24.09.16
শনিবার

bubble sort

```
  i
5 8 3 7
    i
8 5 3 7
```

Lab

Assignment 1

1. Insertion sort
2. Selection sort

24.04.16 রবিবার

CSE 2015: Data Structures

A data structure can be defined informally as an organized collection of values and a set of operations on them:-

* Data structures have three components:-

1) A set of functions
2) A storage structure
3) A set of algorithms

Algo

```
k ← n                          (Assign; n = number of element)
while k ≠ 0 do                 (number of comparisons)
    t ← 0                      [3-1 = 2] [1] [0]
    for j = 1 to k-1 do
        if x_j > x_{j+1} then
            x_j ↔ x_{j+1}
            t ← j
    k ← t                      (k = 3/2/0)
```

Code

```
x[j] > x[j+1]

temp = x[j]
x[j] = x[j+1]
x[j+1] = temp
```

26.04.16 মঙ্গলবার

Sorting

1) Bubble sort:

iteration / Pass 1 ⇒ Largest value comes to end

n = 5

j = 1

8 7 5 1 12

7 8 5 1 12

7 5 8 1 12

7 5 1 8 12

(swaping value; t = 1/2/3)

iteration / Pass 2 ⇒ do bubble sort till swaping val[ue]

t = 1/2

7 5 1 8 12

5 7 1 8 12

5 1 7 8 12

interation / Pass-3

$t=1$

5 1 7 8 12

1 5 7 8 12

② Insertion Sort

$x_0 \leftarrow -\alpha$
for ($j=2$ to n do
  $i \leftarrow j-1$
  $t \leftarrow x_j$
  while $t < x_i$ do
    $x_{i+1} \leftarrow x_i$
    $i \leftarrow i-1$
  $x_{i+1} \leftarrow t$

int min

(int min / int max)

Pass

x_0	x_1	x_2	x_3	x_4	x_5
$-\alpha$	8	7	2	4	6

compare (if 8 > 7 ⇒ $x_2 = x_1$; (if < ⇒ index)

copy

int min

maximum index value

$t=7$

x_0	x_1	x_2	x_3	x_4	x_5
$-\alpha$	8	7	2	4	6
$-\alpha$	8	8	2	4	6
$-\alpha$	7	8	2	4	6
$-\alpha$	7	8	8	4	6
$-\alpha$	7	7	8	4	6
$-\alpha$	2	7	8	4	6
$-\alpha$	2	7	8	8	6
$-\alpha$	2	7	7	8	6
~~$-\alpha$~~	~~2~~	~~4~~	~~7~~	~~8~~	~~6~~
$-\alpha$	2	4	7	8	8
$-\alpha$	2	4	6	7	8

→ (if $-\alpha > 7$ n
else $x_0 = x_1$

⑦

$t=2$

$t=6$

8 7 2 4 6
8 8 2 4 6
7 8 2

interation / Pass-3

t=1

5 1 7 8 12

1 5 7 8 12

② Insertion Sort

```
x0 ← -α
for( j=2 to n do
    i ← j-1
    t ← xj
    while t < xi do
        x(i+1) ← xi
        i ← i-1
    x(i+1) ← t
```

int min

int min
int max

Pass

x_0	x_1	x_2	x_3	x_4	x_5
-α	8	7	2	4	6

int min

compare (if 877 ⇒

if < ⇒

maximum index value

t=7

x_0	x_1	x_2	x_3	x_4	x_5
-α	8	7	2	4	6
-α	8	8	2	4	6
-α	7	8	2	4	6
-α	7	8	8	4	6
-α	7	7	8	4	6
-α	2	7	8	4	6
-α	2	7	8	8	6
-α	2	7	7	8	6
~~-α~~	~~2~~	~~4~~	~~7~~	~~8~~	~~6~~
-α	2	4	7	8	8
-α	2	4	6	7	8

→ (if —

else

t=2

87246
88246
782

t=6

27.09.16 ⇒ बुधवार

Much longer

③ Selection Sort

find
① maximum
② Last index interchange

Algorithm :-

```
for(j = n to 2 by -1 do
    t ← 1
    for k → 2 to j do
        if x_t < x_k then
            t ← k
    x_j ↔ x_t   // swaping
```

index2 — decrement — for(j=n; j>=2; j--) — condition — index

j=7, j=6, j=5, j=4, j=3

① Max1 77 33 44 11 Max2 88 22 66 55

② Max1 77 33 44 11 55 22 66 88

Max1 66 33 44 11 55 22 77 88

Max1 22 33 44 11 55 66 77 88

Max1 22 33 44 11 55 66 77 88

Max1 22 33 11 44 55 66 77 88

Max1 22 11 33 44 55 66 77 88

11 22 33 44 55 66 77 88

using Knock-out tournament

$x_1 = 77$, $x_2 = 33$ → 77
$x_3 = 44$, $x_4 = 11$ → 44
77, 44 → 77
$x_5 = 88$, $x_6 = 22$ → 88
$x_7 = 66$, $x_8 = 55$ → 66
88, 66 → 88
77, 88 → 88

int min assign

Algorithm

from main()
f=1, l=n

QuickSort (f, l)
if f < l then
i ← f+1
while $x_i < x_f$ do
i ← i+1
j ← l
while $x_j > x_f$ do
j ← j−1
while i < j do
$x_i \leftrightarrow x_j$
repeat i ← i+1 until $x_i \geq x_f$
repeat j ← j−1 until $x_j \leq x_f$
$x_j \leftrightarrow x_f$
QUICKSORT (f, j−1)
QUICKSORT (j+1, l)

V.V.9

4. Quick Sort

keyword

f → 27, i → 99, 0, 8, 13, 64, 86, 16, 7, 10, 88 ← j (l)

Large →   ← small

Algorithm

unless (i > j)

(swap)

27 25 0 8 13 64 86 16 7 10 88 99 90

27 25 0 8 13 10 64 86 16 7 64 88 99 90

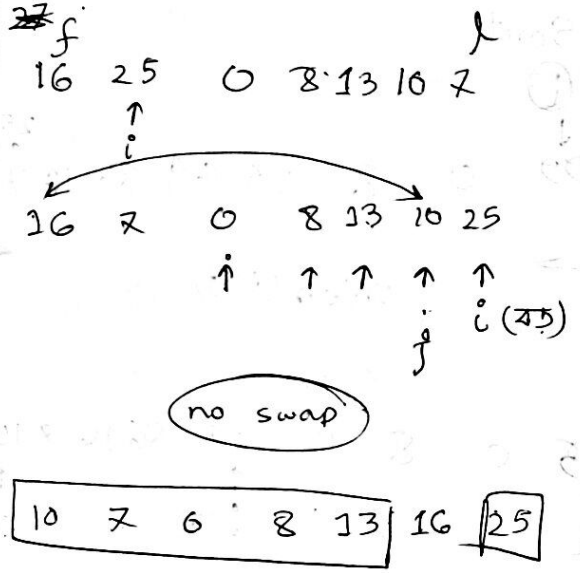
(बड़ from 27) (छोटा from 27)

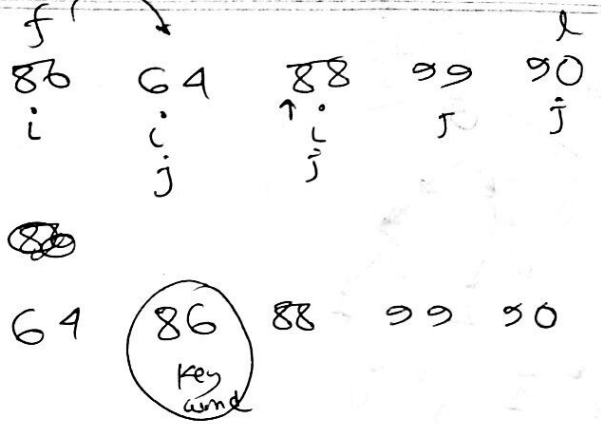
27 25 0 8 13 10 7 16 86 64 88 99 90

swap when i < j

16 25 0 8 13 10 7 | 27 | 86 64 88 99 90

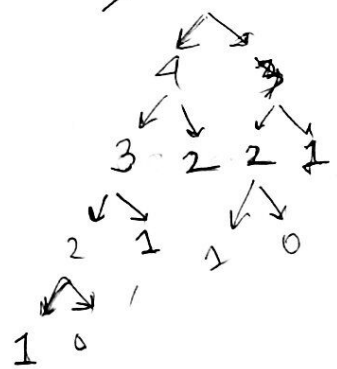
keyword   keyword was   keyword


f
l
16 25 0 8 13 10 7
i
16 7 0 8 13 10 25
i
j
i (~~25~~)
no swap
10 7 6 8 13 16 25


f
l
86 64 88 99 90
i
i
j
i
j
j
64 86 88 99 90
key word

রবিবার – 15-05-16 (Lab)

0	1	1	2	3	5	8	13
position → 0	1	2	3	4	5	6	7



Assignment

MergeSort

```
while (i<j)
{
    swap (arr[i], arr[j])

    i++;
    while (arr[i] >= arr[f])
    {
        i = i+1;
    }

    j--;
    while (arr[j] <= arr[f])
    {
        j = j-1;
    }
}
swap(arr[i], arr[f])
Quick sort (f, j-1)
"      "  (j+1, l).
```

রবিবার → 15-05-16

Algo

Merge-Sort (A, P, r)

if p<r then

$q \leftarrow \text{floor}\left(\frac{p+r}{2}\right)$ ⟹ divide

Merge_sort (A, P, q)

Merge_sort (A, q+1, r)

→ Merge (A, P, q, r)

p q r

Merge (A, P, q, r)

→ $n_1 \leftarrow q-p+1$ 8-4=4 4-1=3 (count length)

→ $n_2 \leftarrow r-q$

create arrays $L[1 \ldots n_1+1]$ and $R[1 \ldots n_2+1]$ (infinity value)

for $i \leftarrow 1$ to n_1 do

$L_i \leftarrow A_{(P+i-1)}$ $\frac{1+1-1}{1}$ / $\frac{1+2-1}{2}$ / $\frac{1+3-1}{3}$ -- upto n_1.

for $j \leftarrow 1$ to n_2 do

$R_j \leftarrow A_{(q+j)}$

$L_{n_1+1} \leftarrow \infty$

$R_{n_2+1} \leftarrow \infty$

$i \leftarrow 1$

$j \leftarrow 1$

for $k \leftarrow p$ to r do

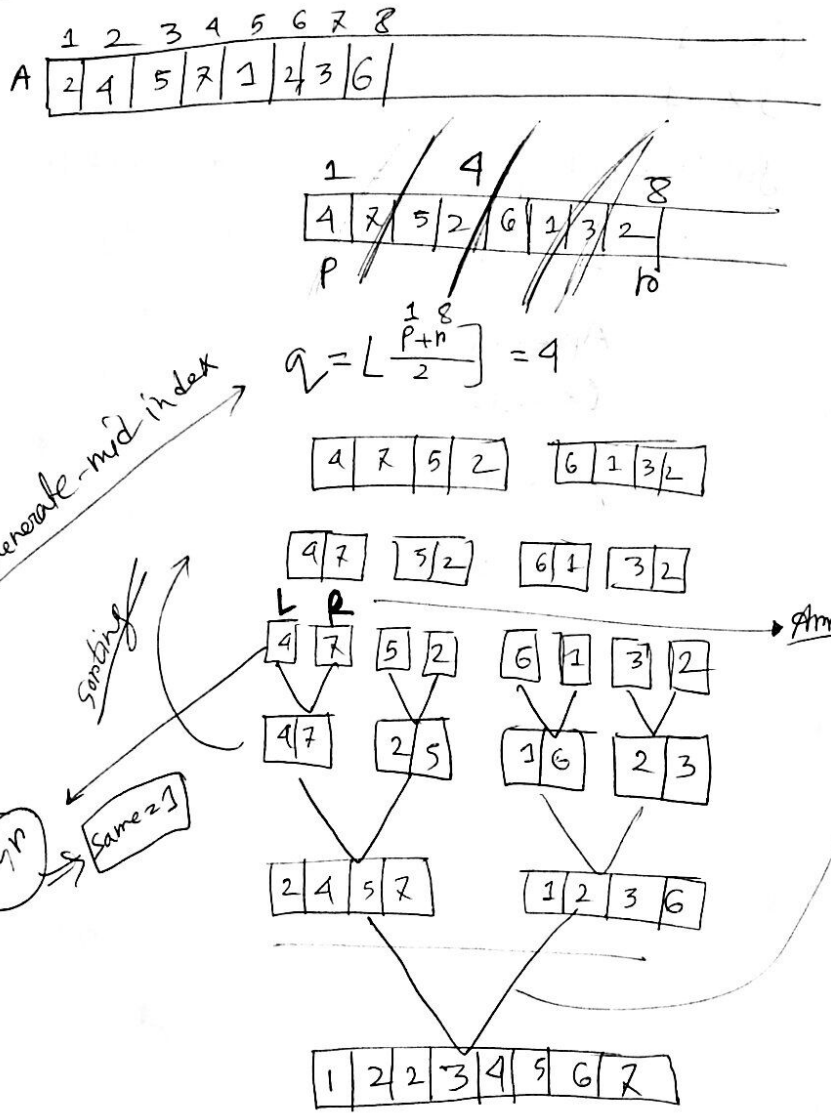
if $L_i \leq R_j$ then

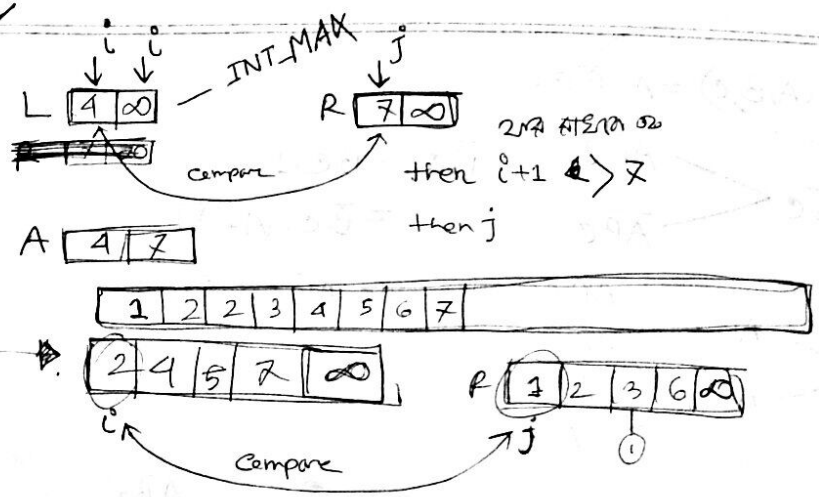
$A_k \leftarrow L_i$

$i \leftarrow i+1$

else $A_k \leftarrow R_j$

$j \leftarrow j+1$


A
Generate-mid index
Sorting
Array
Same?
p, q, r


INT_MAX
Compare
then i+1
then j
A
Compare

সোমবার ⇒ ২৬.০৫.২৬

$x \cdot x = x$

(*) $F(A,B,C) = A + \bar{B}C$

$\bar{B}C \rightarrow \bar{A}\bar{B}C,\ A\bar{B}C$

$\bar{B}C = \bar{B}C \cdot 1$

$= \bar{B}C \cdot (A + \bar{A})$

$= A\bar{B}C + \bar{A}\bar{B}C$

$A \rightarrow AB \rightarrow ABC, AB\bar{C}$

$A \rightarrow A\bar{B} \rightarrow A\bar{B}C, A\bar{B}\bar{C}$

original form

$F = ABC + AB\bar{C} + A\bar{B}C + A\bar{B}\bar{C} + \bar{A}\bar{B}C + \bar{A}\bar{B}C$

