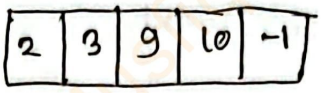
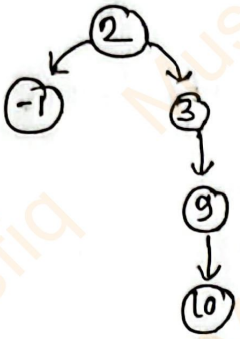
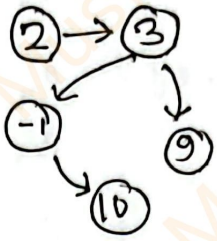
27-05-23

"Data Structure and Algorithms"

Sorting Algorithm →







Sorting Algorithm → 

* ছোট থেকে বড় → aseending
* বড় থেকে ছোট → desending order.

Types of Sort:
- Insertion sort
- Merge sort
- counting sort
- Quick sort
- bubble sort
- Selection sort

"Insertion Sort"

Process ↓

(i) প্রথম থেকে 2nd elements থেকে

(ii) Starting will be run · n times.

(iii) Asending → 1, 2, 3, 4, 5, 6.

(iv) desending → 6, 5, 4, 3, 2, 1

21.07.23

Insertion Sort.

key → যেটা সঠিক জায়গায় আসবে [5, 2, 4, 6, 1 3]

Insertion Sort(A, n)	cost	Times
1, for(i = 2 to n){	c_1	n
2, key = A[i]	c_2	$n-1$
3, j = i - 1	c_3	$n-1$
4, while (j > 0 && A[j] > key){	c_4	$T = t_2+t_3+t_4+t_5+t_6$ (2 2 1 5 4 = 14)
5, A[j+1] = A[j]	c_5	$T-(n-1)$
6, j = j - 1 }	c_6	$T-(n-1)$
7, A[j+1] = key	c_7	$n-1$

$$T(n) = c_1 n + c_2(n-1) + c_3(n-1) + c_4 T + c_5(T-(n-1)) + c_6(T-(n-1)) + c_7(n-1)$$

$$= c_8 T + c_9 n + c_{10}$$

Best case → array already sorted [1, 2, 3]
$(O(n))$

Worst case → array ⁿ ⁿ in decending Order [3, 2, 1]
$(O(n^2))$

C-2
30-05-2023

Sorted →

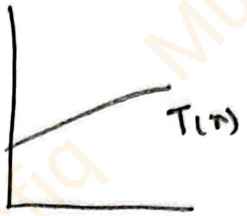
$$T(n) = c_8(n-1) + c_9 n + c_{10}$$

$$= c_8 n - c_8 + c_9 n + c_{10}$$

$$= n(c_8 + c_9) + c_{10} - c_8$$

$$= an + b$$

$$= O(n)$$



unsorted → $2 + 3 + 4 + \cdots + n$

$$= 2 + 3 + 4 + \cdots + n$$

$$= 1 + 2 + 3 + 4 + \cdots + n - 1$$

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

$1 + 2 + 3 + \cdots$

Worst case

$$T(n) = c_8\left[\frac{n(n+1)}{2} - 1\right] + c_9 n + c_{10}$$

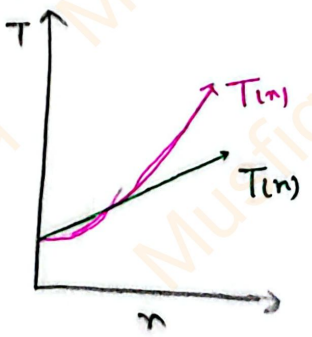
$$= c_8\left[\frac{n^2}{2} + \frac{n}{2} - 1\right] + c_9 n + c_{10}$$

$$= c_8 \frac{n^2}{2} + c_8 \frac{n}{2} - c_8 + c_9 n + c_{10}$$

$$= \frac{c_8}{2} n^2 + n\left(\frac{c_8}{2} + c_9\right) + (c_{10} - c_8)$$

$$= an^2 + bn + c$$

$$= O(n^2)$$



Upper bound Notation

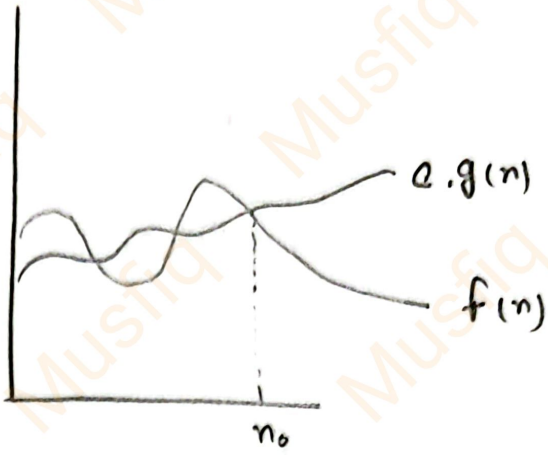
$\exists$ → there exist

$\forall$ → for all

$f(x) = O(g(n))$ বলতে কী বুঝি?

