0.00000000000000005421010862360516365638890624999999999996875000011933203125

" 0.000000000000000027105054311802581828194453124999999999993750000059666015625

" 0.0000000000000000135525271559012909140972265624999999999996875000029833003125

" 0.00000000000000000677626357795064545704861328124999999999993750000149165015625

" 0.0000000000000000033881317889753227285243065624999999999999968750000074582503125

" 0.00000000000000000169406589448766136426215312499999999999999937500000372912515625

" 0.000000000000000000847032947243830682131076562499999999999999968750000018645625

" 0.0000000000000000004235164736219153410655382812499999999999999937500000093228125

" 0.000000000000000000211758236810457670532779132812499999999999999687500000466140625

" 0.0000000000000000001058791184052288352638895656249999999999999999375000002330703125

" 0.000000000000000000052939555402511417631944782812499999999999999996875000011653503125

" 0.000000000000000000026469777701255708815972394132812499999999999993750000058267503125

" 0.00000000000000000001323488885062785440798619716562499999999999999968750000291337515625

" 0.0000000000000000000066174444253139272039430988328124999999999999999375000014566875

" 0.0000000000000000000033087222126569636019715494165624999999999999999968750000072834375

" 0.00000000000000000000165436110632848180098577470828124999999999999999937500000364171875

" 0.000000000000000000000827180553164240900047888534165624999999999999999968750000182085625

" 0.0000000000000000000004135902765821204500023944270828124999999999999999937500000910428125

" 0.0000000000000000000002067951380410602250011972135416562499999999999999999687500000455214375

" 0.000000000000000000000103397569020530112500598606770828124999999999999999999375000002276071875

" 0.00000000000000000000005169878351026505625002993033504165624999999999999999996875000001138035625

" 0.0000000000000000000000258493917551325281250014965167708281249999999999999999993750000005690178125

" 0.000000000000000000000012924695877566264062500074825835041656249999999999999999996875000002845089375

" 0.00000000000000000000000646234793878313203125000374129175041656249999999999999999993750000014225446875

" 0.00000000000000000000000323117396939156601562500018706458750416562499999999999999999687500000071127234375

" 0.00000000000000000000000161558698469578300781250000935322937504165624999999999999999993750000003556361875

" 0.00000000000000000000000080779349234789150390625000046766148750416562499999999999999999968750000017781809375

" 0.00000000000000000000000040389674617394575019531250000233830543750416562499999999999999999937500000088909046875

" 0.00000000000000000000000020194837308697287509765625000011691527375041656249999999999999999996875000000444545234375

" 0.00000000000000000000000010097418654348643754882812500000584576368750416562499999999999999999937500000022227261875

" 0.00000000000000000000000005048709327219321875244140625000029228818437504165624999999999999999999687500000111136309375

" 0.000000000000000000000000025243546636096609375122265625000014614409218750416562499999999999999999375000000555681546875

" 0.000000000000000000000000012621773318048304687506132812500000730720460937504165624999999999999999996875000000277840774375

" 0.000000000000000000000000006310886659024152343751306640625000003653602304375041656249999999999999999993750000001389203875

" 0.0000000000000000000000000031554433295120761718751653125000001826801156250416562499999999999999999968750000000694601875

" 0.00000000000000000000000000157772166475603858437532656250000009134005781250416562499999999999999999937500000003473009375

" 0.00000000000000000000000000078886083237801929218754130625000004567002890625041656249999999999999999999687500000017365046875

" 0.0000000000000000000000000003944304161890096460937520673125000002283501445312504165624999999999999999999375000000086825234375

" 0.00000000000000000000000000019721520809450482304687510312500000114175072265625041656249999999999999999999687500000004341261875

" 0.000000000000000000000000000098607604047250211523437551562500000057087536328125041656249999999999999999999375000000021706309375

" 0.000000000000000000000000000049303802023625105761718752578125000028543768164062504165624999999999999999999968750000000108531546875

" 0.00000000000000000000000000002465190101181255288031251289062500001427188406250416562499999999999999999993750000000054265734375

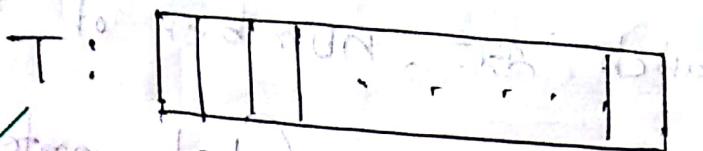
" 0.000000000000000000000000000012325950505906276440156250644531250416562499999999999999999996875000000002713286875

" 0.0000000000000000000000000000061629752529531382200781251322265625041656249999999999999999999937500000013566434375

" 0.000000000000000000000000000003081487626476569110039062526444531250416562499999999999999999999968750000000067832174375

" 0.0000000000000000000000000000015407438132382845550195312551222656250416562499999999999999999999993750000000339160875

" 0.000000000000000000000000000000770371906619142227509765625103125041656249999999999999999999968750000001695804375



array

সংজ্ঞায় পাই

m-1

h is hash function

$$z \% 10$$

$$0 \leq h(z) < m$$

$$T[h(z)] \leftarrow z$$

index

হাশ সহ  $h(z)$  আজ্ঞা মনে Collision

### Lec - 24

Collision Algo

next lec

(01-1) র

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

✓

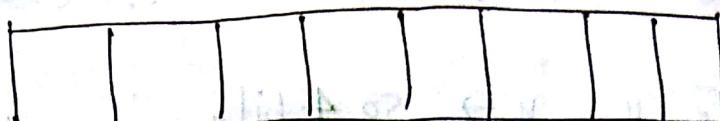
10 टे मर एक 3 bit.

(मर लग 3 bit नित हो)  
but profit data असी same  
3 टा bit

### Extraction Methods

by OR data का माजाह.