$= ABC + A\bar{B}C + A\bar{B}C + A\bar{B}\bar{C} + \bar{A}\bar{B}C + \bar{A}\bar{B}C$

$= \Sigma(7, 6, 5, 4, 1)$

A	B	C	F
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

(*) $F(x,y,z) = xy + \bar{x}z$

$= (xy + \bar{x})(xy + z)$

$= (x + \bar{x})(\bar{x} + y)(x + z)(y + z)$

$= (\bar{x} + y)(x + z)(y + z) \Rightarrow$ P.O.S

missing variable

$\bar{x} + y = \bar{x} + y + z\bar{z} = (\bar{x} + y + z)(\bar{x} + y + \bar{z})$

$x + z = x + z + y\bar{y} = (x + y + z)(x + \bar{y} + z)$

$y + z = y + z + x\bar{x} = (x + y + z)(\bar{x} + y + z)$

$\bar{x} + y = 0$

Variable ordering binary expression Represent

Karnaugh Map

$F = (\bar{x} + y + z)(\bar{x} + y + \bar{z})(x + y + z)(x + \bar{y} + z)(x + y + z)$

$(\bar{x} + y + z)$

$= (\bar{x} + y + z)(\bar{x} + y + \bar{z})(x + y + z)(x + \bar{y} + z) \Rightarrow$ Max term

$= \Pi(4, 5, 0)$

Make T.T

A	B	C	F
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

(*) Map Method

K-Map

2-Variable Map

x \ y	0	1
0	00	01
1	10	11

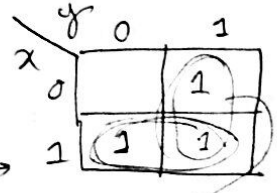
$\bar{x}\bar{y} - 0$

$\bar{x}y - 1$ [01]

$x\bar{y} - 2$

$xy - 3$

$F(x,y) = x\bar{y} + \bar{x}y + xy$



x এর value → 1

$= x$ (1,1) y (1,1)

2

not possible

0

0, 1 or 1, 0

① সবচেয়ে বড় circle বানানো যায়

② 2^n

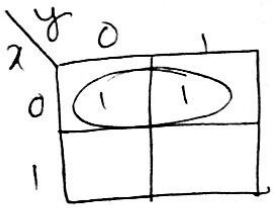
$n=0,\ 2^0 = 1$

$n=1,\ 2^1 = 2$

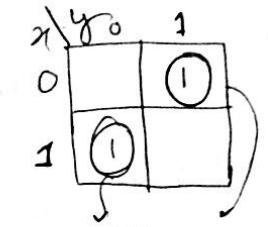
$n=2,\ 2^2 = 4$

$n=3,\ 2^3 = 8$

③ কোনাকুনি circle Not possible



$\bar{x}$



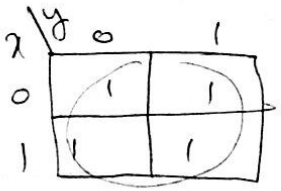
$= x\bar{y} + \bar{x}y$

Exclusive OR

x	y	$F = x \oplus y$
0	0	0
0	1	1
1	0	1
1	1	0

Expression sum of minterm

$F = x\bar{y} + x + \bar{y}$

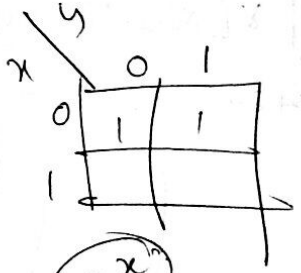


$\bar{x}\bar{y} + \bar{x}y + x\bar{y} + xy$

$= \bar{x}(\bar{y}+y) + x(\bar{y}+y)$

$= \bar{x} + x$

$= 1$



$= \bar{x}$

$\bar{x}\bar{y} + \bar{x}y$

$= \bar{x}(\bar{y}+y)$

$= \bar{x}$

১৫/05/16 ⇒ রবিবার

only positive — not negative

Radix sort

use the fields Link1, link2, --, Linkn. to form $x_1, x_2, --, x_n$ into an input Queue, Q.

for $j = 1$ to p do → no. of passes = no of digits

initialize each of the r queues $Q_0 - Q_{r-1}$ to be empty

while Q not empty do

$x \Leftarrow Q$ ⇒ (pop)

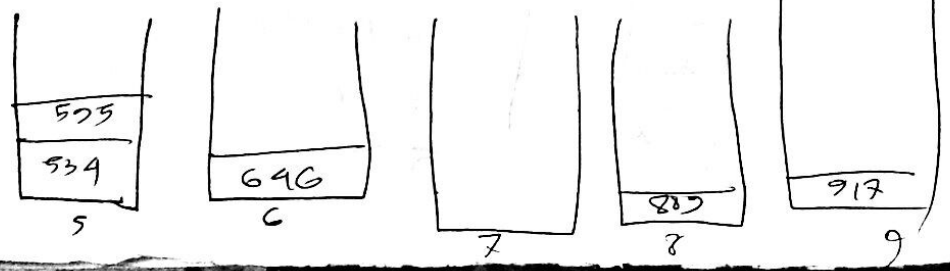
let $x = (x[p], x[p-1], --, x[1])$

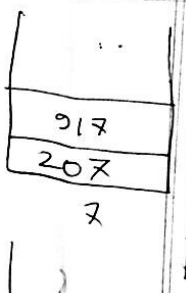
$Q_{x[j]} \Leftarrow x$. | $Q[x[j]] = [Q_0 \text{ —— } Q_9]$; $Q[x[j]]$ [

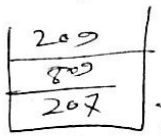
Concatenate queues $Q_0 - Q_{r-1}$ together

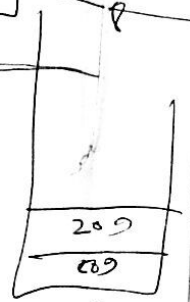
to form the new Queue, Q. | $x[x[1], x[2], x[3]]$.

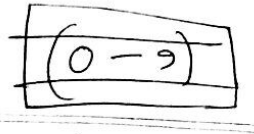
207 809 209 310 917 534 646 376 181 095 595 198





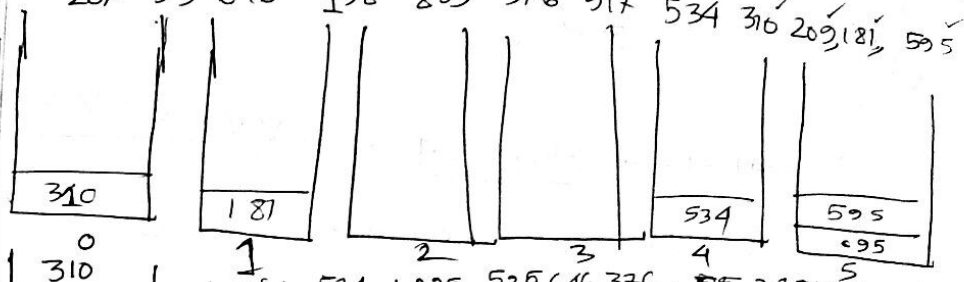


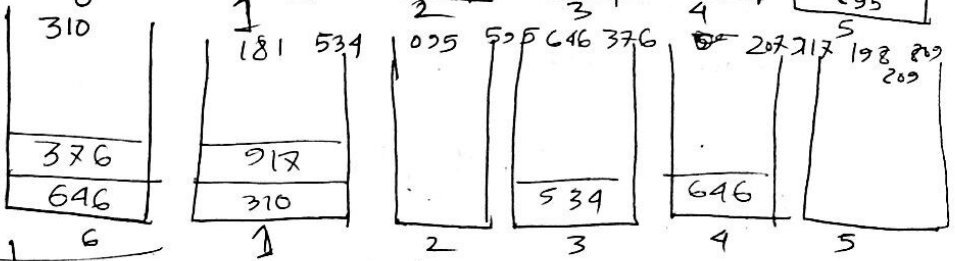


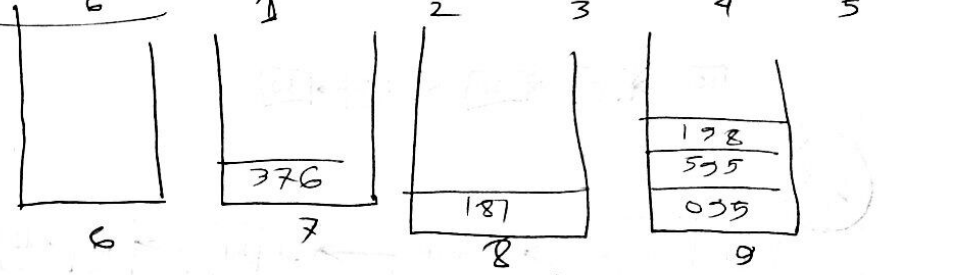


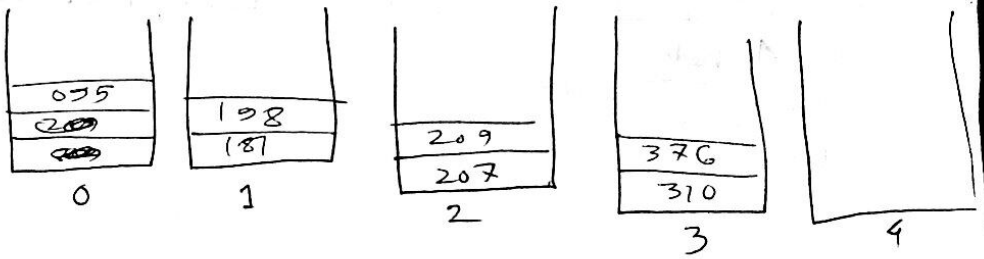
$x \in Q$

207 95 646 198 809 376 917 534 310 209 181 595







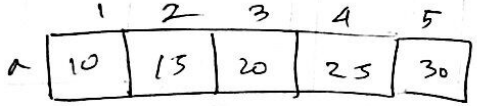


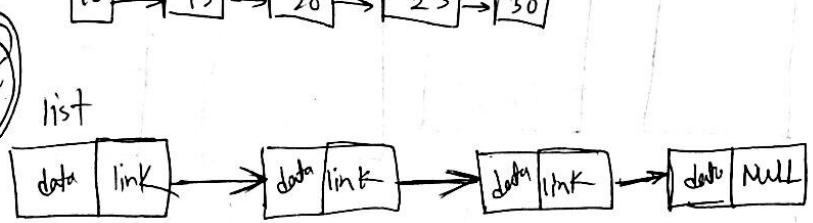
18.05.16 ⇒ 2주차

structure pointer

(3-4) set → chapter-4 ⇒ V.V.V.V.V.V.V.V.V.

A data structure list is a finite sequence of elements.





To go any index have to go through from head

struct node


```
{
    int data;
    struct node * link;
} *P1, *P2, *P3, *P4, *head, *P;
```

size of (struct node)

type casting

P1 = (struct node*) malloc (sizeof(structure node))

P2 = 4
P3 = 4
P4 = 4

Head = P1;
P1 → info = 10;
P1 → link = P2;

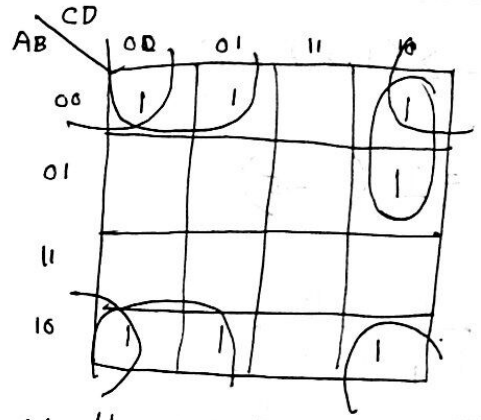
P2 → info = 20;
P2 → link = P3;

P3 → info = 30
P3 → link = P4;

P4 → info = 40;
P4 → link = NULL;

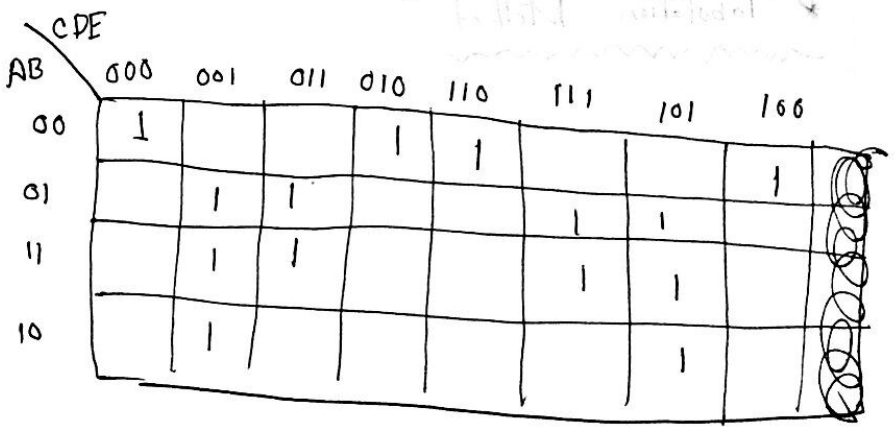
```
P = head;
while (P != NULL)
{
    printf("%d ", P->info);
    P = P->link;
}
```

* $F(A,B,C,D) = \bar{A}\bar{B}\bar{C} + \bar{B}C\bar{D} + \bar{A}BC\bar{D} + A\bar{B}\bar{C}$



* 5 – Variable Map

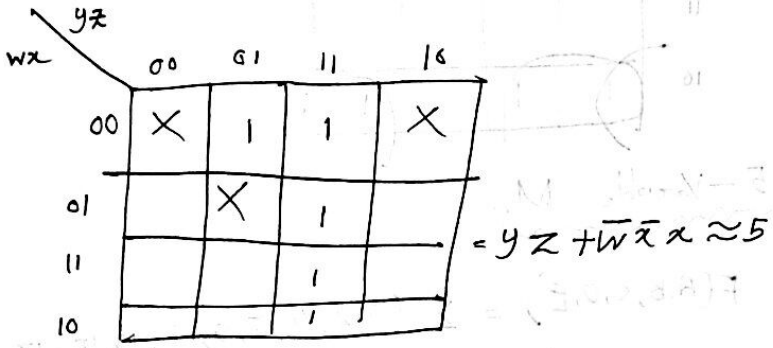
$F(A,B,C,D,E) = \Sigma (0,2,4,5,9,11,13,15,17,21,25,27,29,31)$