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

$an^2 + bn + c = O(n^2)$



$0 \leq f(n) \leq c.g(n)$

for some c and n_0.
where, $n \geq n_0$

Qn: Show that $f(n) = 10n^2 + 7n + 4 = O(n^2)$
$g(n)$

$0 \leq f(n) \leq c.g(n)$

Let $c = 15$

$n=1$, $f(1) = 10 + 7 + 4 = 21$, $c.g(n) = 15$ ($15 \to c$, $n^2 = 1 \to g(n)$)

$n=2$, $f(2) = 10 \times 4 + 14 + 4 = 58$, $c.g(n) = 60$

$n=3$, $f(3) = 90 + 21 + 4 = 115$, $c.g(n) = 135$

$c=15$, $n_0 = 2$ → satisfied.

Show that $f(n) = 10n^2 + 4n = O(n^3)$

$= On^3 + 10n^2 + 4n = O(n^3)$

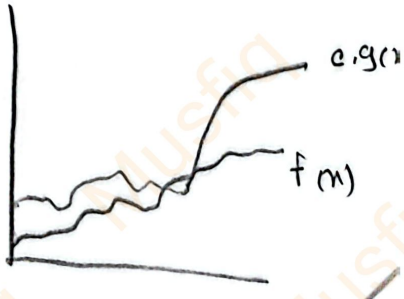
$c=5$

$n=1$, $f(1) = 14$; $c.g(n) = 5$

$n=2$, $f(2) = 48$; $c.g(n) = 40$

$n=3$, $f(3) = 102$; $c.g(n) = 135$

$c=5$, $n_0 = 3$



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

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

$= O(n^3)$

$On^3 \rightarrow On$
ফিরে হালকা না

Lower Bound Notation

$0 \leq f(n) \leq c.g(n) \rightarrow$ Upper

$0 \leq c.g(n) \leq f(n) \rightarrow$ lower

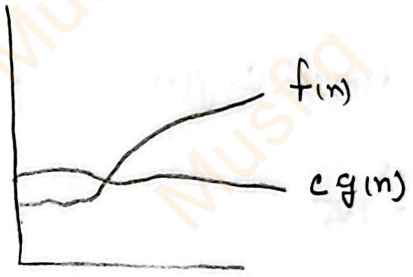
$5n^2+6n+4 = O(n^3)$

$c = 2$,

$n = 1$, $f(1) = 5+6+4 = 15$, $c.g(n) = 2$

$n = 2$, $f(2) = 20+12+4 = 36$, $c.g(n) = 8$

$c = 2, n = 1$



Asymptotic Tight Bound (Θ):

$f(n) = 5n^2+6n+3$

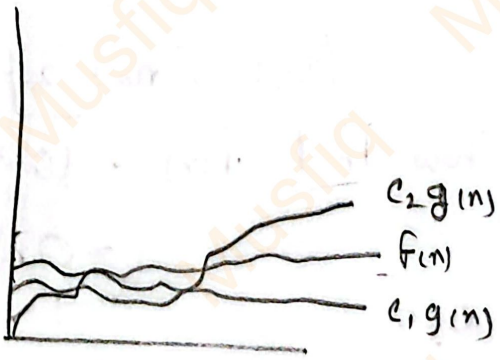
condition $\rightarrow c_2 g(n) \leq f(n) \leq c_1 g(n)$

$f(n) = 5n^2+6n+3 = O(n^2)$

$c = 5, n = 3$

$f(n) = 5n^2+6n+3 = \Omega(n^2)$

$c = 2, n = 2$



3-06-2023
L-3

unting Sort

(i) Extra array লাগে।
(ii) Maximum number Asymptotic Assumption এর উপর নির্ভর করে।
(iii) Negative মান থাকে না।

Simulation →

A - 2 5 3 0 2 3 0 3 $\quad k=5$

C - $\frac{2}{0}$ $\frac{0}{1}$ $\frac{2}{2}$ $\frac{3}{3}$ $\frac{0}{4}$ $\frac{1}{5}$