# ques :-



0 1 2 3 4 5 6 7

T, H, E आवाज  
word (210)  
Given,

THE :  $\frac{101000100000101}{15 \text{ bit आहे बरु 3 bit तरजावणा आहे}} = 101 = 5$

A : 0 0 0 0 1

OF :  $\frac{00000100000101}{15 \text{ bit आहे बरु 3 bit तरजावणा आहे}} = 0$

B : 0 0 0 1 0

AND :  $\frac{000000000000000}{15 \text{ bit आहे बरु 3 bit तरजावणा आहे}} = 0$

TO :  $\frac{000000000000000}{15 \text{ bit आहे बरु 3 bit तरजावणा आहे}} = 0$

IN :  $\frac{000000000000000}{15 \text{ bit आहे बरु 3 bit तरजावणा आहे}} = 0$

THAT :  $\frac{000000000000000}{15 \text{ bit आहे बरु 3 bit तरजावणा आहे}} = 0$

IS :  $\frac{000000000000000}{15 \text{ bit आहे बरु 3 bit तरजावणा आहे}} = 0$

### Hash Functions

① Extraction

② Compression

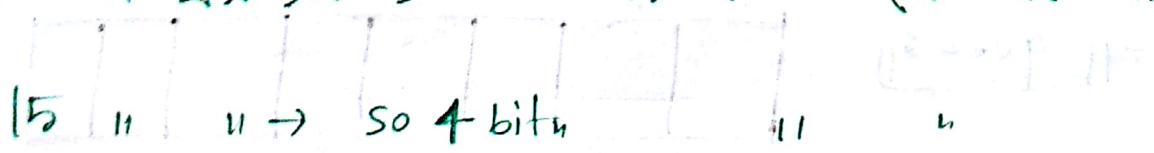
③ Division with  $h(z) = z \mod m$

④ Multiplication  $h(z) = [m (z \mod 1)]$ ; where  $0 < m < 1$

Extraction

AND	A	IN	IS	THAT	THE	OF	TO	*	Ans
0	1	2	3	4	5	6	7		

৬) দি এর  $\rightarrow$  so 3 bit এর বেশি নম্বা মানেন।



15 11 11  $\rightarrow$  so 4 bits 11

## Design Principles (কী কী মনে

রাখতে হবে)

Primary Clustering (একই typ value ইতো একটি group)

Natural

naturally

আলান্ত value হবে

# Data হত কৈমি involve বয়স্কতা কর

কৈমি collision

① the hash function should be a

function of every ~~one~~ bit of the

element.

③ A hash func should break up naturally occurring clusters of elements.

১৮৪ - ৮৯

$$\begin{array}{r} \text{even} = 0 \\ \text{odd} = 1 \end{array}$$

৩) A hash function should be very "quick and easy to compute."

আলোর ques - এই compression method:

#	T	:	1 0 1 0 0
H	I	:	0 1 0 0 0
E	N	:	0 0 0 1 1 (PML)
<hr/>			
1 1 0 0 1			

X-OR  
AND OR / NOT  
result PCF  
but bit involve  
(১০) 26

mod = গুণকৰ্ত্তা

Lec - 25#  $h_1 \downarrow$ 

compression method apply

#  $h_2 \downarrow$ 

Division

#  $h_3 \downarrow$ 

Multiplication

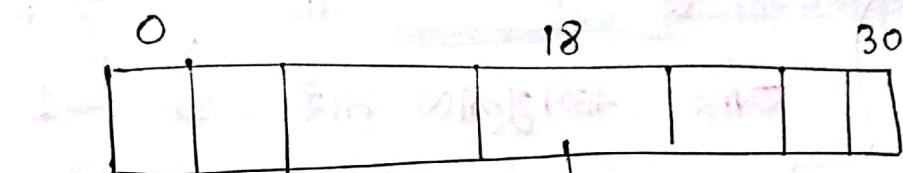
 $m = 31$ Exam Collision :

- ① Separate Chaining
- ② Coalesced "
- ③ Open addressing / Linear Probing
- ④ Double hashing

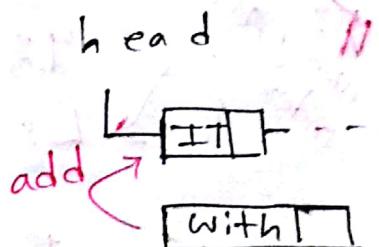
$29 \rightarrow q \text{ vi } z(4)$

page: (336 + 343)

## 1) Separate Chaining: hi 1 column 23 value same value



18  
23  
20



Linked List এর algo: insert func  
head & add  
easy

don't dig too much

যেকোনো অসম্ভব chain

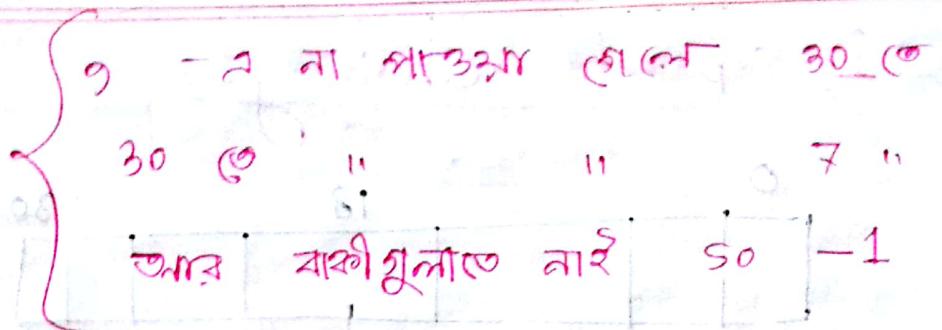
## 2) Coalesced Chaining:

page: (336 + 344)

1	7	9	25	30	THAT
Link	-1	30	-1	.	?