* Dont Care Condition

$F(w,x,y,z) = \Sigma(1,3,7,11,15)$

$d(w,x,y,z) = \Sigma(0,2,5)$

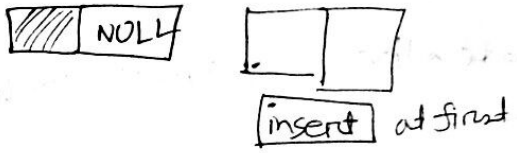


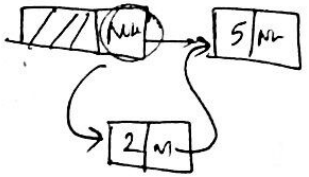
$= yz + \bar{w}\bar{x}z \simeq 5$

* Tabulation Method

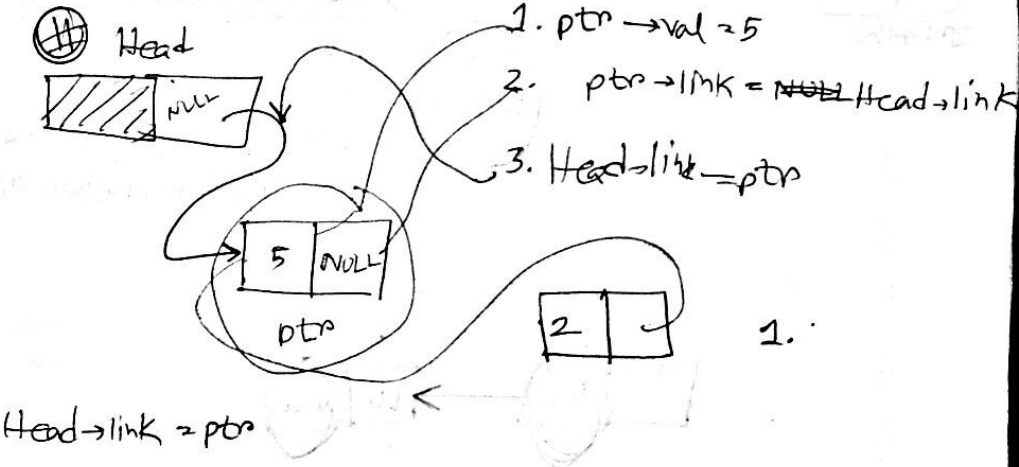
22 Mar

Lab-3





Head → link



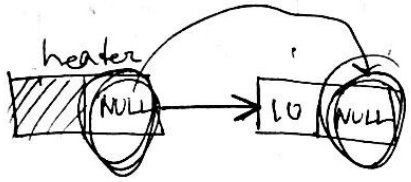
Head → link = ptr

self referential pointer → struct node → link → node এর link

struct node (a header;) ⇒ pointer

header = (node *) malloc (sizeof(node));

- header: টেম্পরারি ভ্যারিয়েবল
- (node *): type casting
- malloc: memory allocation, void type এর pointer return করে
- sizeof(node): size

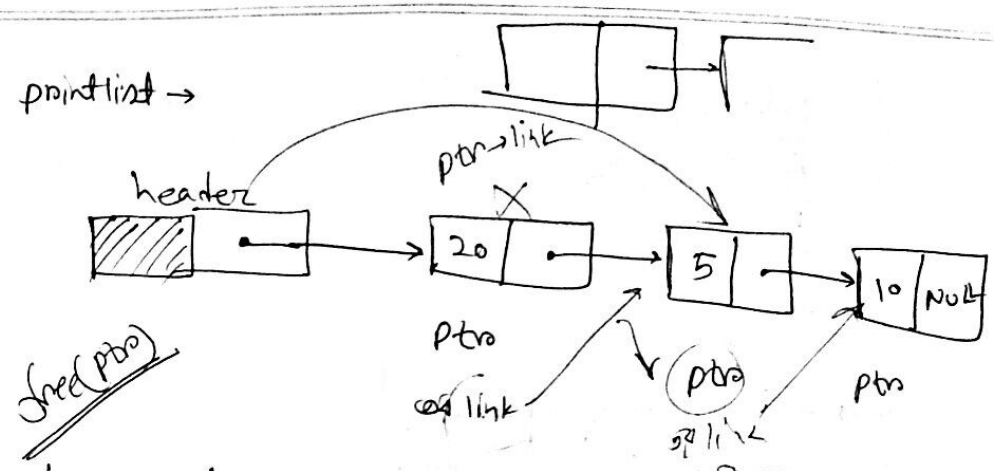


NewNode → data = data

NewNode → link = header → link

header → link → link = newnode;

update



1.
2. ptr → link = pt → link
3.

value index

100

1. wy
2.
3. ptr → link = ptr → link

the size of LinkList

Enter:
Random array
printLinkList
~~Enter option:~~

choose option:

1. Insert → First / Last / At index / after Value → which value you want to update
2. update
3. Delete
4. search → Enkter seach key: → print index
5. Exit

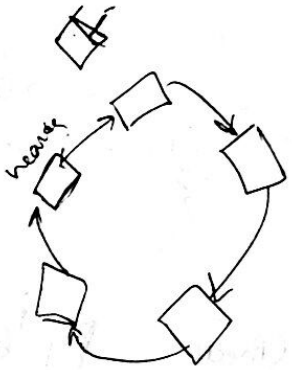
if 6

always display

circular linkList

Function এর মধ্যে function

body main এর ভিতে-



22 May

Algo's insertion of a new record in a sorted linked list

if L = nil or number(new) < number(L) then

Link(new) ← L

L ← new

35 < 45

else

pred ← L

P ← Link(L)

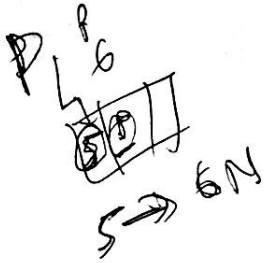
while P ≠ nil and number(P) <= number(new) do

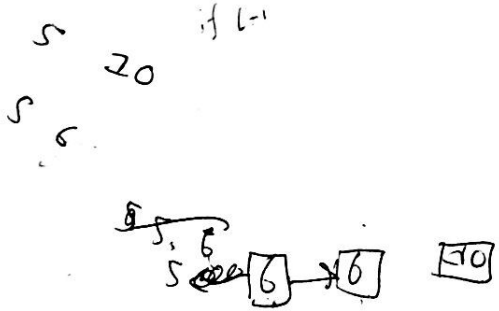
pred ← P

P ← link(P)

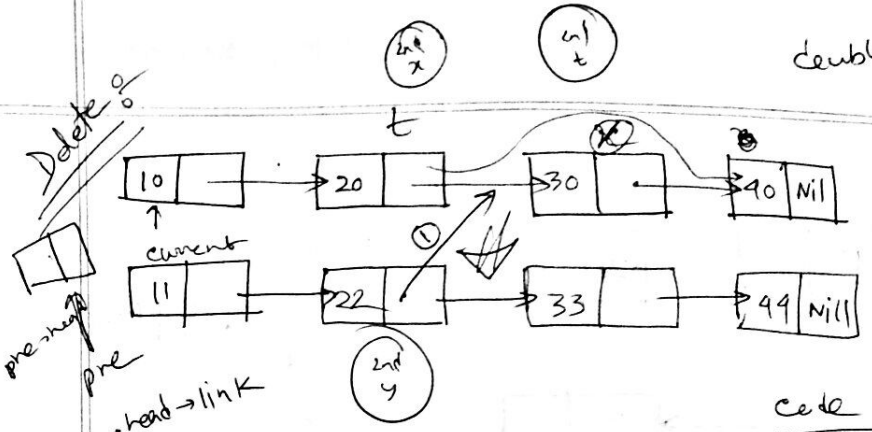
link(new) ← link(Pred)

link(pred) ← new





Delete :-



if(curr→info == 30)

{ pre→link = curr→link;

}

pre ← curr

curr ← curr→link

code vs Algo

t → link → link(t)

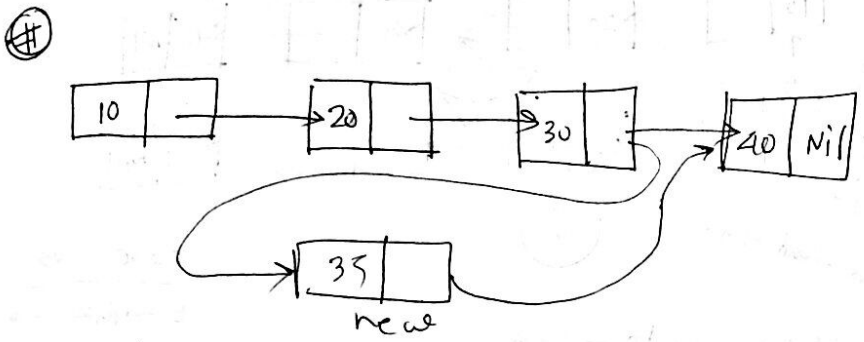
link(t) ← link(x)

① link(y) ← link(x)

Delete → link(s) ← link(x)

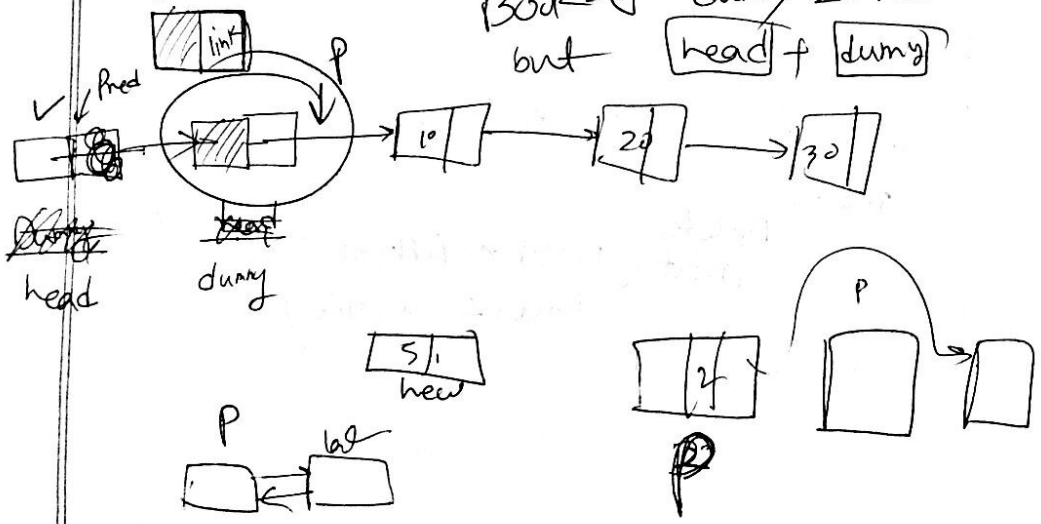
move

Delete insert (link(t) ← link(y)
link(t) ← link(x)/t)



Algo

Dummy elements — Fn? ② Advantage?



Bad ← ① dummy = head

but head + dummy

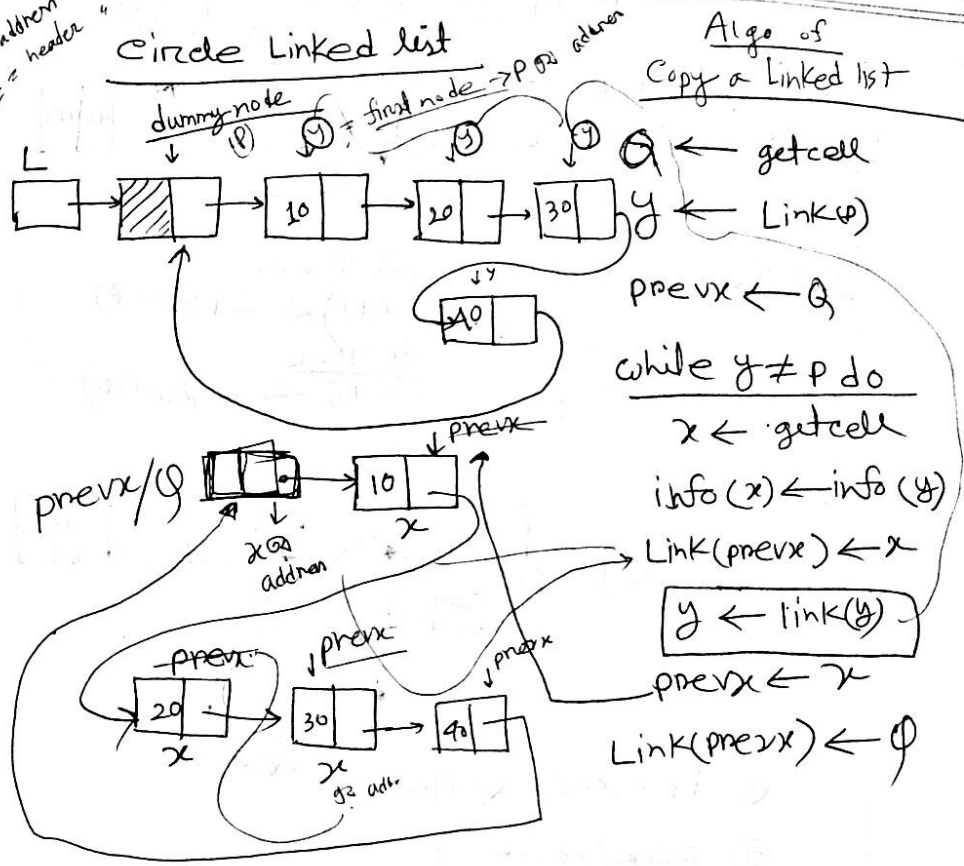
তারিখ: 24.05.18

Circle Linked list

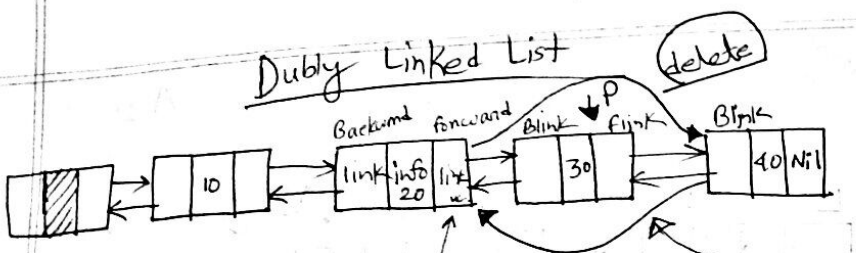
Last address = header

dummy node (P) (y) = first node → P এর address

Algo of Copy a Linked list



Dubly Linked List

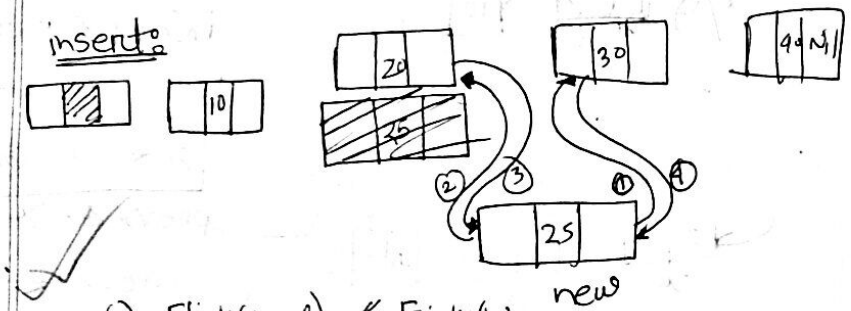


Delete:

if Blink(p) ≠ nil then
Flink(Blink(p)) ← Flink(p)

if Flink(p) ≠ nil then
Blink(Flink(p)) ← Blink(p)

insert:



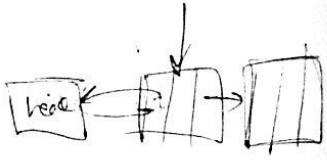
① Flink(new) ← Flink(y)

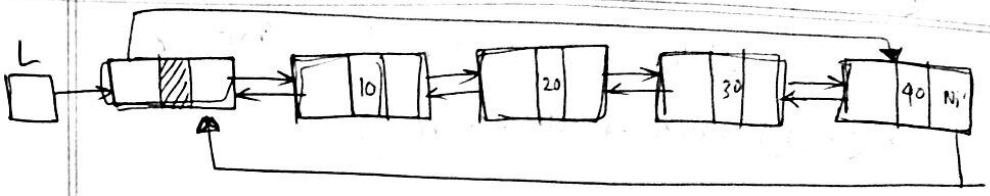
② Blink(new) ← y

③ Flink(y) ← new

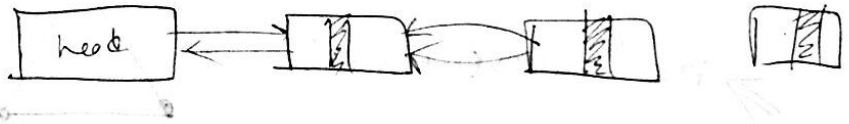
④ Blink(Flink(y)) ← new

if dobly circular Linked list





Ad



2

(pc)

Set

$^nc_1$ combination + / $^nc_3$ / $^nc_5$

$^nc_2$ " –

$n(A \cup B) = n(A) + n(B) - n(A \cap B)$

↳ $^nc_1 + ^nc_2 + ^nc_3$

$\sum_{i=1}^{n} {}^nc_i = 2^n - 1$

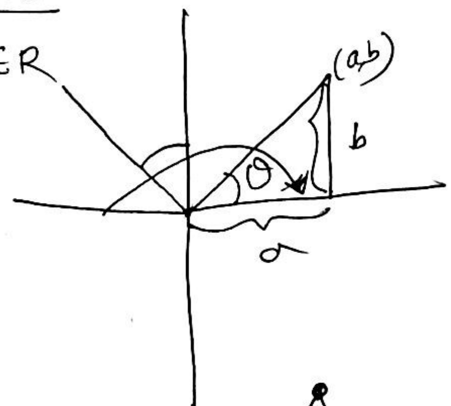
Complex Number

$a + ib$; $a, b \in R$

$i = \sqrt{-1}$

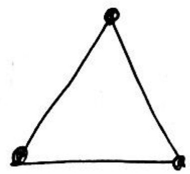
Regular n-gon

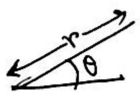
90° rotation $\times i$

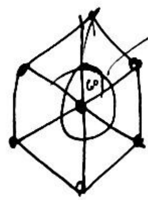


z

$\Rightarrow re^{i\theta} = r\cos\theta + ir\sin\theta$







$360/n = 60$

$(r, \theta) \to (1, 0)$

Polynomial: 1 x^2 $x^{2/3}$ x

↳ Degree

$x^2 - (a+b)x + ab = 0$

Fa~~ctorial~~:

$f(x)$

$f(x) = (x-a)P(x)$

Factor/Remainder theorem

$f(x)$

$f(a)$

$(x-a)$

Matrix determinate:

Associative

$A(BC)$

$= (AB)C$

$O(n^3) \Rightarrow$ matrix multi.

System of linear equation

^nP_r, $^nP_{r+1}$

permutation combination

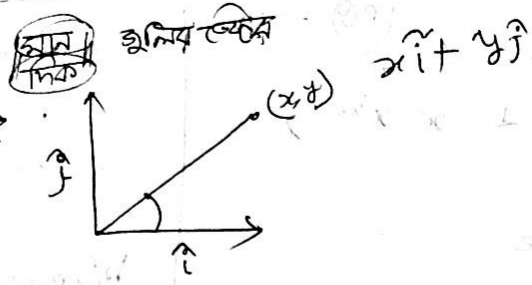
Sumation:

VVg $\sum n^4$

straight line circle points

2004 ⇒ Geometry (9-10)

ভেক্টর:



স্থান ভিত্তিক

অবস্থান ভিত্তিক

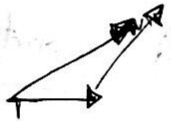
$x\hat{i} + y\hat{j}$

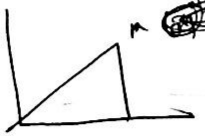
$\frac{\vec{A}}{|\vec{A}|} \rightarrow \hat{A}$

$\overrightarrow{AB} \rightarrow B - A$

P_1, P_2

(A) ভেক্টর এর sum:





$\Rightarrow x\hat{i} + y\hat{j}$

(B) projection:

$|\vec{A}| |\vec{B}| \rightarrow |\vec{A}| |\vec{B}| \cos\theta$

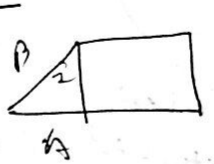
Ahunt

$\vec{A}\,\vec{B}$

$\vec{A} \times \vec{B}$

$= |\vec{A}| |\vec{B}| \sin\theta \cdot \hat{A}$

(মাত্রায়) (অবস্থা)



$\Rightarrow$ 'x' products

$$\begin{vmatrix} \hat{i} & \hat{j} & \hat{k} \\ A_x & A_y & A_z \\ B_x & B_y & B_z \end{vmatrix}$$

User input

empty() - List

Linked list

Stack
Insert Last ⇒
Delete Last ⇒

Radix sort

Enter n:
if n=5
n1
n2
n3
n4
n5

Queue
Insert first → Enqueue
Delete first → Dequeue

Terminal checking

রবিবার ⇒ 29.05.16

H.W: Sequential implementation
Stack & Queue

Linked implementation:

Stack.

getcell = memory allocation

S ∈ x l ← getcell

info(l) ← x

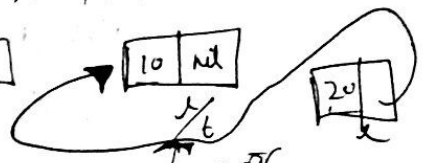
link(l) ← t

t ← l

x ∈ s ; if t = nil then underflow

else x ← info(t)

t ← link(t)

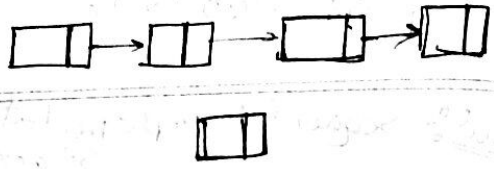


underflow → empty

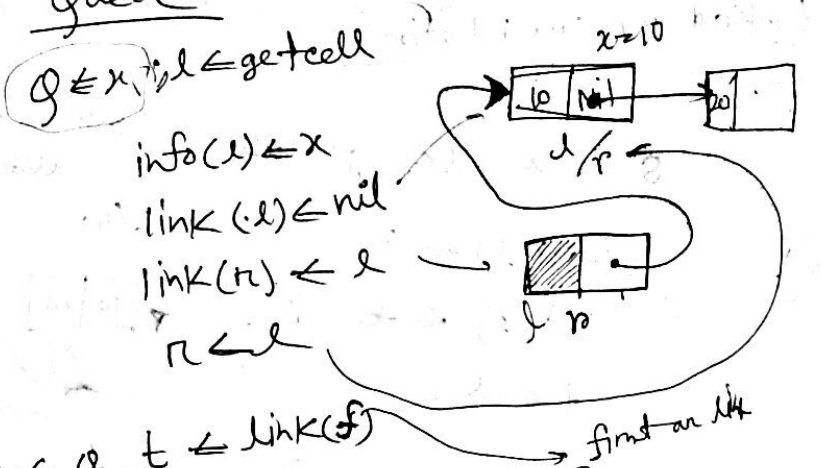
void pointer

NULL pointer

pointer



Queue



x ← q t ← link(f) → first or link

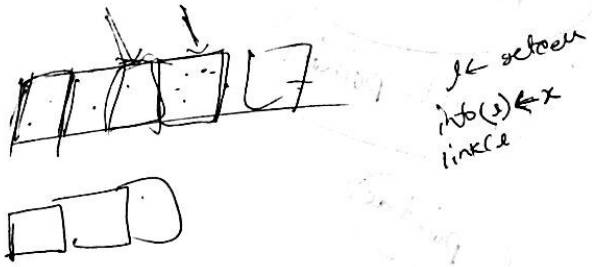
if t = nil then underflow

else x ← info(t)

link(f) ← link(t)

if link(t) = nil then r ← f

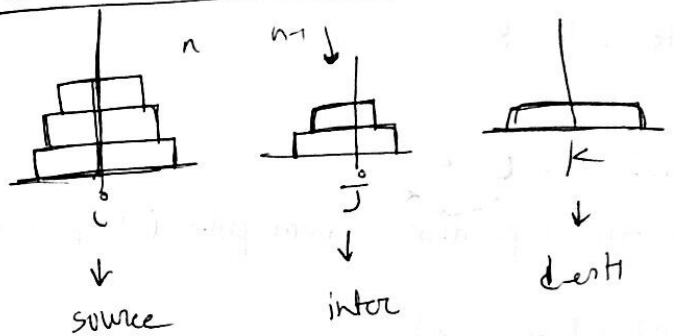
free(t)



Application of stack and Queues

Stacks & Recursion :-

Towers of Hannoi



i → k
i → j
k → j
i → k
j → i
j → k
i → k

HANOI (n, i, j, k)

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

else

HANOI (n-1, i, k, j)

move the top disk from pole i to pole k

HANOI (n-1, j, i, k)

$H_n = 2H_{n-1} + 1$

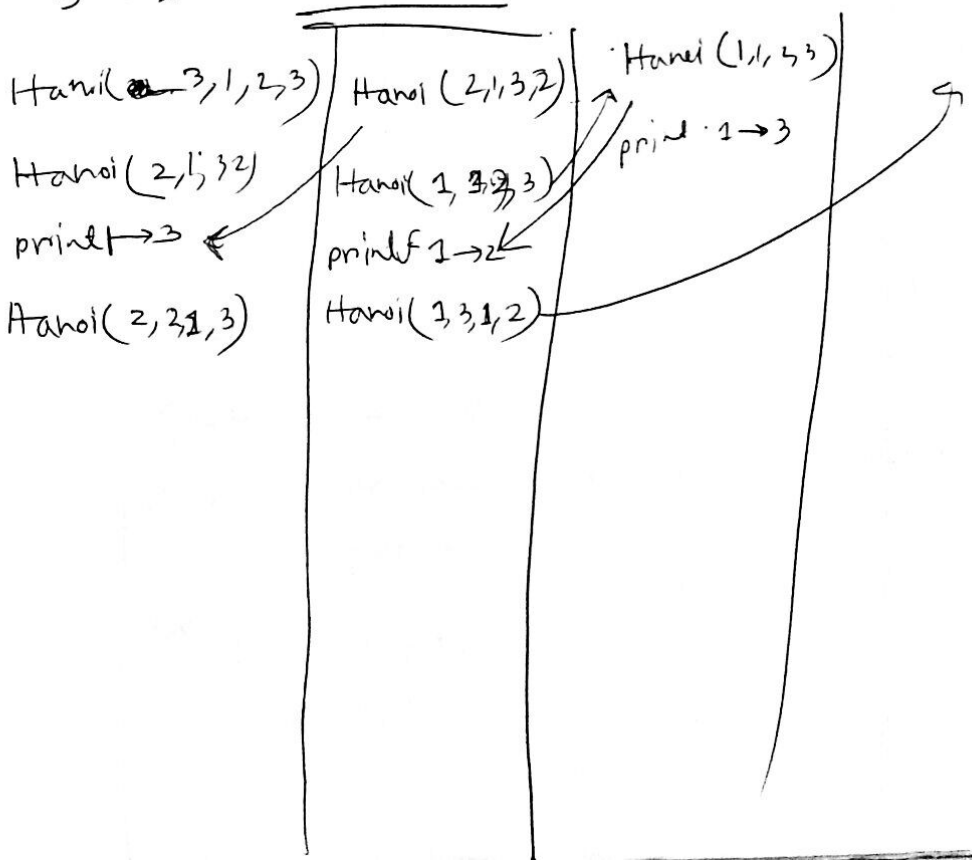
$H_1 = 2H_0 + 1$

$= 2 \cdot 0 + 1 = 1$

$H_2 = 2H_1 + 1$

$= 2 + 1 = 3$

$H_3 = 2H_2 + 1 = 7$



if
i if

Hanoi(3,3,1,2)	Hanoi(3,1,2,3)	Hanoie(1,3,3,1)
printf; 3-2	~~Hanoi(3,3,1)~~	print. 2→1
	Hanoi(1,2,3,3)	
	print 2→3	
	Hanoi(1,1,3,3)	

1→3
1→2
3→2

1→3
2→1

1→3
1→2
3→2
1→3
2→1
2→3
1→3
if n=1

Hanoi(3,1,3,3)
print 1→3

1	2	3	1	×8
1	2	1	3	×9
1	1	2	3	×10
1	1	2	3	×3
1	1	2	3	×4
1	3	1	2	×5
2	~~1~~	3	2	×2
1	1	2	3	×6
2	2	1	3	×7
3	~~1~~	~~2~~	3	×1

S

n	i	j	k
3	1	2	3
2	1	3	2
1	1	2	3
1	1	3	2
1	3	1	2
1	1	2	3
2	2	1	3

Stack algorithm for the towers of Hanoi problem.