Adding

C : $\frac{2}{0}$ $\frac{2}{1}$ $\frac{4}{2}$ $\frac{7}{3}$ $\frac{7}{4}$ $\frac{8}{5}$

Adding B:

A : $\frac{2}{1}$ $\frac{5}{2}$ $\frac{3}{3}$ $\frac{0}{4}$ $\frac{2}{5}$ $\frac{3}{6}$ $\frac{0}{7}$ $\frac{3}{8}$

B : $\frac{0}{1}$ $\frac{0}{2}$ $\frac{2}{3}$ $\frac{2}{4}$ $\frac{3}{5}$ $\frac{3}{6}$ $\frac{3}{7}$ $\frac{8}{8}$ $\frac{0}{1}$ $\frac{0}{2}$ $\frac{2}{3}$ $\frac{2}{4}$ $\frac{3}{5}$ $\frac{3}{6}$
$\frac{5}{7}$ $\frac{5}{8}$

C : $\frac{2}{0}$ $\frac{2}{1}$ $\frac{4}{2}$ $\frac{7}{3}$ $\frac{7}{4}$ $\frac{8}{5}$ $\frac{2}{0}$ $\frac{2}{1}$ $\frac{4}{2}$ $\frac{7}{3}$ $\frac{7}{4}$ $\frac{8}{5}$
1 0 3 6 7 1 0 3 6 7
0 2 5 4 2 5

Final C : $\frac{0}{0}$ $\frac{2}{1}$ $\frac{2}{2}$ $\frac{4}{3}$ $\frac{4}{4}$ $\frac{7}{5}$

Show that $f(n) = 5n^2 + 6n + 3 = \Theta(n^2)$

$= O(n^2)$

$= \Omega(n^2)$

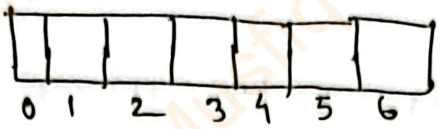
Condition ↓

$c_2 g(n) \leq f(n) \leq c_1 g(n$

Annay Mapping :

L_1 = First index.

A [0 .... 6] →



l_1 u_1

Formula →

$$addr(A[i]) = b + (i - l_1) \times L$$

b → base address, l_1 → array first index, L → variable type size

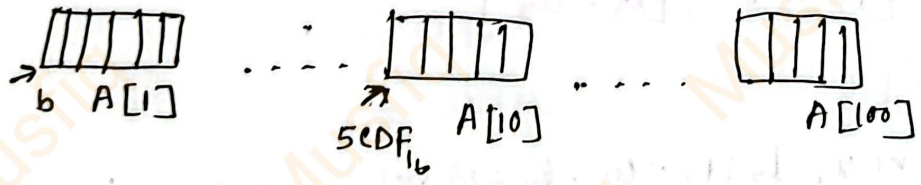
(1D)

$$add(A[3]) = b + (i - l_1) \times L$$
$$= 10 + (3 - 0) \times 4$$
$$= 10 + 12$$
$$= 22.$$

(2D) double [1 .... 100]

l_1 u_1

Given, $addr(A[10]) = 5CDF_{16}$. Find the address of A[60]=?



$$add(A[10]) = b + (i - l_1) \times L$$
$$b = addr(A[10]) - (i - l_1)L$$
$$= 5CDF_{16} - (10 - 1) \times 8$$
$$= 5CDF_{16} - 72$$
$$= 23775 - 72$$
$$= 5C97_{16}$$

Now,

$$addr(A[60]) = b + (i - l_1) \times L$$
$$= 5C97 - (60 - 1) \times 8$$
$$= 5C97 + 472$$
$$= 23703 + 472$$
$$= 23931$$
$$= 24175$$
$$= (5E6F)_{16}$$

(2D) $A[0 \cdots 2][0 \cdots 3]$ (l_1 u_1 l_2 u_2)

Row Major:

$$addr(a[i],[j]) = b + (i-l_1) \times (u_2-l_2+1) \times L + (j-l_2) \times L$$

$$addr(a[1][2]) = 10 + (1-0) \times (3-0+1) \times 4 + (2-0) \times 4$$

$$= 34.$$

Column Major:

$$addr(A[i][j]) = b + (j-l_2) \times (u_1-l_1+1) \times L + (i-l_1) \times L$$

$A[1 \cdots 3][1 \cdots 4]$

$$addr(A[1][2]) = 10 + (2-1) \times (3-1+1) \times 4 + (1-1) \times 4$$

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

$$= 10 + 12$$

$$= 22.$$

(3D) → $a[l_1 \cdots u_1][l_2 \cdots u_2][l_3 - u_3]$

$a[i][j][k]$

$$= b + (i-l_1) \times (u_2-l_2+1) \cdot \frac{(u_3-l_3+1)}{3} \times L + (j-l_2) \times \frac{(u_3-l_3+1)}{3} \times L + \frac{(k-l_3)}{3} L$$