1st collision  
that  $\rightarrow$  30  
Collision রেখে যেকো মুল থানি  
ওঠান্ত রাখবে।

344 -> last pic



Exam => last pic → ans

344 -> last pic final ans

Algo:

location ←  
एक संस्कृत hash  
func use रखें

loc ← h(z)

found ← false

if T[loc] is not empty then

i ← loc

repeat

-> if T[i] = z then found ← true

-> else

prev ← i

Until i ← Link[i]  
found OR i = -1

*data ki AT3Ngi CTCT*  
*AT3AN-CTCT Fat3Ngi*

if not found then  
    if  $T[\text{loc}]$  is empty then  $T[\text{loc}] \leftarrow z$   
    else  
        repeat  $\text{free} \leftarrow \text{free} - 1$  until  $T[\text{free}]$  is empty  
        if  $\text{free} = -1$  then "Overflow"  
        else  
             $T[\text{free}] \leftarrow z$

NULL but int 27  
array so -1  
Link [free] ← -1

Lec - 26

(page 336 + 350)



3 এর নুন

3) ||

hi column



that → 1st collision



যেই ঘর খালি নাই তার পচারের ঘরে স্থান



বিন্দু ঘর খালি না থাবলে 10-এ থাকে

10 "

"

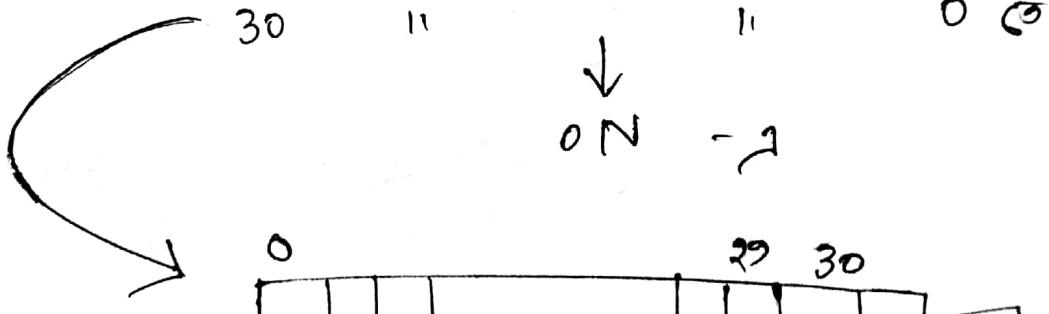
11 "

30 "

"

0 তে যাবে 28 পর্যন্ত

ON - 2



searching 28 মাঝে যাবে  
যাবে তেমনি যেন search করতে  
না 28-এর পরে extra করে  
যদি নিয়ে

# Algo for

1, 2, 3, 4 collision  
method.

# hash func & description

# as data arr

division method 1535

hash func use

↓ (10)

Collision ৩লে কৰ

Collision method (1, 2, 3, 4)

প্রিয়-solve

## Algo

$i \leftarrow h(z)$

found  $\leftarrow$  false

while  $T[i]$  is not empty and not found do

if  $T[i] = z$  then found  $\leftarrow$  true

else  $i \leftarrow (i+1) \bmod m$

if not found then

if  $n = m - 1$  then "Overflow"

else

$T[i] \leftarrow z$

$n \leftarrow n+1$

page (334+353)

### ④ Algo

$i \leftarrow h(z)$

$\Delta \leftarrow \delta(z)$

found  $\leftarrow$  false

while  $T[i]$  is not empty and not found

if  $T[i] = z$  then found  $\leftarrow$  true

else  $i \leftarrow (i + \Delta) \bmod m$

if not found then

if  $n = m - 1$  then "Overflow"

else

$T[i] = z$

$n \leftarrow n + 1$

That.

## Lec - 2 X

### Chapter - 3

A :

13	15	25	2	3	5	17	37	7
1	2	3	4	5	6	7	8	9

X :

1	2	3	4	5	6	7	8	9
---	---	---	---	---	---	---	---	---

X :

4	5	6	9	1	2	7	3	8
---	---	---	---	---	---	---	---	---

# Array এর ডাটা এবং length কে কৈমনি sorting ?

একজে array রে ডাটা নামকে আস্কে পড়ে index

হাত ধরে দিচ্ছি

⇒

মেরি sorting রে লাগ



compare করবো



index num change



পর্তি index sorting

Algo:

for  $t = n$  to 2 by -1 do

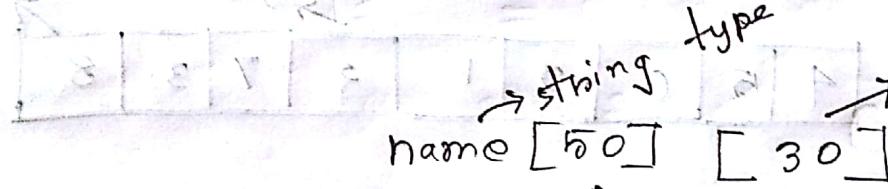
$i \leftarrow 1$

for  $k = 2$  to  $t$  do

if  $A[x[i]] < A[x[k]]$  then

$t \leftarrow k$

$x[t] \leftrightarrow x[i]$



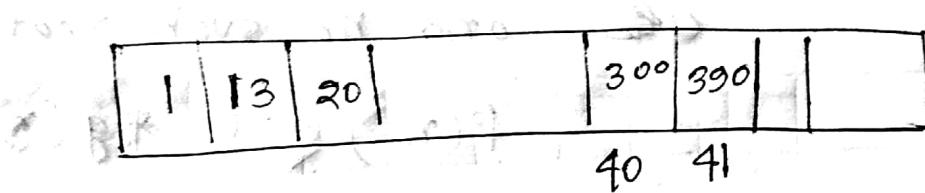
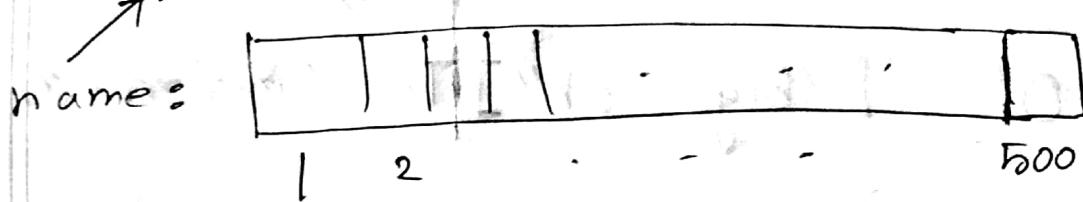
name [50]  $\rightarrow$  string type

[30]  $\rightarrow$  max length

of class name

1	2	3	4	5	6	7	8	9	10
1	2	3	4	5	6	7	8	9	10
10	9	8	7	6	5	4	3	2	1
student									

প্রতি array এর  
name:  $\rightarrow$  arr



# যেকী array এর element এর length vary

বস্তু? তাদের কোণাৰে store কৰিবো?

( আগেৰ ques -এ length কৃত হিলা )

Notations:  $\rightarrow$  ৩ টি  
( 3 marks এর ques  
 $\downarrow$   
definition )

$$n(n+1) = O(n^2)$$

$$2n+5 = O(n)$$

প্রতি notation

"O" [Big "oh"]

①

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

iff there exist positive constants

$c$  and  $n_0$  such that

$f(n) \leq c * g(n)$  for all  $n, n \geq n_0$

Example:

$$3n+2 = O(n)$$

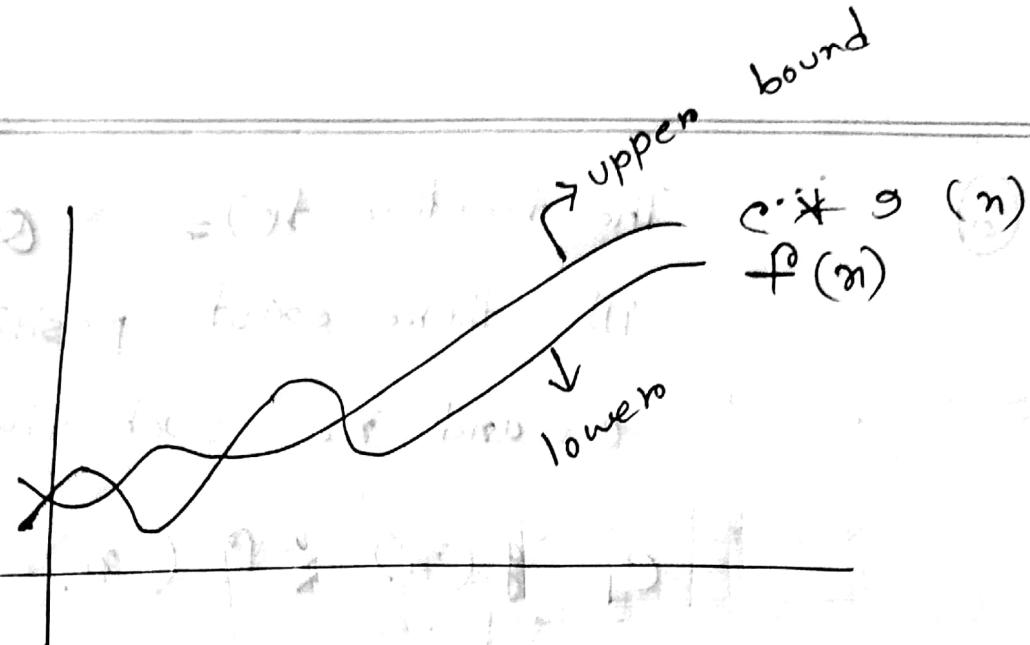
as  $3n+2 \leq c n$

$c = 4$

$g(n) = n$

$f(n) = 3n+2$

1 2 3 4 5 6 7 8 9 10  
2 4 6 8 10 12 14 16 18 20  
3 6 9 12 15 18 21 24 27 30  
3 6 9 12 15 18 21 24 27 30



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

- ② The function  $f(n) = \underline{O}(g(n))$   
 iff there exist positive constants  
 $c$  and  $n_0$  such that  
 $f(n) \geq c * g(n)$  for all  $n, n \geq n_0$ .

③ The function  $f(n) = \Theta(g(n))$   
 iff there exist positive constants  
 $c_1$  and  $n_0$  such that  
 $c_1 g(n) \leq f(n) \leq c_2 g(n)$   
 for all  $n, n > n_0$ .

Ex - 1 (a) ~~notion~~ subject terms select  
chap - 2, 3, 4, 5, 6, 7 (a)  
last due off 6.00

as seen in the last (iv) B is in (v) 4

## Lec - 28

### Chapter - 3

#### Packed Word

age :



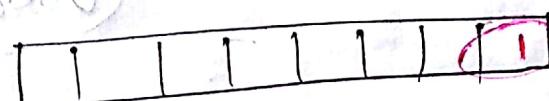
$$\text{age} = 38$$

$$= 120$$

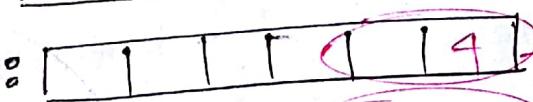
age :



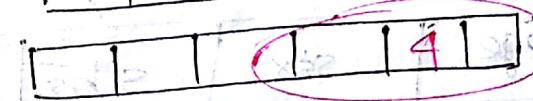
sex :



Siblings :



Kids :



Person :

age	sex	siblings	kids
7	1	4	4

Max bit 25  
31 bits

0101

AND

1111000000

00 . . .