1	2	3	1
1	1	2	3

S ← empty stack

S ⇐ (n, i, j, 3) (insert)

while S ≠ empty do

(n, i, j, k) ⇐ S (pop)

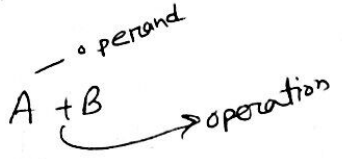
if n = 1 then move the top disk from pole i to pole k.

else S ⇐ (n-1, j, i, k)

S ⇐ (1, i, j, k)

S ⇐ (n-1, i, k, j)

Stacks & Arithmetic Expression



+AB → prefix form

AB+ → postfix "

character	Precedence
#	0
)	1
(	2
+, −	3
*, /	4

H.W → infix → prefix

Algorithm:

Conversion on infix expression to postfix Expression

S ← Empty stack

S ⇐ "#"

j ← 0

i ← 0

while top(S) ≠ "$" do

i ← i+1

case

① E[i] is an operand

j ← j+1

P[j] ← E[i]

② E[i] = "(" : S ⇐ E[i]

③ E[i] = ")"

x ⇐ S

while x ≠ "(" do

j ← j+1

P[j] ← x

x ⇐ S

(A) E[i] is an operator

while Prec(E[i]) ≤ Prec(top(s)) do

j ← j+2

P[j] ← S

S ← E[i]

PostE

(A+B) * ((C-D) ↑ E+F)

A+B ÷ AB+

((C-D) ↑ E+F) = (CD- ↑ E+F

= (CD-E↑) + F

= (CD-E↑)F+

(AB+) * (CD-E↑F+)

= AB+CD-E↑F+*

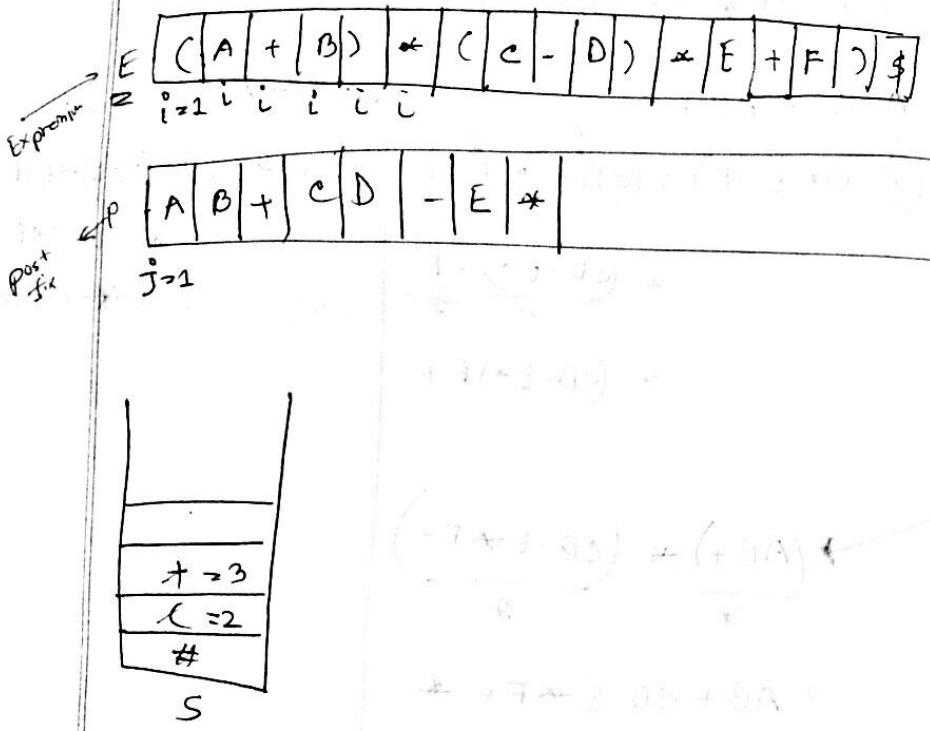
Pre

A+B = +AB

((C-D) ↑ E+F) = (-CD) ↑ E+F

= (↑-CDE)+F

= +(↑-CDE)F



| + 2 + 3 + 4 (7)

0 1 2 3 4 5 6

তারিখ ⇒ ০২.০৫.২৬

Algorithm

Evaluation of Postfix Expression

S ← empty stack

for i = 1 to n do

if P[i] is an operand then

S ⇐ P[i]

else (pop)

y ⇐ S

x ⇐ S

S ⇐ value of the operator P[i] applied to x and y

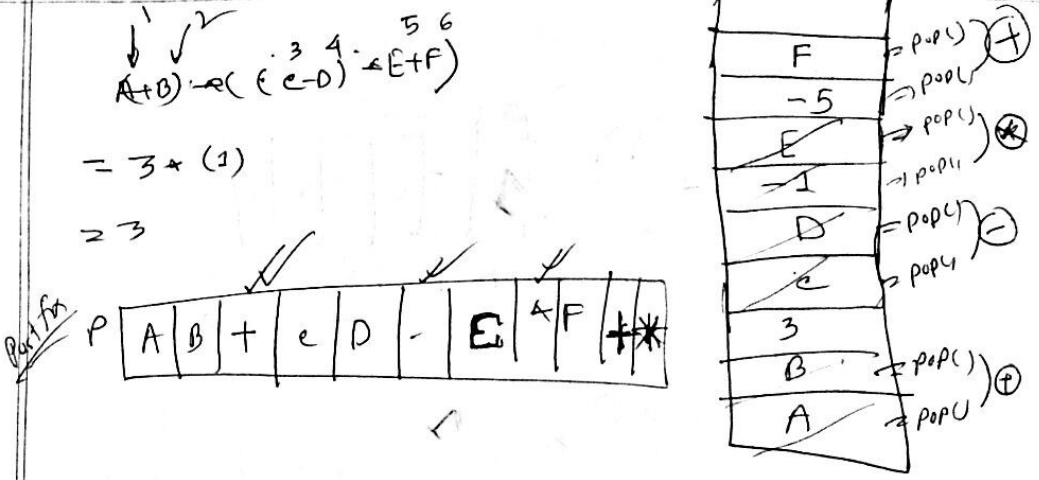
Priority Queue

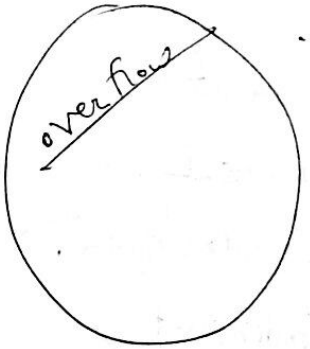
- It is an abstract data type which is like a regular queue or stack data structure, but where additionally each element has a priority associated with it.

- An element with high priority is served before an element with low priority.

- If two elements have the same priority, they are served according to their order in the queue

- Application : An array, Linked list, Binary search tree





- y ≤ s
- x ≤ s
- x = y
- x > y

PQ

1	2	3	4
10	20	30	40
1	4	2	2

n

R = R % size
= 8 % 8
= 0

$n(n-1)$

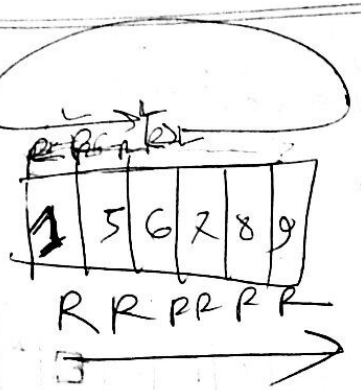
3

$2+1$

$4+2+2+1$

overflow

$|R-L| = 1$



$R = R \% n$

$8 \% 8 = 0$

Assignment

(i) DoubleLL.

(ii) circular queue

(iii) Radix sort

Deadline :- Friday 11:59 pm

08.06.16 => বুধবার

Abstract data type: Deque (Double Ended Queue)

- Queue() creates a new queue i.e empty. It needs no parameter and returns an empty queue

- enqueue(item) adds a new item to the rear of the queue it needs the item & return nothing

- dequeue() removes the front item it needs no parameters & return the item.

- IsEmpty() tests to see whether the queue is empty it needs no parameter & returns a boolean value.

- size() returns the number of items in the queue. It needs no parameter & returns an integer.

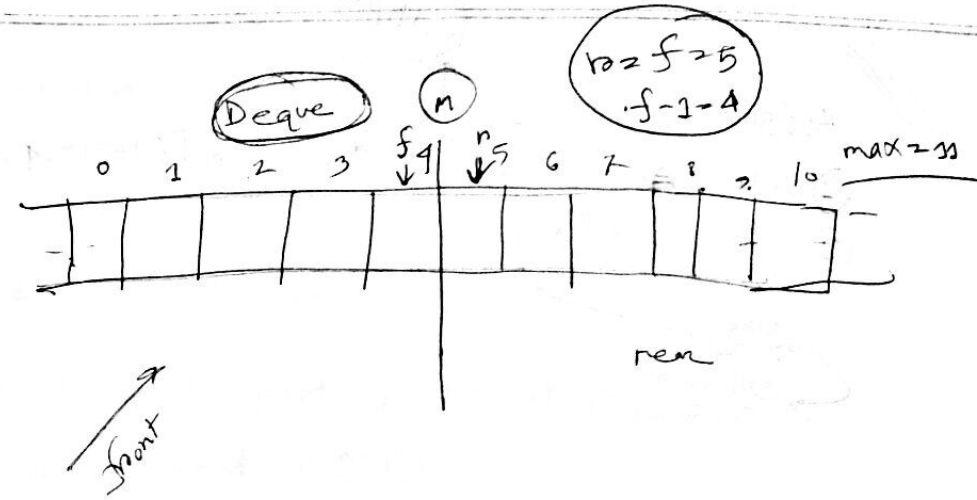
Application:-

- Bank (a queue of waiting patrons)
- printer.

Four operations of Deque

Algorithm insert front

1. start
2. check the queue is full or not as if (r = max-1)
3. if false update the pointer f as f = f-1.
4. Insert the element at pointer f as q[f] = element
5. stop



Algorithm Remove front:

1. start
2. check the queue is empty or not as if (f=r)
3. if false update the pointer f as f = f+1 and delete element at position f as element = q[f].
4. if (f=r) reset pointer f and r as f=r=0 -1
5. stop

After-Eid

bonus

multiple diget → Online

Assignment:-

① online ⇒ $[\{(5+4)*(3-2)\}]*[(4+3)*(4-2)]$

② infix to postfix

Infix

আসল

calculation

রবিবার ⇒ 13.06.16

4. Algorithm Remove-Back of DEQUE

দিক পিছনের remove করা

1. Start
2. check the queue is empty or not
   as if (f = = r) . If yes queue is empty
3. If false update pointer at position r as element = Q[r]
4. Update pointer r as r = r - 1
5. If (f = = r) reset pointer f and r as f = r = -1
6. Stop.

E[ ]

V[ ]

traverse → BFS

white ← not visible
gray ← Entry in Queue
black ← Visited

(Breadth first search)

BFS (V, E, S) — S: starting node → start node

for each u in V-{s} do

color [U] ← white

d[U] ← ∝

π[U] ← NIL (-1) (step)

Starting node [illegible] white color → node → white color & distance → ∝

parent node

colors [s] ← Gray

d[s] ← 0

π[s] ← NIL

Q ← empty queue → Q [illegible] S insert

ENQUEUE (Q, S) → push operation

while Q is non-empty do

u ← DEQUEUE (Q)

for each v adjacent to u do

if color[v] ← white then

color [v] ← Gray

d[v] ← d[u]+1 → distance plus

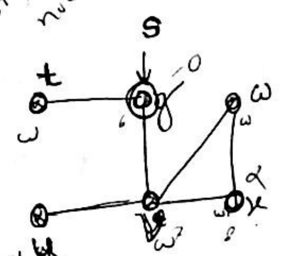
π [v] ← u

ENQUEUE (Q, v)

if (E[ ][ ]>0)

~~DEQUEUE(Q)~~

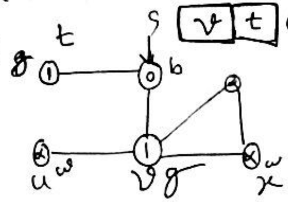
color [u] ← Black

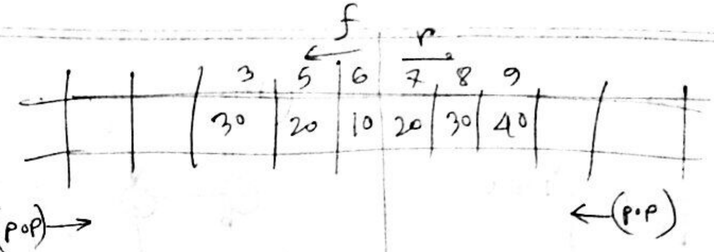


S Q

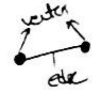
→ pop S — parent node pop

v t Q





color graph



Graph

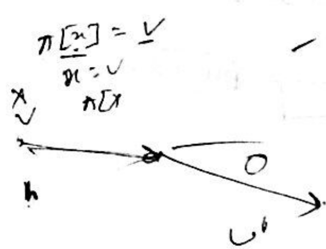
HW

→ definition of graphs, edges, vertices

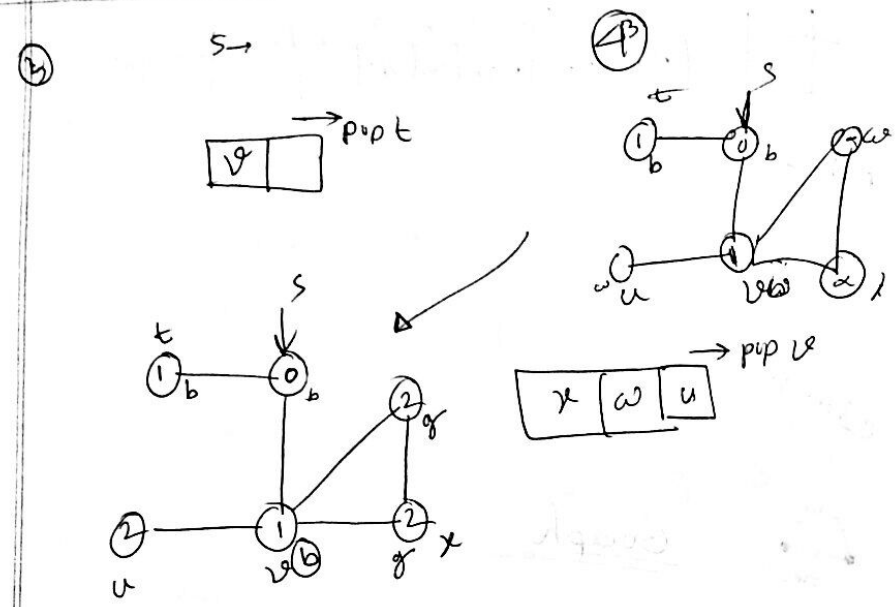
→ diff types of graphs

→ Adjacent Matrix.