(4D) → $a[l_1 \cdots u_1][l_2 \cdots u_2][l_3 - l_3][l_4 \cdots u_4]$

$a[i][j][k][l]$

$$= b + (i-l_1) \times (u_2-l_2+1)(u_3-l_3+1)(u_4-l_4+1) \times L + (j-l_2) \times (u_3-l_3+1) \cdot (u_4-l_4+1) \times L + (k-l_3) \times (u_4-l_4+1) \times L + (x-l_4) L$$

Linear Search

```
int LinearSearch (int b[], int skey) {
int i;
for(i=0; i<size; i++)          — n+1
  if b[i] == skey              — n
    return i                   — 1/0
return -1                      — 1/0
```

Time Complexity

$T(n)_w = c_1(n+1) + c_2 n + c_3 1$

$= O(n)$

$= \Omega(n)$

$= \Theta(n)$

Best case : $T(n)_b : c_1 + c_2 + c_3 = c(1)$

$= O(1)$

$= \Omega(1)$

$= \Theta(1)$

Average Case:

$$\text{Avg time} = \frac{\text{Time Required for all possible case}}{\text{Total number of case}}$$

$$= \frac{(1+2+3+\cdots+n)c}{n}$$

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

$$= \frac{n+1}{2} \times c$$

$$= O(n)$$

Binary Search:

```
Binary Search (A, n, key)
1,  i = 1 , j = n
2,  while (i <= j)
3,      mid = ⌊(i+j)/2⌋
4,  if (array [mid] == key)
5,      return mid
6,  else if (A[mid] < key)
7,      i = mid + 1
8,  else if (A[mid] > key)
9,      j = mid - 1
10, return -1
```

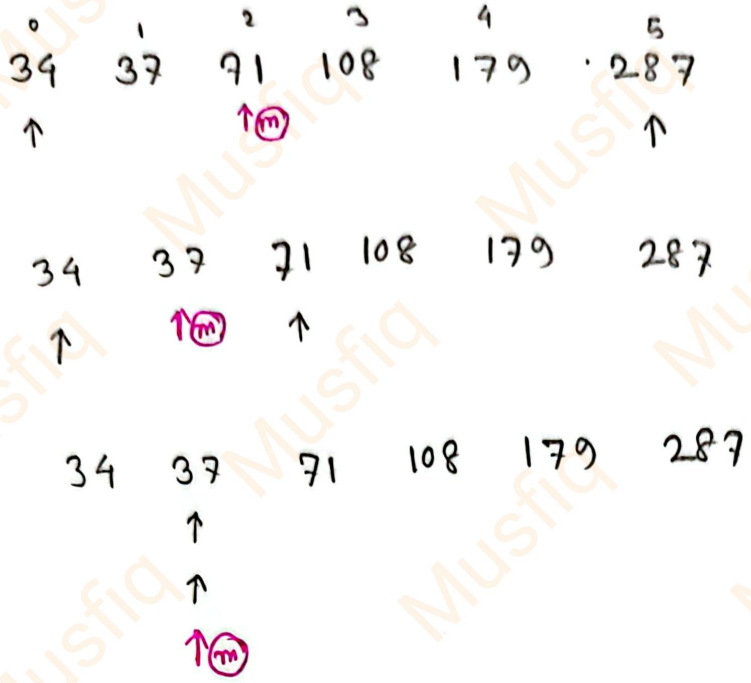
```
RBin Search (A, l, h, key)
 if (l > h)
  · return -1
 else · mid = ⌊(l+h)/2⌋
 if key == A[mid]
     return mid
 if A[mid] < key
     Return RBinSearch (A, mid+1, j, key)
 if A[mid] > key
     return RBinSearch (A, i, mid-1, key)
```

x, y, z, P, n, t / high = n-1

key = y low = 0

x = 34, y = 37, z = 71, P = 108, n = 179, t = 287

key = 37



now, found in index 1

Ans

while(l ≤ h)