0101 . 00 . .

Sibs → Shift right (person data  $\frac{00F0_{16}}{\downarrow} 16, 9$ )

and  $\frac{00F0_{16}}{\downarrow} 16, 9$  Hex value

Sex → Shift right (person - data AND  $0100_{16}, 8$ )

age → n of (01 11 AND FE 00 $_{16,9}$ )

AND

0100

\_\_\_\_\_

\_\_\_\_\_

0000 0001 0000 0000

OR =

Sex → shift-right

(0000 0001 0000 0000) $_2, 8$

#	person:	age	sex	sibs	kids
---	---------	-----	-----	------	------

11 11 11 11 11 11

0000 0000

11 11 11 11 11 11

110 0000

AND

E 0

sibs ← Shift - right ( person data AND

0018 $_{16, 3}$ )

sex { , 8 )

age { , 9 )

## Packing:

Person :

age	sex	Kids	Sibs
-----	-----	------	------

7 1 ↑ 4 84

Sibs :

1	1	1	
---	---	---	--

AND 3  
কার্ড

left shift এরপে + OR করতে হবে 0  
speciate এরপে) কার্ড - নিম্ন হবে

Person :

--	--	--	--

7 1 ↑ 4 84

sibs :

1	1	1	
---	---	---	--

Person - data ← Shift - left (age, 9)

age এর প্রস্তুত কর্তৃত  
shift

OR Shift - left (sex AND 116, 8)

OR Shift - left (siblings AND F16, 4)

OR Shift - left (kids AND F16)

প্রক্রিয়াজটি



value field

dec      int →

# 0101011101001000

→ Integer : 22344

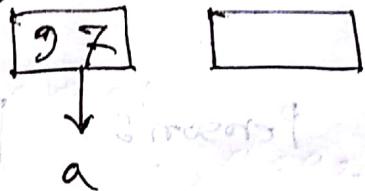
Logical  
and Hex  
value

Logical : 5748

convert তালিতে রয়ে

A - Z  
a - z  
0 - 9

Character strings



মুক্তি, এবং  
বাইনে বাইনে  
আনন্দ নিয়ে  
না

# (10011)<sub>2</sub> calculate it.

$$10011 \rightarrow 1 \times 2^4 + 0 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$$

$$1 \times 2^1 + 1 \times 2^0$$

$$\cancel{1} \quad (10011)_10$$

$$\Rightarrow (1 \times 10^4 + 0 \times 10^3 + 0 \times 10^2 + 0 \times 10^1$$

$$= 10000 + 10 + 1$$

$$= 10011$$

$$= 10000 + 10 + 1$$

$$= 10011$$

$$= 10000 + 10 + 1$$

$$= 10011$$

$$= 10011$$

Loc. 29

Bim to Dec;

(10011) 9

$$= 1 \times 2^4 + 0 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$$

Dec to Dec 2000

$$(10011)_{10}$$

$$= 1 \times 10^4 + 0 \times 10^3 + 0 \times 10^2 + 1 \times 10^1 + 1 \times 10^0$$

= 100 11

b base ଏହି ଶକ୍ତି ନମ୍ବର୍ସ:

$$a_{K-1} \ a_{K-2} \ a_{K-3} \ \dots \ a_3 a_2 a_1 a_0$$

$$= a_{K-1} x b^{K-1} + a_{K-2} x b^{K-2} + a_{K-1} x b^{K-1} \xrightarrow{b=1} \\ + \dots + a_3 b^3 + a_2 b^2 + a_1 b^1 + a_0 b^0 \dots \textcircled{1}$$

$$= (a_{k-1} b^{k-2} + a_{k-2} b^{k-3} + \dots + a_2 b + a_1) b + a_0$$

$$= (c a_{k-1} b^{k-3} + a_{k-3} b^{k-4} + \dots)$$

$$+ a_3(b+a_2)(b+a_1)(b+a_0)$$

$$= ((\dots((a_{k-1}b + a_{k-2})b + a_{k-3})b + \dots + a_2)b + a_1)b + a_0$$

— (11)

(digit  $\times$  base + next digit) base ..

eq<sup>n</sup> (1) is called Horner's Method.

Q11 Derive Horner's Method?

# (10011)<sub>2</sub> traditional way Horner's Method - value করে (5213)

Multiplication

$$= 1 \times 10^4 + 0 \times 10^3 + 0 \times 10^2 + 1 \times 10^1 + 1 \times 10^0$$

$$= 1 \times 10 + 10 \times 10 +$$

# Horner's method এর পদ্ধতি multiplication?  $\Rightarrow (10011)_2$

$$= (((((1 \times 10 + 0)10 + 0)10 + 1)10 + 1)10 +$$

4 bits  
multi

Signed Integers

2's complement:

-8 to +7

1's complement:

$$01111111 = -7 + 6 + 7 = 8$$

0000

ତୀର୍ତ୍ତ ହାଫ୍ ନିମ୍ନଲିଖିତରେ,

0001

0000, 1111 overlap କରିବାରେ

0010

ଶାଖାବଳୀ  $0000 = +0, 1111 = -0$

0011

ଯେତେବେଳେ ଏହାଟି ନିମ୍ନଲିଖିତ ହେବୁ

0100

0101

0110

0111

1000

1001

1010

1011

1100

1101

1110

1111

1000

1001

1010

1011

1100

1101

1110

1111

Negative ଶତ୍ୟ ସଂଖ୍ୟାର କାର୍ଯ୍ୟକାରୀ ପାଇଁ:

① 2's complement:  $-3 \Rightarrow 2 \rightarrow 0010 \rightarrow 1101 \Rightarrow 1110$

2's complement eq'n:

$$-n = 2^K - n ; K = \text{number of bits represented}$$

$$-3 = 2^4 - 3 = 16 - 3 = 14 = 1110$$

$\swarrow$  1's com  $\searrow$  2's com  $\downarrow$

Bin

# Float থেকে ১০ bit  $\rightarrow$  bin  $\rightarrow$  কিমি?

$\Rightarrow$  Float

Float to binary

$$2.5 = 10.1$$

$$2^4 \quad 2^3 \quad 2^2 \quad 2^1 \quad | \quad 2^0 \quad 2^{-1} \quad | \quad 2^{-2} \quad | \quad 2^{-3} \quad 2^{-4}$$

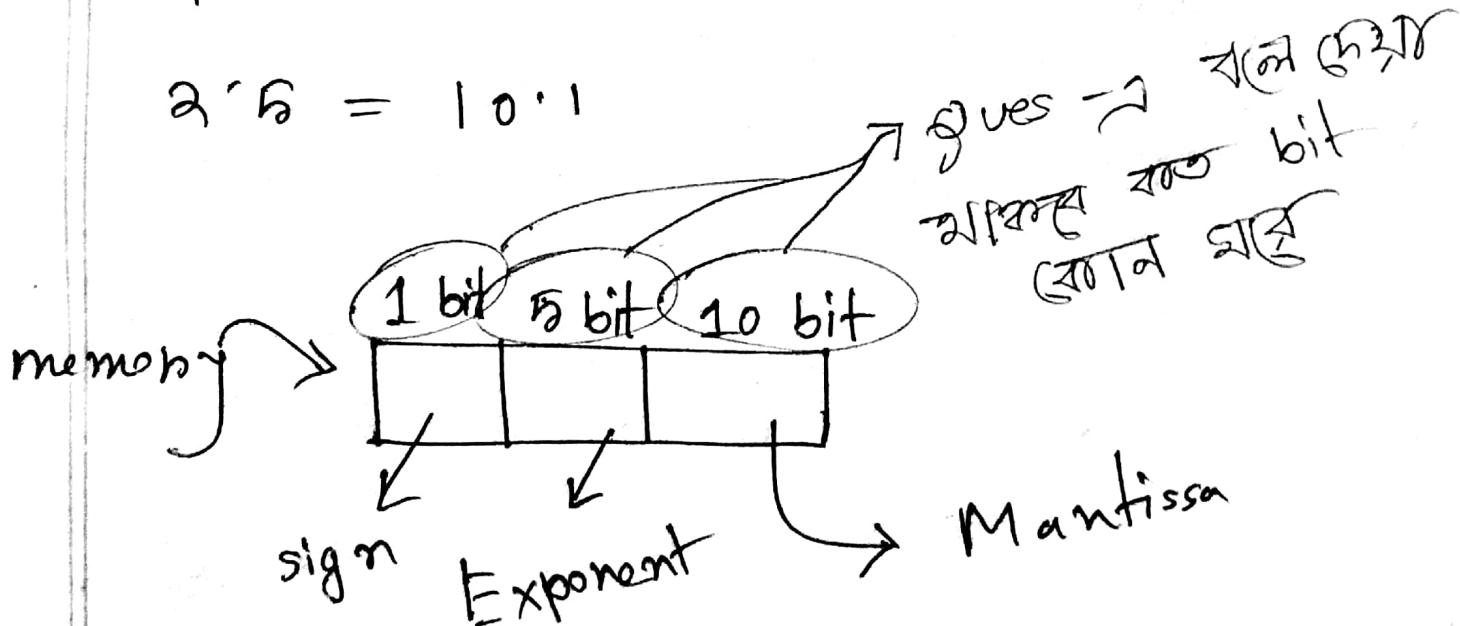
$$16 \quad 8 \quad 4 \quad 2 \quad 1 \quad .5 \quad .25 \quad .125 \quad .625$$

$$13 = 0.11011$$

$$0.1110$$

$$2^4 = - - -$$

$$2^5 = 10.1$$



$$12.35 = 1235 \times 10^{-2}$$

$$= .1235 \times 10^2$$

$$2.5 = 10.1 = + .101 \times 2^2$$

পুরো  
ক্রম  
করা  
পদক্ষেপ

0	000101	1010000000
---	--------	------------

power minus করা | 2's complement

→ এখন এটি

2) 1's complement:  $-3 \Rightarrow 2 \Rightarrow 0010 \Rightarrow 1101$

$$-n = 2^k - n - 1$$

$$-3 = 2^4 - 3 - 1 = 16 - 2 - 1 = 13 = 1101$$

### 3) Sign Magnitude:

$$-2 \Rightarrow \boxed{\phantom{0} \phantom{0} \phantom{0} \phantom{0}} \Rightarrow 1010$$

↑  
প্রথম bit sign এর জন্য

0 মানে pos  
1 " neg

$$-n = 2^{k-1} + n$$

$$-3 = 2^{4-1} + 3 = 8 + 3 = 10 = 1010$$

4) Offset: 0000 মানে -8

$2^{k-1} - n$
$-3 = 2^3 - 3 = 8 - 3 = 5 = 0110$
$= -2$

;

1111 মানে 0

Ques || চার বক্স পদ্ধতির description? eq^n

বিষয় value?