π[x] = v



	0	1	2
0	0	1	0
1	1	0	0
2	0	1	0



15.0616 ⇒ বুধবার = next বুধবার Quiz ⇒ Arithmetic Expression

BFS (Print_~~First~~ Search) Path)

Print-Path (G, s, v) → starting node v → f → array → b

if v = s then print s → b – b

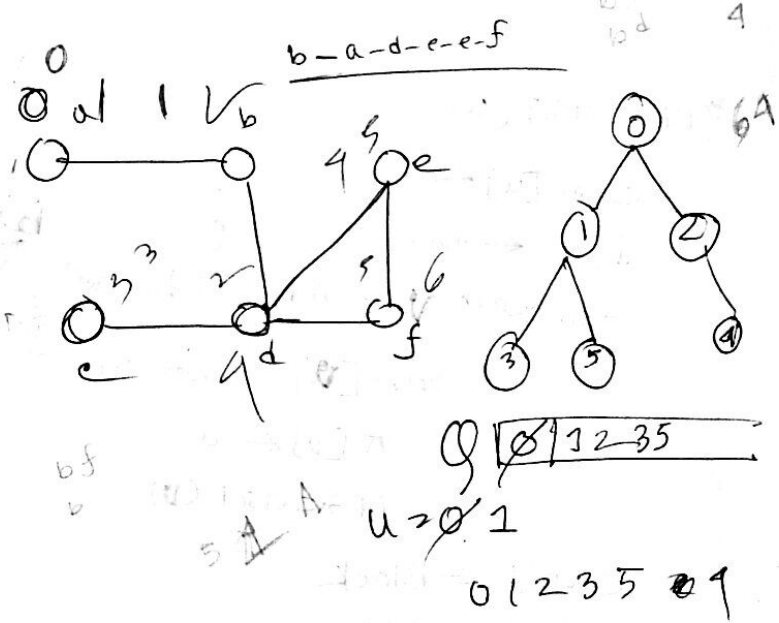
else if π[v] ← NIL then → Parent node NULL কিনা? যদি হয়, then Path নাই

Print 'no path exists' from 's' to 'v'

else Print-Path (G, s, π[v]) → Parent node দিয়ে আবার call

print v

b–f
b–d
b–b



DFS (Depth-First-Search)

DFS(G)

for each vertex $u \in V[G]$ do

color[u] ← white. → initially white

$\pi[u] \leftarrow NIL$ → parent node NULL

time ← 0

for each vertex $u \in V[G]$ do

if color[u] = white then

DFS_VISIT(u).

DFS_VISIT(u)

color[u] ← Gray

d[u] ← time +1

for each $v \in Adj[u]$ do

if color[v] = white then

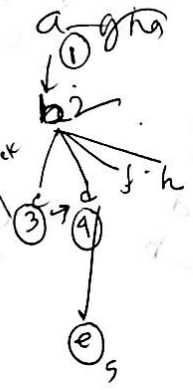
$\pi[v] \leftarrow u$

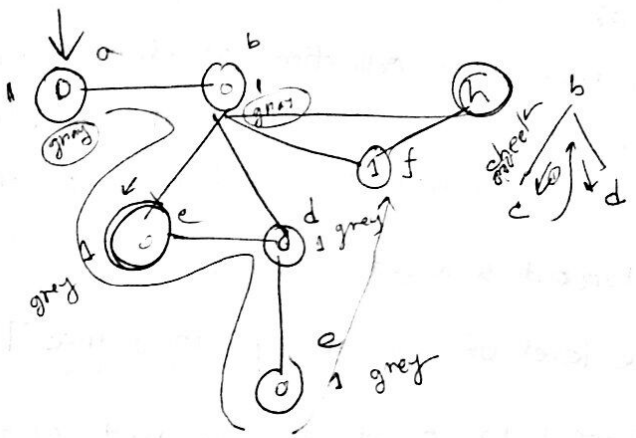
DFS_VISIT(v)

Color[u] ← Black

f[u] ← time+1

Color[u] ← Black





(19-06-16 ⇒ রবিবার)

Trees:-

- A tree is a collection of elements, of which one is the root and the rest are partitioned into trees called the subtrees of the root.

- A Forest is a set of trees.

- The level of a node p in a tree T is defined recursively as 0 if p is the root of the tree T, otherwise the level of p is,

$p = 1 + \text{level}(\text{Father}(p))$

- The height h(T) of a tree T is defined by

$$h(T) = \max_{\text{nodes } p \text{ in } T} \text{level}(p)$$

$$= \max \text{level}(0,1,1,1,2,2,3)$$

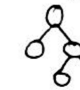
$$= 3$$

Binary Tree:-

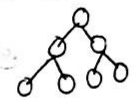
Types:-

- A rooted Binary tree is a tree with a root in which every node has at most two children.

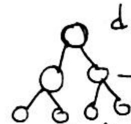
- A Full Binary tree is a tree in which every node other than the leaves has two children.



- A perfect Binary tree/ Complete Binary tree is a binary tree in which every level, except possibly the last, is completely filled and all nodes are as far left as possible

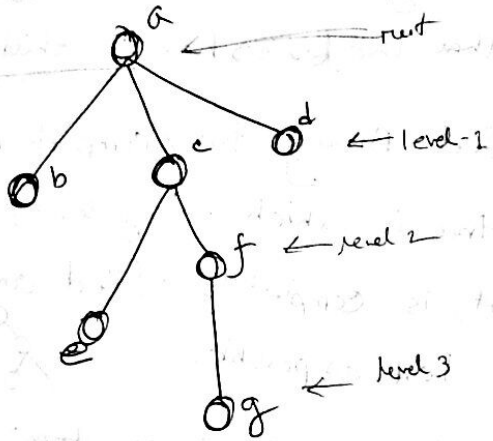


- An infinite complete binary tree is a tree ~~in which~~ with levels, where for each level d the ~~number~~ number of existing nodes at level d is equal to 2^d.



Advances

- Trees reflect structural Relationships in the data
- " are used to present hierarchies
- " provides an efficient insertion & searching
- " are very flexible data, allowing to move subtrees around with minimum effort



root-এর level = 0

leafs: b, d, e, g

g-এর Parent → f

f , child → g

sibling → (b) → c, d

✓ maximum level ≡ height

f-এর level = 1 + level(father(f))

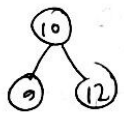
= 1 + level(c)

= 1 + 1 + level(father(c))

= 1 + 1 + level(a)

= 2 + 0 = 2

binary search tree



4 condition

① Maximum 2 vertex

② L R

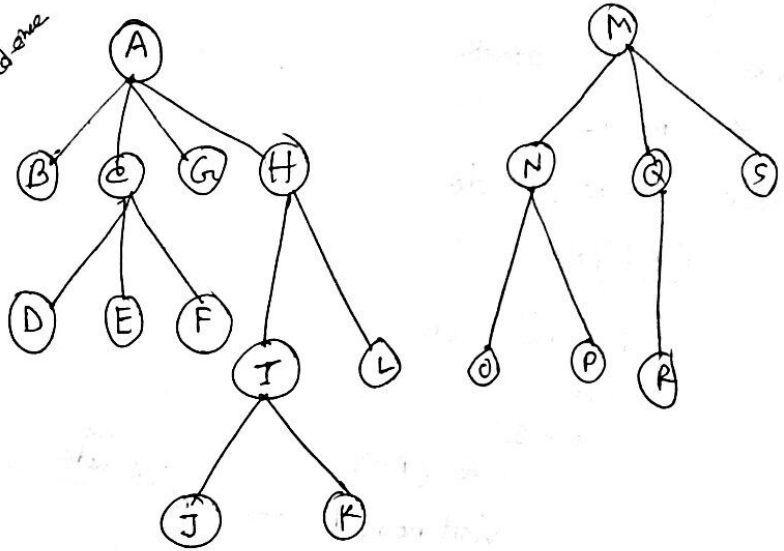
③ একটা node থাকবে

④ বামের copy ডানদিকে দিবে।

prerequisite → 13.07.16 → set of trees

④ A forest & its binary tree. Representation

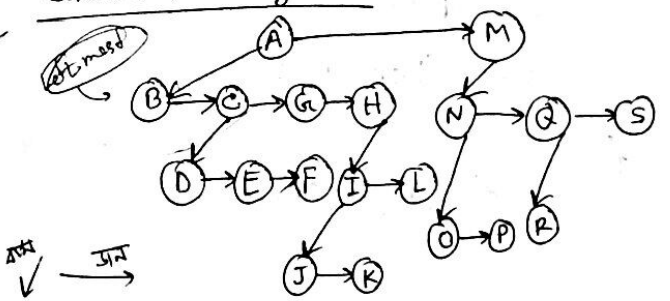
process & Natural correspondence



convert to binary tree

Left link

right node



Exercise

Forest → Binary tree conversion

traverse

Level order

write

Algorithm: Preorder tree traversal of a binary tree.

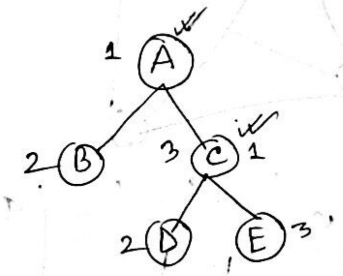
```
S ← empty stack
s ⇐ (root, 1)
while s ≠ empty do
    (P, i) ⇐ S
    if P ≠ nil then
      case
        i = 1:
            s ⇐ (P, 2)
            visit node P  ⟶ print the node
        i = 2:
            s ⇐ (P, 3)
            s ⇐ (Left (P), 1)
        i = 3:
            s ⇐ (Right (P), 1)
```

A B C
A B C D E

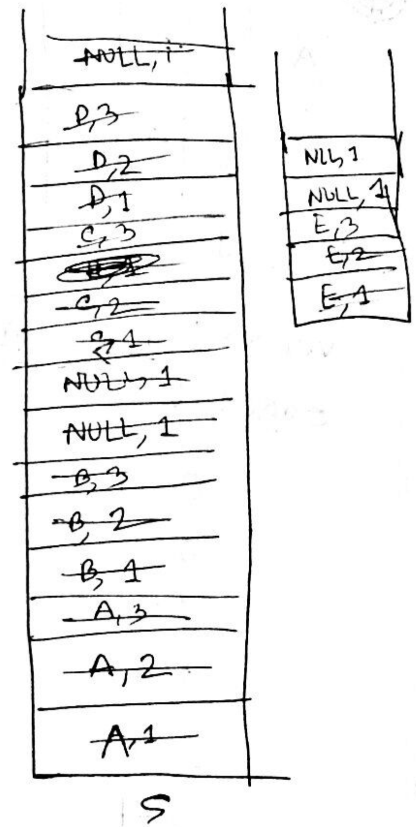
S ⇐ insert
⇐ S (pop)

step — define

1. visit root (1)
2. Left node (2)
3. Right node (3)



stack are update



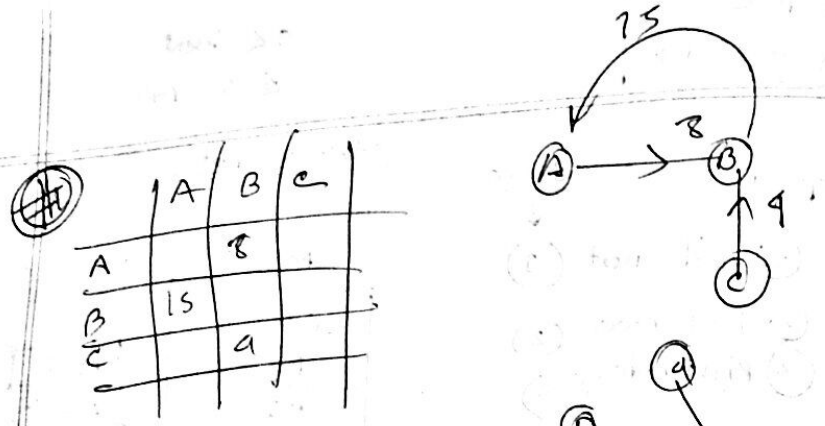
(pop) P = A, i = 1

(case 1) P = A (print A)

~~P = A, i = 2~~

Dnar pop()

Not ⊗



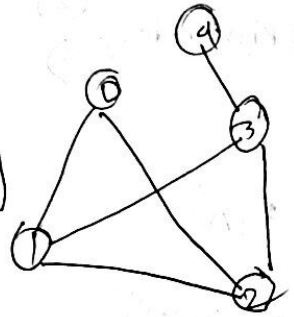
	A	B	C
A		8	
B	15		
C		4	

(#) input:

vertex: 5 → 2D Array

edge: 6

0 1
0 2
1 2
1 3
2 3
3 4



	0	1	2	3	4	5
0		1	1			
1	1		1			
2	1	1		1		
3		1	1		1	
4				1		

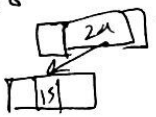
17.07.16 → রবিবার

Algorithm: Inorder binary tree traversal

```
S ← empty stack
S ⇐ (root, 1)
while s ≠ empty do
    (P, i) ⇐ S
    if P ≠ nil then
        if i = 1 then
            S ⇐ (P, 2)
            S ⇐ (left(P), 1)
        else visit node P
            S ⇐ (Right(P), 1)
```

```
(node a nil
if (root != NULL)
)
```

24 15 40 8 20 30 46



→ first input

15

if (15 < 24) if 24 এর left ও nil থাকবে

if (40 > 24) " " NULL

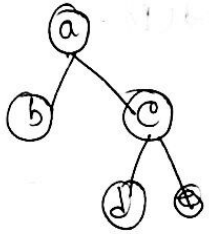
Root → left → Right

H.W: Algorithm 5.7 → Postorder binary tree traversal.

Algorithm:- Level order traversal of a binary tree

Q ← empty queue.
Q ⇐ root
while Q ≠ empty do
    P ⇐ Q
    if P ≠ nil do
        visit node P
        Q ⇐ (Left(P))
        Q ⇐ (Right(P))

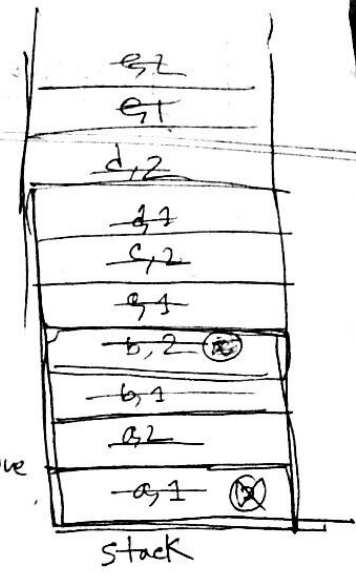
Inorder traversal



b a c

b	a	d	c	e

First Algo



Forrest representive as Binary tree

Book.

Q. level-order traversal of a forrest represening as a binary tree via natural correspondence.

Algorithm:-

Level order traversal of a binary tree
    Q ← empty queue
    Q ⇐ root
    while Q ≠ empty do
        P ⇐ Q.
        while P ≠ nil do
            visit node P.
            if Left(P) ≠ nil then Q ⇐ Left(P)
            P ← Right(P).

19.07.16 → বৃহস্পতিবার

Threaded Binary tree

- A binary tree is threaded by making all right child pointers that would normally be NULL point in the in-order succesor of the node and all left child pointers that would normally be null point to the in-order predecesor of the node

Advantages:

- It is possible to discover the parent of the node from a threaded binary tree, without explicit use of parent pointers or stack
- This can be useful where stack space is limited or where a stack of parent pointers is unavailable.