$mid = \lfloor \frac{0+5}{2} \rfloor$

$= \lfloor 2.5 \rfloor$

$= 2$

A(m) > k

h = m - 1

A(m)

l = mid + 1

$m = \frac{0+1}{2}$

$= \lfloor 0.5 \rfloor$

$= 0$

$1 \leq 1$

$m = \lfloor \frac{1+1}{2} \rfloor$

$= \lfloor 1 \rfloor$

$= 1$

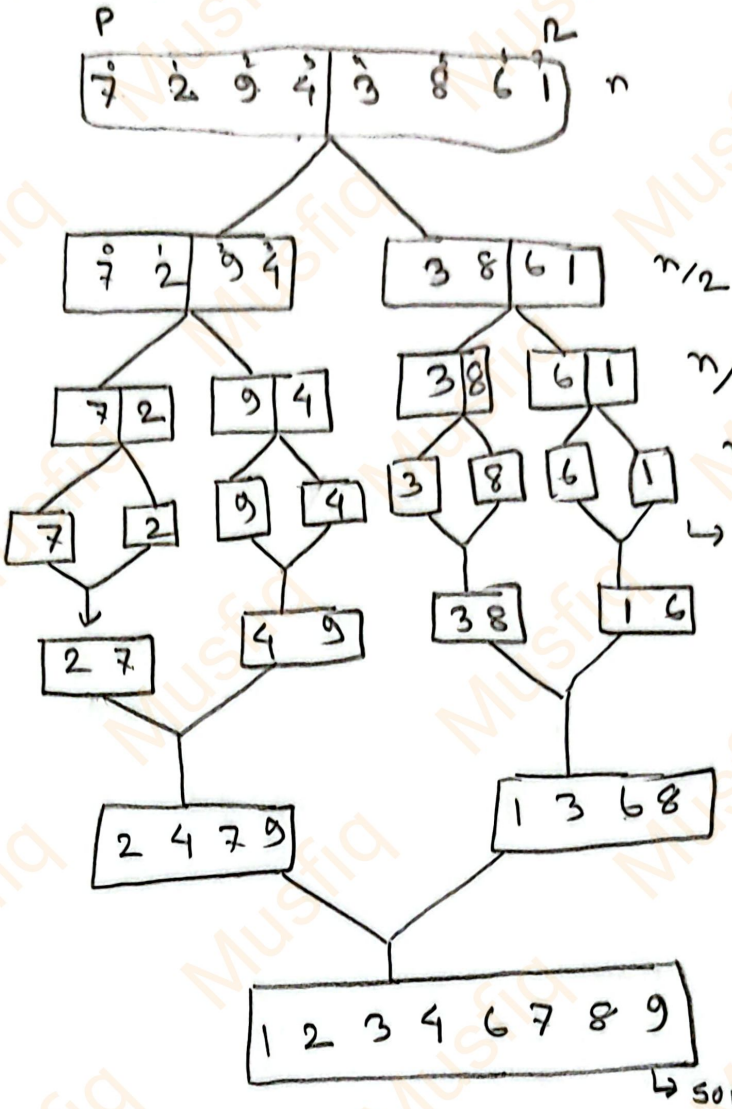
A[m] = = k

Merge Sort - Simulation

7 2 9 4 3 8 6 1

$p = \lfloor \frac{0+7}{2} \rfloor$
$= \lfloor 3.5 \rfloor$
$= 3$

$\lfloor \frac{0+3}{2} \rfloor$
$= \lfloor 1.5 \rfloor$
$= 1$



Quick sort Partition

"Partition Function"

F L

20 10 5 6 7 3 4

3 4 5 6 7 10 20 — sorted.

① 'i' keep track of the elements smallest then the Pivot (backward)

② 'j' keeps track of the elements that are yet to be explore (Forward)

· 'j' keep track of the element that are greater the the Pivot (backward)

আমরা

① i শুরু হবে 0/-1 থেকে, (if index → 1, if index → 0) j শুরু হবে First index থেকে

② j compare হবে Pivot value এর সাথে, যদি Pivot থেকে বড় হয় তা হলে j++

Partition (A, P, R) (A → array, P → F, R → L)

```
1   Pivot Position = random (P,R)
2   x = A[Pivot Position]
3   i = P-1
4   for (j=1 to R)
5     if (j == Pivot Position)
6       continue
7     if (A[j] < x)
8       i = i+1
9       swap A[i] with A[j]
10  i=i+1
11  swap A[i] with A[Pivot Position]
12  return i.
```

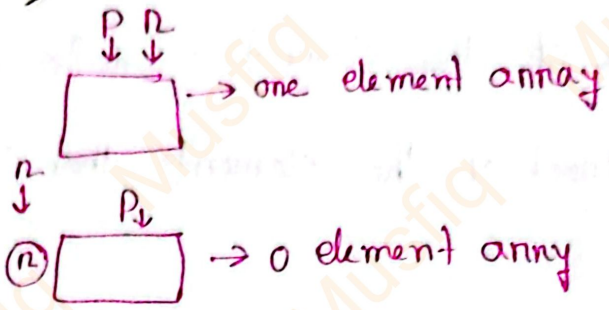
③ Pivot না ছোট না i++ and value of j swap with value of i.