Ques || 4 bit দিয়ে অস্তুলা সংজ্ঞা লিখা  
যাচ্ছ- অস্তুলা হোট (থেকে বড় মাঝারি) (+ve)

৩ (-ve) সব একসাথে আছে,

## Lec - 30

### ~~gm~~ ~~Chapter 3~~

page - 92, 94 Ex - 2, 6, 7(a)

page - 117, 118 Ex

~~- 1, 5~~



Upper

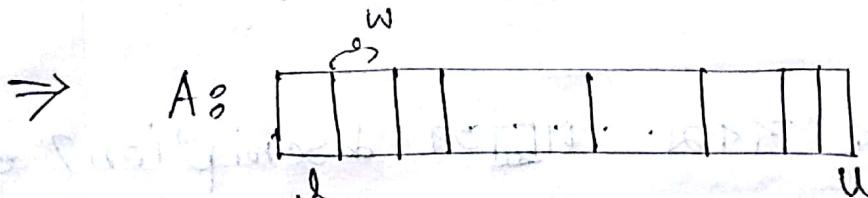
lower

triangular

matrix

Find  
N dimensional  
array.  
memory  
position

$\Rightarrow A \text{ is array } [l : u] \text{ of items of length } w$



ব্যবহার

$$(u-l+1)w$$

$$u-l+1$$

memory

গ্লোবাল  
ফার্স

Mark

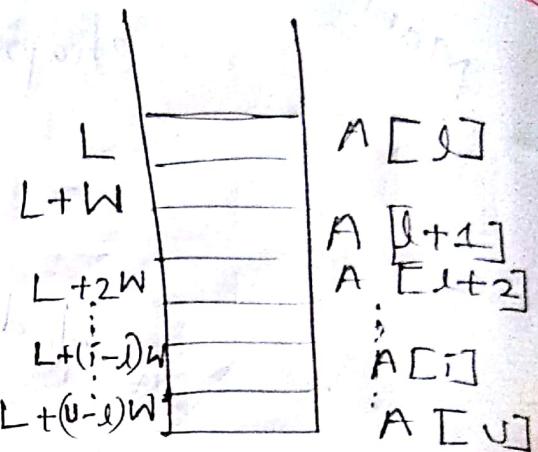
distribution

data struc & definition

chap 1 + 3 → 1 6r ques

today's lecture

- a)
- b)
- c)



$$\text{loc}(A[i]) = \text{loc}(A[l]) + (i-l)w \quad \dots \quad (1)$$

1 dimensional array

$A[1, 1]$	$A[1, 2]$	$\dots$	$A[1, m]$
$A[2, 1]$	$A[2, 2]$	$\dots$	$A[3, m]$
:	:		:
$A[n, 1]$	$\dots$	$\dots$	$A[n, m]$

two dimensional array

$n \times m \times w$

$$\text{loc}(A[i : j]) = \text{loc}(A[i]) + (j - i)w - 1$$

*2 dimensional*

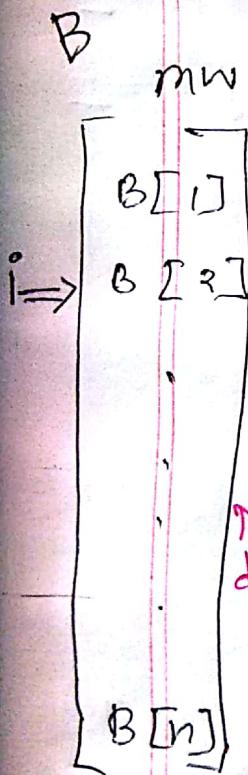
$$\text{loc}(B[i : j]) = \text{loc}(B[1]) + (i - 1)mw$$

*convert 1 to 2 dimensional*

$$\text{loc}(A[i, j]) = \text{loc}(B[1]) + (i - 1)mw + (j - 1)w$$

$$= \text{loc}(A[1, 1]) + (i - 1)mw + (j - 1)w$$

— ⑪



*N dimensional*

$A[l_1 : u_1, l_2 : u_2, \dots, l_n : u_n]$

$$\text{loc}(A[i_1, i_2, \dots, i_n]) = \text{loc}(\hat{A}[i_1, i_2, \dots, i_{n-1}]) + (i_n - l_n)w$$

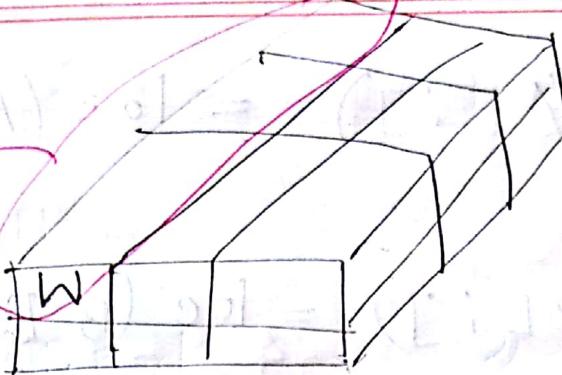
$$= \text{loc}(\hat{A}[i_1, i_2, \dots, i_{n-2}])$$

$$+ (i_{n-1} - l_{n-1})(u_n - l_{n+1})w$$

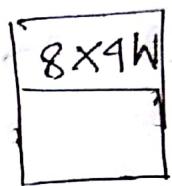
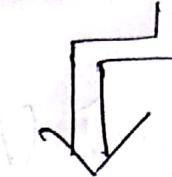
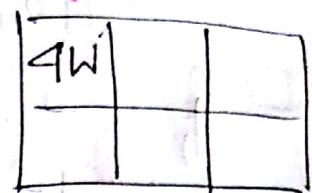
$$+ w \{ \begin{matrix} (i_n - l_n)w \\ + \end{matrix} \}$$

*for 2D matrix*

3 dimensional array  $\rightarrow$   $24 \times 3 \times 1$



$4 \times 3 \times 3$   
Final 4W



⑪

and  $\hat{A}$  is a diagonal matrix with 1's on the diagonal.