Types:-

1. Single threaded: Each node is threaded towards either the inorder predecessor or succesor.

10/11/12

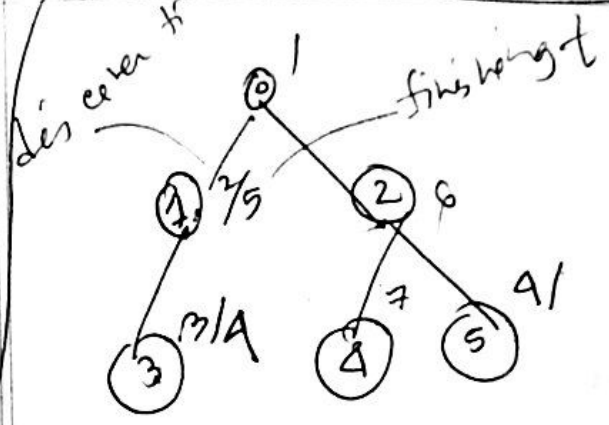
(2) Double threaded each node is threaded towards both in the in-order predecessor and succesor.

Threaded tree traversal:

- we start at the leftmost node in the tree, print it and follow its right thread.
- If we follow a thread to the right, we output the node and continue to its righ
- If we follow a link to the right, we go to the leftmost node, print it and continue.

* inorder में Right - की link use करते
* Preorder " Left " " "

Lab-24.07.10

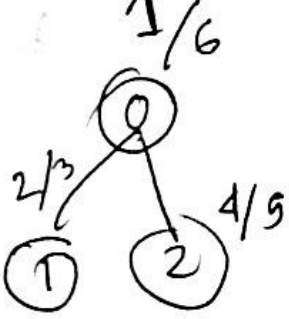


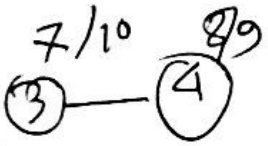
Lab

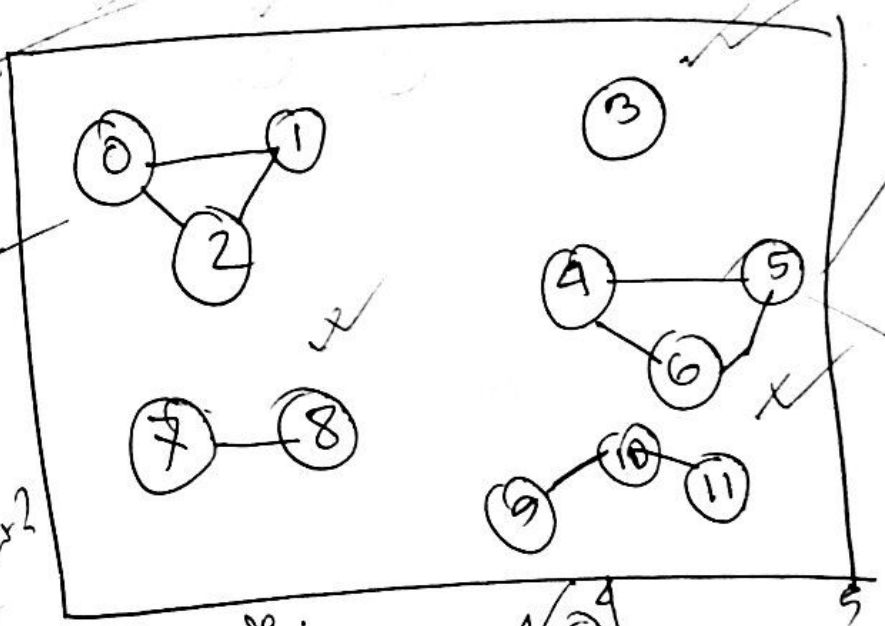
3 1 5 5 2 0

x	y
+2	±1
-2	±1
+1	±2
-1	±2

d - discover time
f - finishing.





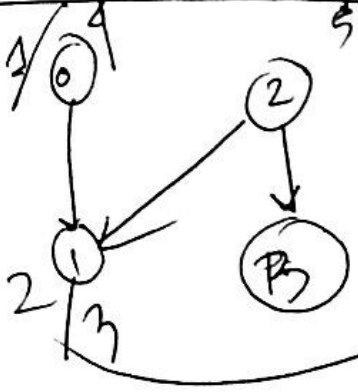


cycle

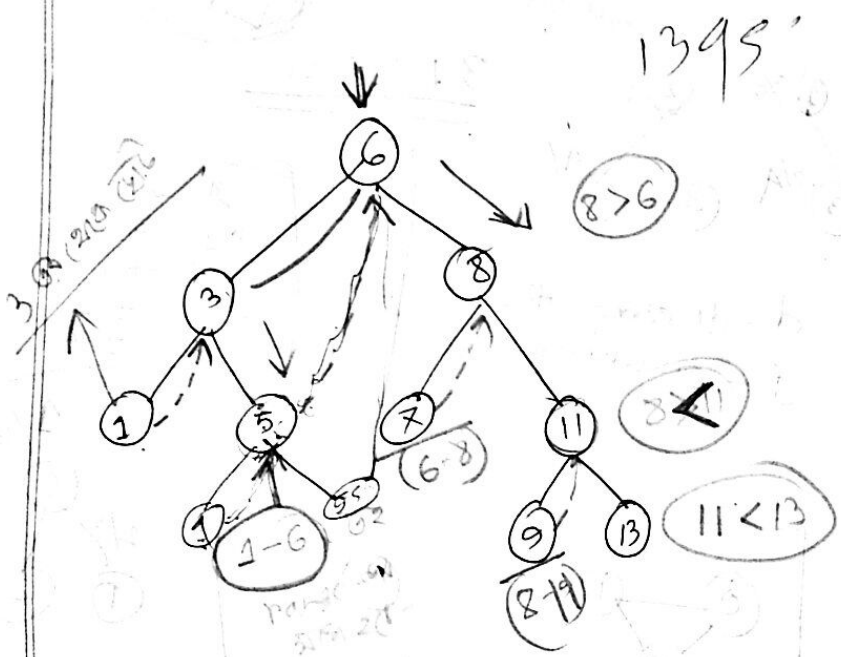
Number of components
Number of cycle

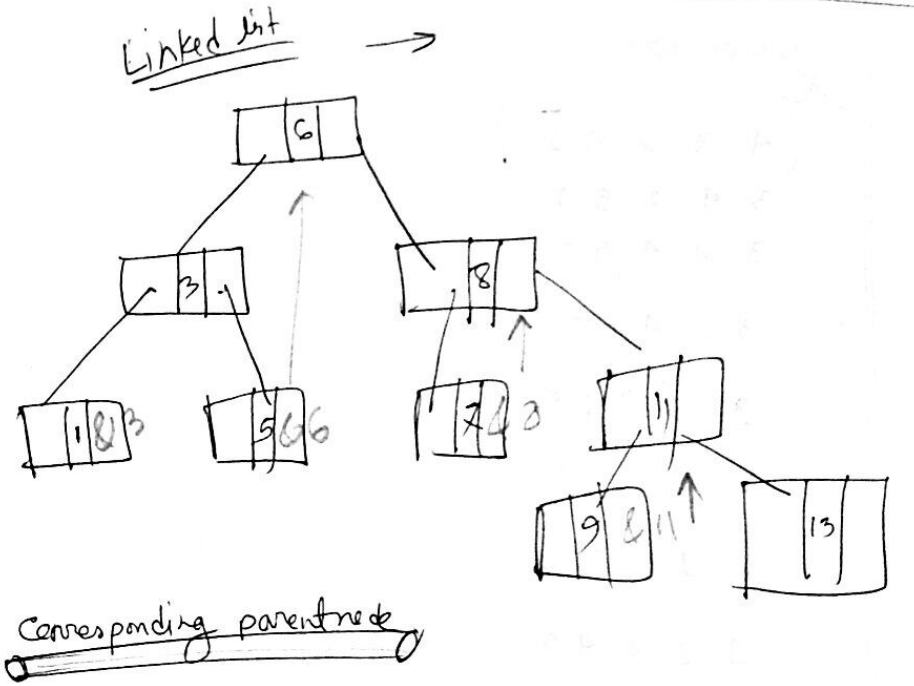
tree = No of components - cycle

finishing



- we start at the leftmost.





bubble sort

[4 3 2 5 1]

3 4 2 5 1

3 2 4 5 1

3 2 4 1 5

2 3 4 1 5

2 3 1 4 5

2 1 3 4 5

1 2 3 4 5

```
int n;

Struct node
{
   int data;
   struct node *link;
}

int main()
{
  struct node *head;
  head = (node*) malloc (sizeof(node));
  if(head == NULL)
  {
    cout<< "Out of Memory" <<endl;
   }
  else
  {
        head -> data = -999999;
        head -> link = NULL;

        cout<<"Enter size:";
         cin>>n;
          for(int i=0; i<n; i++)
          {
            insertLast(head, rand()% 500);
          }
}         printList(head);
```

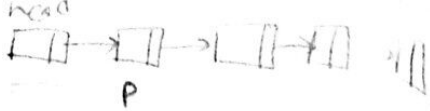
```
void insertLast(node* head, int value)
{
   struct node* newnode;
   newnode = (node*) malloc(sizeof(node));

   if(newnode == NULL) cout<<"out of memory!";

   else{
       node* p = head;
       while(p->link != NULL)
       {
            p = p->link;
       }
        newnode->data = value;
        newnode->link = p->link;
         p->link = newnode;

   }
}
```

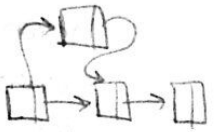
Last node 찾기



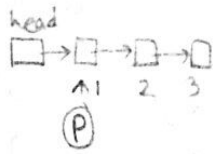

```
void printList(node* head)
{
  node* p = head->link;
  cout<<"List:"<<endl;
  while(p!=NULL)
  {
    cout<<p->data<<" ";
    p=p->link;
  }
  cout<<endl<<endl;
}
```

if p->link != NULL → Last element will not print → loop ও ঢুকবে না।

```
void insertFirst(node* head, int value)
{
  struct node* newnode;
  newnode=(node*)malloc(node);
  if(newnode=NULL)
  else{
      newnode->data=value;
      newnode->link=head->link;
      head->link=newnode;
  }
```

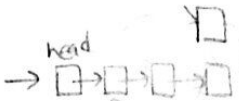


```
void insertAtIndex(node* head, int value, int index)
{
  newnode=
  if
  else{
    int total=1;
    node*p=head->link;
    while(p!=NULL)
    {
      total++;
      if(total==index)
      {
        newnode->data=value;
        newnode->link=p->link;
        p->link=newnode;
        break;
      }
      p=p->link;
    }
}
```



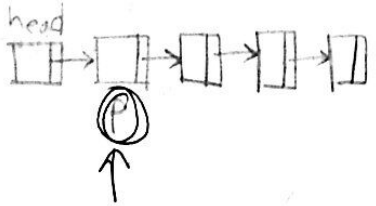
→ not for index first & Last

```
void insertAftervalue (node* head, int value, int aftervalue)
{
  newnode =
  if(newnod ==    )
  else{
     node* p = head->link;
     while (p!=NULL)
     {
        if(p->data == aftervalue)
        {
           newnode->data = value;
           newnode->link = p->link;
           p->link = newnode;
           break;
        }
        p=p->link;
     }
     printList(head);
}
```

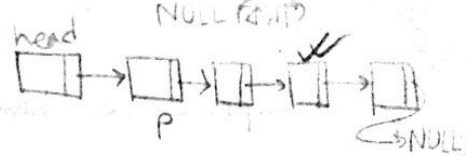



```
void deletefirst (node* head)
{
    node* p = head->link;
    head->link = p->link;
    free(p);
    printList(head);
}
```

find head next node




```
void deleteLast (node* head)
{
   node* p = head->link;
   while(p->link->link!=NULL)
   {
      p=p->link;
   }
   p->link=NULL;
   printList(head);
}
```



element → hashing function → 2907.16

hash funct → h(z)

return proper value / location / index

Chapter 7 (Hashing)

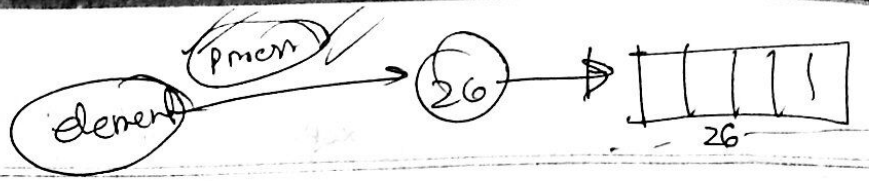
- Suppose, we have an array of m table location T[0], T[1], - - T[m-1]. Say, and given an element z to be inserted we transform it to a location h(z), he is called a hash function.

$0 \le h(z) < m$

- The hash function takes an element to be stored in a table and transform it into a location in the table.

(*) Principles of hash function:

- The hash function should be a function of every bit of the element.
- A hash function should break up naturally occuring clusters of elements.



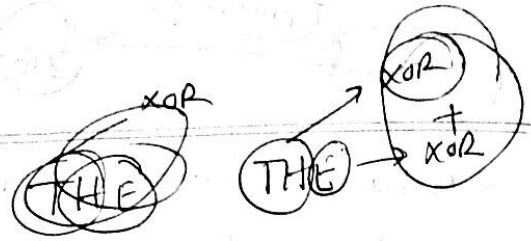
- A hash function should be very quick and easy to compute.