④ j কখনই Pivot position এ যাবে না, যদি Pivot শেষ index হয় তাহলে j close হবে, i++ হয়ে pivot value এর সাথে exchange/swap করবে।

⑤ Pivot mid index হলে ঐ position j skip করবে।

i -> <-L

Quick sort (A, P, R)
if (P < R)
q = Partition (A, P, R)
Quick Sort (A, P, q-1)
Quick Sort (A, q+1, R)



Time complexity :

(i) Best case → O(nlogn)
(ii) Worst case → O(n²)
(iii) Average case → O(nlogn)

Qn – Given Array. 2 8 7 1 3 5 6 4, Start with 1 index

P R

i=0, j=1 → 2 8 7 1 3 5 6 4

1 2 3 4 5 6 7 8

Pivot x = 4

i=1, j=1 → 2 8 7 1 3 5 6 4

2 8 7 1 3 5 6 4
(i) (j)

2 8 7 1 3 5 6 4
(i) (j)

2 8 7 1 3 5 6 4
(i) (j)

2 8 7 1 3 5 6 4
(i) (j)

2 1 7 8 3 5 6 4
(i) (j)

2 1 7 8 3 5 6 4
(i) (j)

2 1 7 8 3 5 6 4
(i) (j)

2 1 3 8 7 5 6 4
(i) (j)

2 1 3 8 7 5 6 4
(i) (j)

2 1 3 8 7 5 6 4 → i++ , j continue
(i) (j)

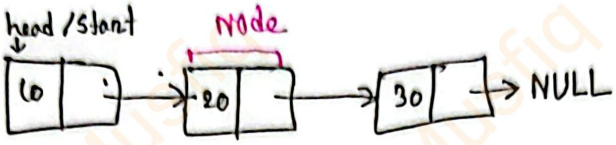
2 1 3 4 7 5 6 8 → i swap with A[Pivot]
(i) (j) return i;

nked List"

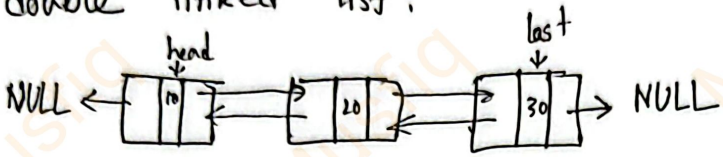
(i) Single Linked list
(ii) double "
(iii) Circular linked list

(1) Linear / Single linked list ;

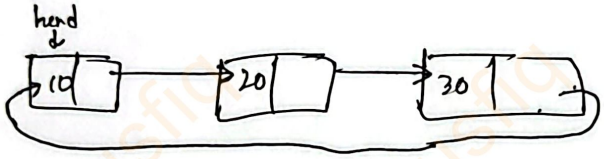
box = node



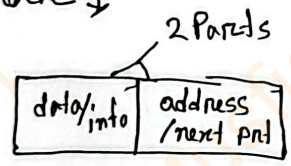
(2) double linked list :



(3) Circular linked list :



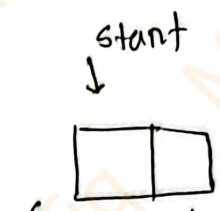
(1) Each node :



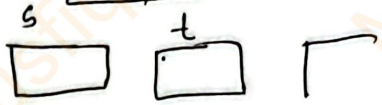

```
struct Node {
    int value;
    struct Node* next_ptr
};
```

* Insert
* Delete
* Search
* Print

① start = (node*) malloc(sizeof(node));



② temp=(node*) malloc(sizeof(node));



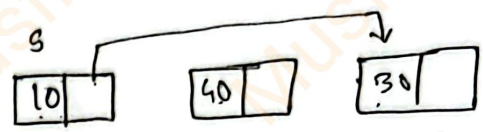
③ temp1=(node*)malloc(sizeof(node)));

④ start->data = 10;

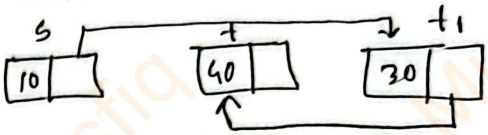
⑤ temp->data = 40;