$(\alpha_1 - \alpha_i) + (\beta_{1-i} - \beta_1)$  sol =

$(\alpha_1 - \alpha_i) + (\beta_{1-i} - \beta_1)$  sol =

$\alpha_1 - \alpha_i + \beta_{1-i} - \beta_1$  sol =

$\alpha_1 - \alpha_i + \beta_{1-i} - \beta_1$  sol =

Successor → আগস্ট  
predecessor → মেজুন্দুর

Lec - 31

# Binary Search Tree

Chapters ←  
S1G2  
S1G3, S1G4  
Searching chapters  
Search →  
Tree

Man

Minimum ( $x$ )

1) while ~~x.left~~ ≠ NIL

1) while ~~x.left~~ ≠ NIL

$$x_i = x_i : \text{left} \quad \text{Right}$$

2)  $\frac{1}{x} = \frac{1}{x-3}$  triple,  $x = 0$  setze

③ return n

## # Tree - Successor ( $x$ )

④ if  $x.\text{right}_{\text{list}} \neq \text{Nil}$

② return Tree - Minimum ( $x.\text{right}$ )  
Max ( $x.\text{left}$ )

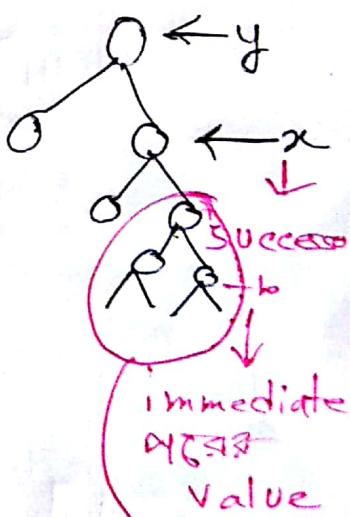
$$③ \quad -\cancel{y} = x \cdot P$$

while  $y \neq \text{NIL}$  and  $x == y$ , height  
 $x = y$   
 $H = H + 1$

$$x = y$$

$$y = y_p$$

return y



immediate  
AT&T

Value

ପ୍ରାଚୀମନ୍ଦିର

Ch. 6  
be sides?

३६७

## Succession

108 side

ମାଧ୍ୟମିକ

ପ୍ରକାଶକ୍ତି

୫୯ ମର୍ତ୍ତ୍ତା

ପ୍ରମାଣ

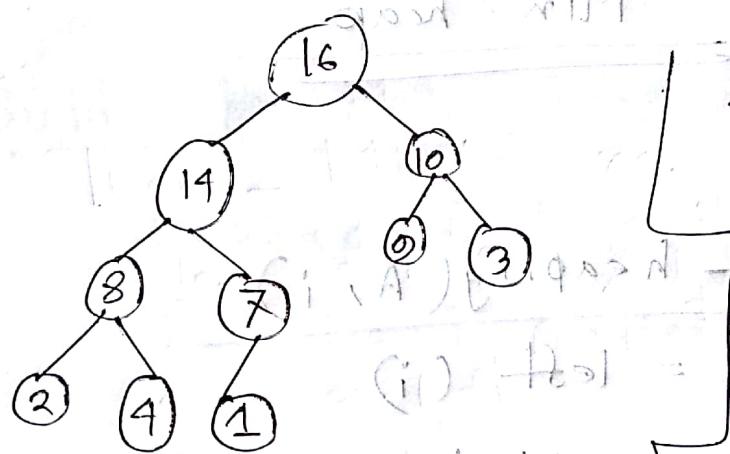
## # Tree - Insert ( $T, z$ )

- 1)  $y = \text{NIL}$
- 2)  $x = T.\text{root}$
- 3) while  $x \neq \text{NIL}$
- 4)  $y = x$
- 5) if  $z.\text{key} < x.\text{key}$
- 6)  $x = x.\text{left}$
- 7) else  $x = x.\text{right}$
- 8)  $z.P = y$
- 9) if  $y = \text{NIL}$
- 10)  $T.\text{root} = z$
- 11) else if  $z.\text{key} < y.\text{key}$
- 12)  $y.\text{left} = z$
- 13) else  $y.\text{right} = z$

## Lec - 32nd XM

Max-heap sorting is good

16	14	10	8	7	9	3	2	4	1
----	----	----	---	---	---	---	---	---	---



Parent

i

return  $\frac{n}{2}$

left

i

return  $2i + 1$

# Ques: 3  
no  
node  $\Rightarrow$  left  
right child  
(কোথায়?)  
index i

# array

Max-heap

বাস্তু

A[i], A[2i],  
A[2i+1] এর-

বড় ভি বড় - পৰে  
parent থেকে

## Max heap

যে heap এ parent এর value child

এর মাঝে বড় যা হবেন,

## Min - heap

Do it  
yourself

1)  $l = \text{left}(i)$

2)  $r = \text{right}(i)$

3) If  $l \leq \text{A.heapsize}$  and  $A[i] > A[l]$

4) Largest =  $A[l]$

5) else largest = i

6) if  $r \leq \text{A.heapsize}$  and  $A[r] > A[\text{largest}]$

7) largest = r

- 8) if  $i \neq \text{largest}$
- 9) exchange  $A[i]$  with  $A[\text{largest}]$
- 10) Max-heapify ( $A$ ,  $\text{largest}$ )

### Build Max-heap ( $A$ )

- 1)  $A.\text{heapsize} = A.\text{length}$
- 2) for  $i = \lceil A.\text{length}/2 \rceil$  down from to 1
  - Max-Heapify ( $i$ )

### Heap sort

- 1) Build - Max - Heap ( $A$ )
- 2) for  $i = A.\text{length}.$  down to 2
  - exchange  $A[i]$  with  $A[1]$
  - $A.\text{heapsize} = A.\text{heapsize} - 1$
  - Max-heapify ( $A, 1$ )