A, B, C, D, E, F, G...
0 1 2 3 4 5

word	Binary form	h(z) = third bit, last two bit	$h_1(z)$	$h_2(z)$	$h_3(z)$
THE	10100 01000 00101 (20, 8, 5)	$(101)_2 = 5$	25	2	23
OF	01111 00110	$(110)_2 = 6$	7	21	21
AND	00001 01110 00100	$(000)_2 = 0$	11	19	4
TO	10100 01111	$(111)_2 = 7$	27	9	7
A	00001	$(001)_2 = 1$	1	1	19
IN	01001 01110	2	7	23	30
THAT	10100 01000 00101	4	9	18	17
IS	01001 10011	3	26	28	2

Not 5 bit

$(1111)_2 = 31$ $(11110)_2 = 30$



$h_1(z)$

T 10100
H 01000 → 11100
xor
E 00101
xor 11100 → 11001 → 25

$h_2(z)$

z mod m

THE = 10100 01000 00101

$= 2^{19} + 2^{12} + 2^8 + 2^2 + 2^0$

$= 20741$

z mod m

$\Rightarrow 20741 \% 31$

$h_3(z)$

20741 → Decimal value of the word

$\Rightarrow 20741 \times \underbrace{.6125423371}_{\text{fixed}}$

~~$\Rightarrow 2\,3\,2\,1\,8\,3$~~

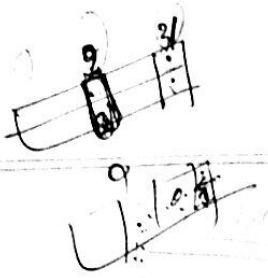
$\Rightarrow 12709.74061$

$\Rightarrow (0.74061 \times 30) + 1$

$\Rightarrow 23.2183$

$\Rightarrow 23$

26.07.16 ⇒ মঙ্গলবার



Algo:- the

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 ← i
            i ← link(i)
    until found or i = -1

if not found then
    if T[loc] is empty then T[loc] ← z
    else
        repeat free ← free − 1 until T[free] is empty
        if free = -1 then overflow
        *
        else
            link(prev_i) ← free
            T(free) ← z
            link(free) ← -1

Algo:- search & insertion into a hash table with collision Resolved by coalesced chaining

compression → (H.W) → flow

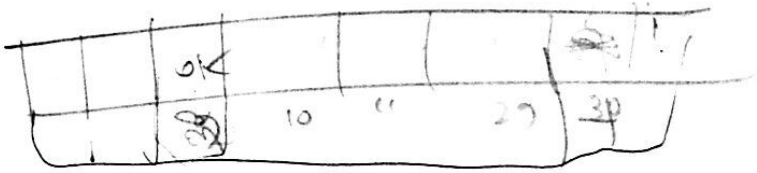
Division: $h(z) = z \bmod m$

Multiplication: $h(z) = \lfloor m(z.v \bmod 1) \rfloor$

v = real number of degree $0 < v < 1$

Collision Resolution:-

① coalesced chaining



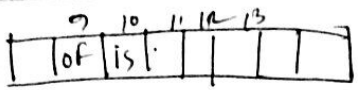
Allocation and Linked

27.07.16 => बुधवार

Page-350

Data structure
Edward M Reingold
Wilfried

Algorithm : Linear Probing



searching

```
i ← h(z)
found ← false
while T[i] is not empty and not found do
    if T[i] = z then found ← true
    else i ← (i+1) mod m
```

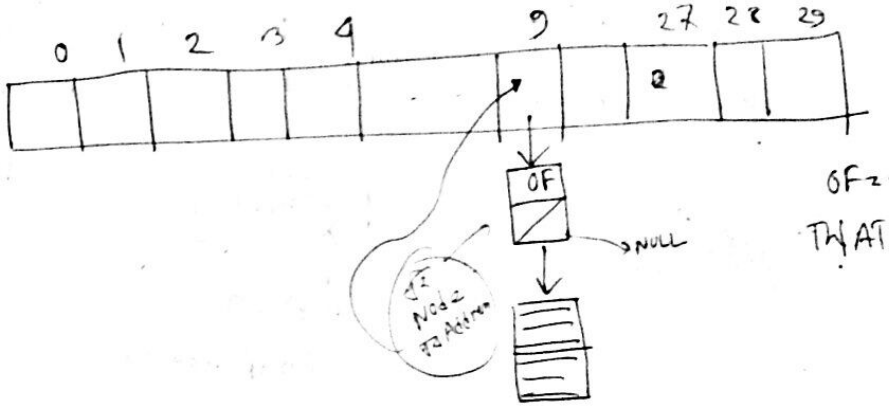
inserting

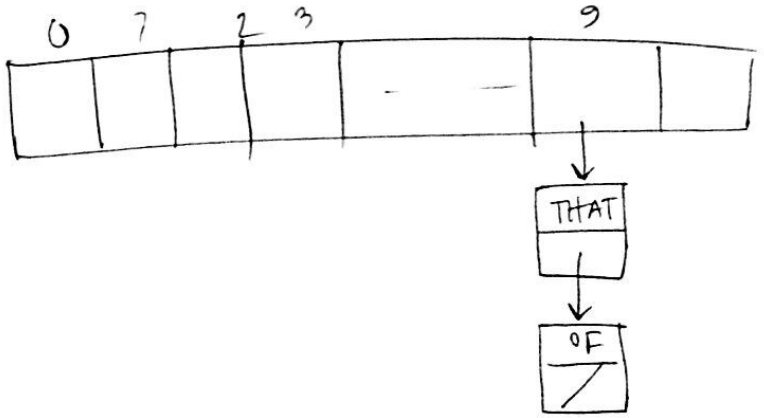
```
if not found then
    if n = m-1 then table overflow
    else n ← n+1
         T[i] ← z
```

initially n=0

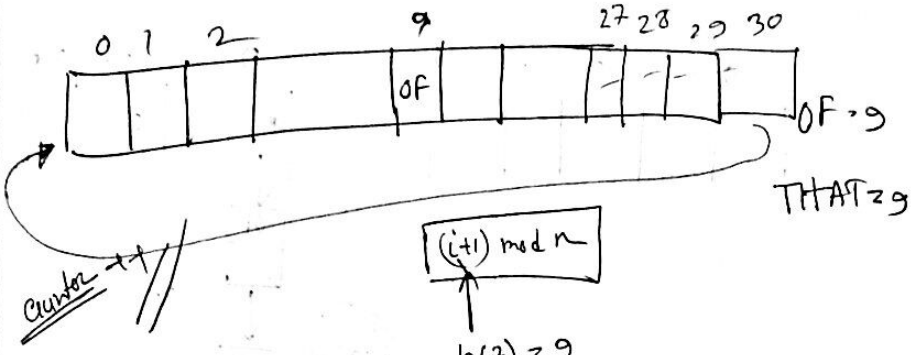
Double Hashing

2. Seperate chaining





③ Linear Probing



31.0816 → বাইনারি

H.W

① Binary – 0, 1 → Base-2

integer ② Decimal – 0–9 → 10

logical ③ Hexadecimal – 0–F → 16

④ Octal → 0–7 → 8

	Hexa		
0	10	20	
1	11	21	
2	12	22	
⋮	⋮	⋮	
F	1F	28	
		29	
		2A	
		⋮	
		2F	

Binary → Hexa/logical

0–F → 15 → 4 bit

0010 1001 1011 ₂

2 9 B

(29B)₁₆

Binary → octal

0–7 → 3 bit

1 2 3 3

001 010 011 011

(1233)₈

⑤ Character string Representation

⑥ Floating

$(10011)_2 = 1\times2^4 + 0\times2^3 + 0\times2^2 + 1\times2^1 + 1\times2^0$

$(10011)_{2\to b} = a_4b^4 + a_3\times b^3 + a_2\times b^2 + a_1\times b^1 + a_0\times b^0$

Number of digit = K = 5

$= \sum_{i=0}^{4} a_i b^i$

$\sum_{i=0}^{K-1} a_i b^i$

$(10011)_{10} = 1\times10^4 + 0\times10^3 + 0\times10^2 + 1\times10^1 + 1\times10^0$

1010 × 1010 × 1010 × 1010

$= 1\times(1010)^{100} + 0\times(1010)^{011} + 0\times(1010)^{10} + (1\times1010^1 + 1\times1010^0)$

$= 100111000|0000 + 1010 + 1$

$= (100111000|1011)_2$

b = base

a = digit

Horner's Method :-

current value

Decimal

convert by this formula

$\sum_{i=0}^{K-1} a_i b^i$

$= a_{K-1}b^{K-1} + a_{K-2}b^{K-2} + \dots + a_3b^3 + a_2b^2 + a_1b + a_0$

$= a_{K-1}b^{K-1} + a_{K-2}b^{K-2} + \dots + a_3b^3 + a_2b^2 + a_1b + a_0$

$= (a_{K-1}b^{K-2} + a_{K-2}b^{K-3} + \dots + a_3b^2 + a_2b + a_1)b + a_0$

$= ((a_{K-1}b^{K-3} + a_{K-2}b^{K-4} + \dots + a_3b + a_2)b + a_1)b + a_0$

$= (((a_{K-1}b^{K-4} + a_{K-2}b^{K-5} + \dots + a_3)b + a_2)b + a_1)b + a_0$

$(10011)_2 = ((((1)2+0)2+0)2+1)2+1$

$= 19$

H.W

Signed Integers

total number of bits

① 1's complement — if n is positive integer, $-n = 2^k - n$

② 2's

③ signed Magnitude

④ offset Representation

$n = 5$

$= (0101)$

1010

$+\ 1$

1011

$-5 = 2^4 - 5$

$-5 = 11$

$= 1011$ //

if n is negative:

$n = 2^{K-1}$

$-5 = 2^3 + (-5)$

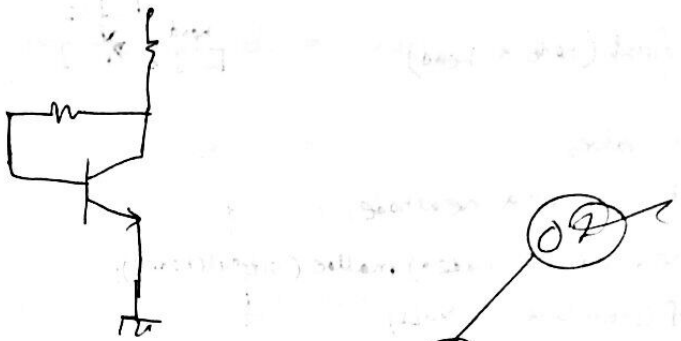
$2^3 = (0011)$.

$+n$

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

$-n$

$-n = 2^{K-1} - n$



$-V_{cc} + I_e' R_e + I_B R_B + V_{BE} = 0$

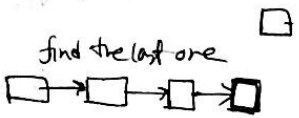
$\Rightarrow -V_{cc} + I_c R_e + I_B R_B + V_{BE} = 0$

$\Rightarrow -V_{cc} + \beta I_B R_e + I_B R_B + V_{BE} = 0$

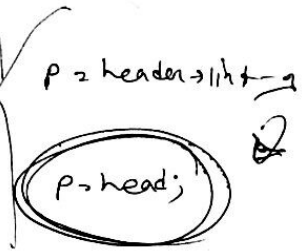
$\Rightarrow -V_{cc} + I_B (\beta R_e + R_B) + 0.7 = 0$

$\Rightarrow I_B = \frac{V_{cc} - 0.7}{\beta R_e + R_B}$


```
void insertfirst (node * head)
{
    cin>>value;
    struct node * newnode;
    newnode =(node*) malloc (sizeof(node));
    if (newnode == NULL)
    {
        cout<<"out of memory"<<endl;
    }
    else
    {
        newnode->data = value
        newnode->link = head->link;
        head->link = newnode;
    }
```



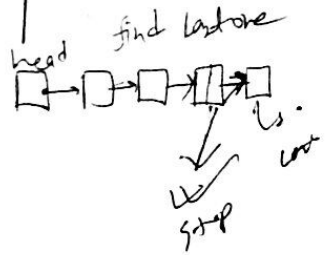

```
void insertLast (node * head)
{
    else{
        node *p = head->link;
        while (p->link!=NULL)
        {
            p=p->link;
        }
        newnode->data=val;
        newnode->link = p->link;
        p->link = newnode;
    }
```

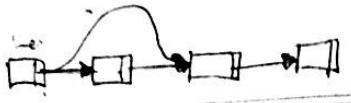



```
void printList (node * head)
{
    node * p=head->link;
    while (p!=NULL)
    {
        cout<<p->data<<" ";
    }   p=p->link;
    cout<<endl;
}
```

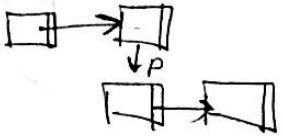
```
# deleteLast (node *head)
{
    node *p = head;
    while(p->link->link!=NULL)
    {
        p=p->link;
    }
    ->link=NULL
}
```

```
isEmpty (node *)
if (head->link==NULL)
    return true
else return false.
```

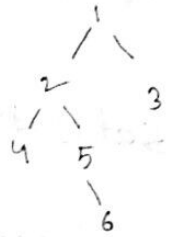


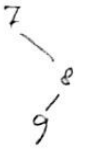


```
void deleteFirst (node* head)
{
    node *p = head->link
    head->link = p->link
    free(p);
}
//
```



for(i=1, to no) if we
if visit[i] = 0.
DFS_VISIT(i).

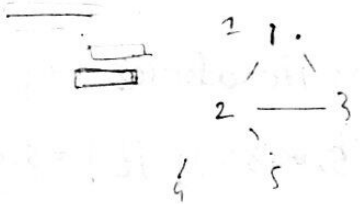


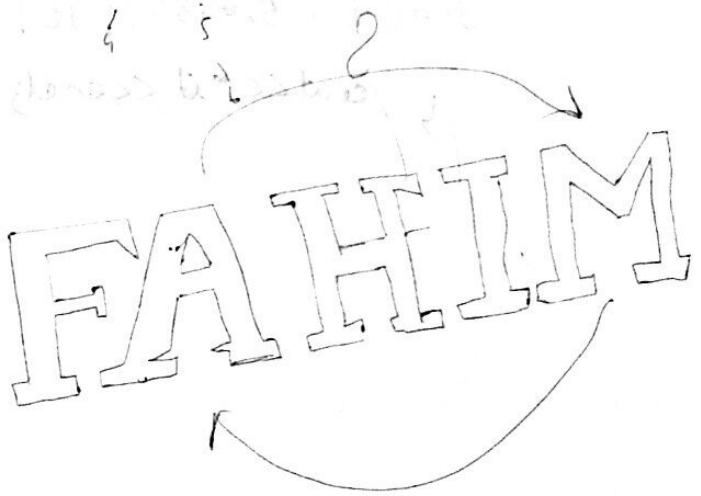


DFS_V ( )
{

}

10





set

set<int>s.

* সবগুলোকে 1 বার করে রাখে
* sort করে রাখে

```
s.insert(10);
  "  (9);
     (5);

set<int>:: iterator it;
for(it = s.begin(); it != s.end(); it++) {
    cout<< *it <<endl;
}
```

String a;
getline (cin, a)

```
while( cin>> vale)
{

}

char line[101].
while (cin.get (line,100)){

    Two Alias Endure breaks
}

char ara[100];
while(NULL != gets(ara)){
}
```

queue

```
queue<int> q.
q.push();
while(!q.empty()){
cout<<q.front<<endl. // printtile the frone
} q.pop(); // removing that one
#priority_queue<int> a.
  a.push
  whol <name>
```

```
bool compare(data A, data B){
  if(a.iname == b.iname){
    if(a.height >= b.height){
    | if(a.weight >= b.weight){
      . return strlen(a.name) > strlen(b.name);
    . else return a.weight < b.weight
    else if return a.height > b.height
  else return a.iname > b.iname
```

heapsort → ds
mergesort → ~~hard~~ divide-e
quicksort → randomization

naive Algo

Searching for Patterns

```
int find_match(char *P, char *t)
{
    int i, j;
    int plen, tlen;

    plen = strlen(p);
    tlen = strlen(t);

    for(i=0; i<=(tlen-plen); i++){
        j=0
        while((j<plen) && (t[i+j]==P[j]))
        {
            j++;
            if(j==plen) return(i);
        }
        return(-1);
}
```

~~find(s);~~