⑥ temp->data = 30;

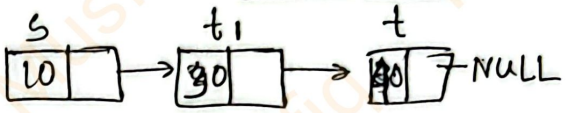
⑦ start->next = temp1



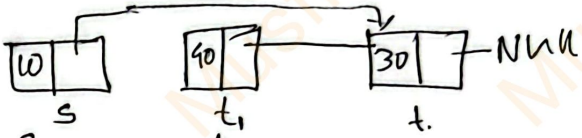
⑧ start->next->next = temp



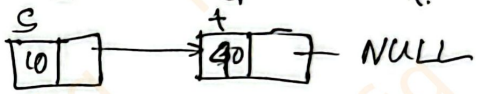
⑨ temp->next = NULL



⑩ start->next = temp1->next



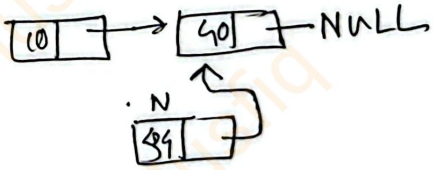
⑪ Free(temp1)



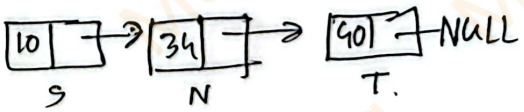
⑫ newitem = (node*) malloc (sizeof(node));

⑬ newtiem -> data = 34

⑭ newtiem -> next = start -> next



⑮ start -> next = newItem

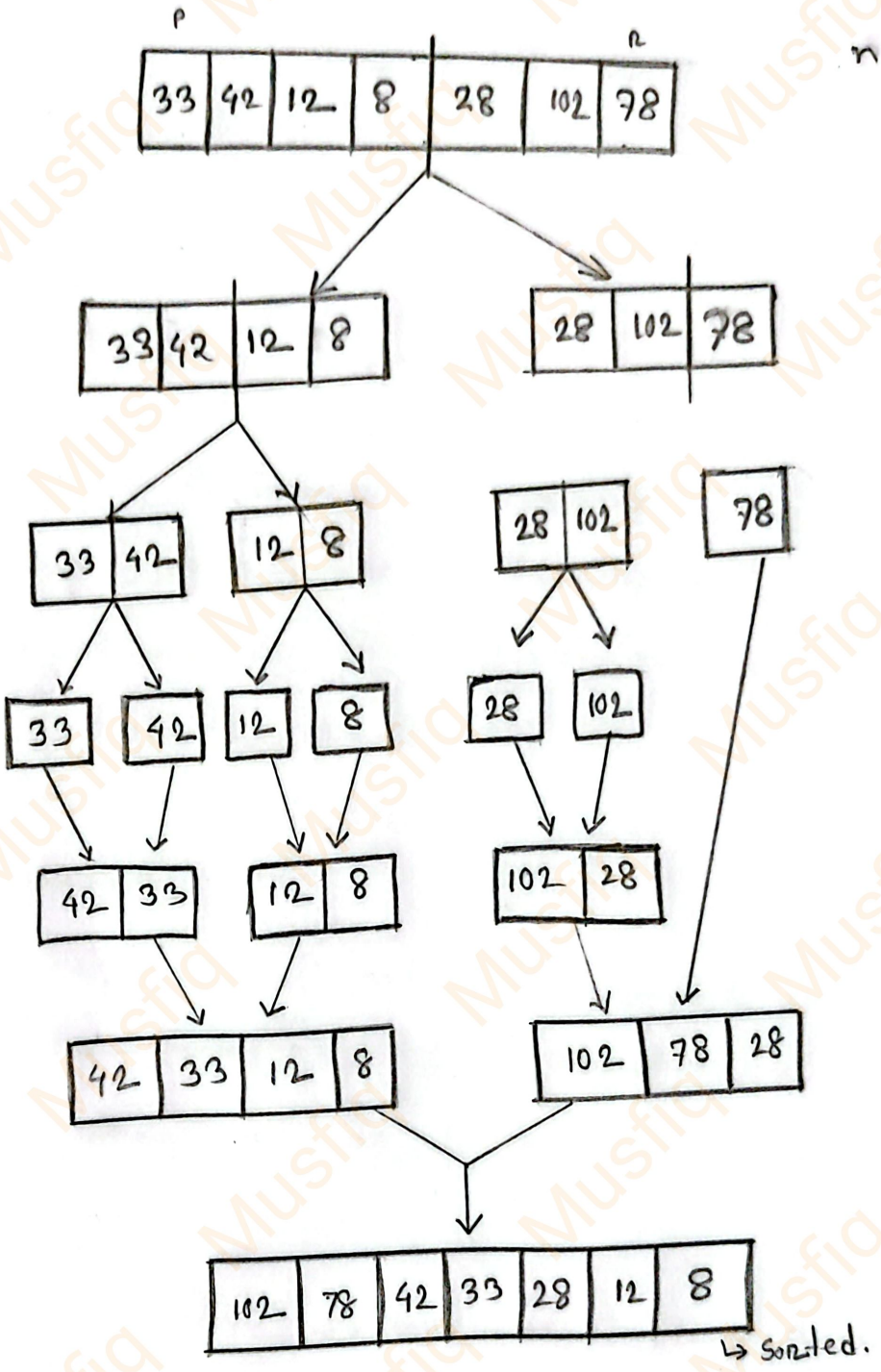


(5)

```
void insertBefore(int elem, int num) {
struct node *newItem = (struct node*) malloc(sizeof(struct node));
newItem->value = 25;
newItem->next = NULL;
if(head == NULL){
    printf("L.List is empty"); return;
}
if(head->value == 30)
{ newItem->next = head;
  head = newItem; return;
}
struct node*current = head;
struct node*prev = NULL;
while(current != NULL && current->value != 30)
{ prev = current;
  current = current->next;
}
if(current == NULL)
{printf("%d : not exist in List", 30);
 free(newItem);
 return;
}
newItem->next = current;
prev->next = NewItem;
}
```

Ans to the ques no-01

* 33, 42, 12, 8, 28, 102, 78



Descending order merge sort recursive tree.

Ans to the ques no-02

Ticket Reservation Timeout:

Let's consider an online train ticketing system where customer can reserve tickets for their disired train journey. The queue follow a FIFO order.

Queue : A → B → C → D.

For this seenario there is a limited time to complate the transaction.

NOW, let's assume customer C encounters a technical issue during payment process, causing their reservation to time out and expire. For, this the system removes Customer C from the queue.

the new queue is now,

Queue : A → B → D.

The queue has drifted because Customer C, is no longer in the queue for time limitation in transaction and D, who came later to C gets served befor customer C.

This is the illustrative example of drifting Queue.

Ans the ques no - 03.

code.

```
sum =0
for(i=1; i<=n; i++)
  { for(j=1; j<=i; j++) {
      sum = sum+i+j;
    }
  }
printf("%d", sum);
```

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

$$= n+1+\frac{n^2}{2}+\frac{n}{2}+n+\frac{n^2}{2}+\frac{n}{2}+1+1$$

$$= an^2+bn+c$$

$$= O(n^2) \text{ Ans.}$$

④ · Last digits → 34 Ans to the ques no – 04

P = 34 + 5 = 39

q = P + 3 = 42

r = P + q = 81

s = r - 1 = 80

t = r + s = 161

ⓐ head -> next -> next -> value = 81 (r)

ⓑ last -> prev -> next -> value = 161 (t)

ⓒ temp -> prev -> prev -> prev = NULL

ⓓ · temp -> next -> prev -> prev -> value = 42 (q)

ⓔ · last -> prev -> prev -> next -> value = 80 (s)

beg del.

```
struct Node -*Current;
 head = head -> next node;
  free(head);
  };
```

/ temp = head.
head = head -> next
free(head)

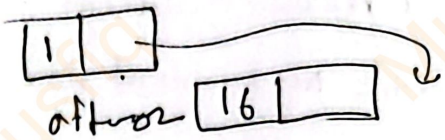
Pendient.

```
Void deleteAny ( int x)
if (head = = NULL)
   { printf ("Sony");
    }
  struct node * curr;
    "    "    * prev;

while(curr != NULL && curr -> value != x);

    { pre = curr
      curr = curr -> nextNod.)
if(curr = = NULL)
 { prtf ("Not F);
}
if ( Prev != NULL);
     prev -> nextNod = curr -> next nod.
 }
 eles
     hed = curr -> nex. nod.e
  .free(curr)
}
```

After (Particular)




```
void add_Af,( ) {
    if (Head == NULL)
    { print ("sorry");
      return;
    }
struct node *NewNode = malloc(struct*) (sizeof (stuct Node));
  newNode -> value = 1;
  newNode -> next = NULL;
Stuct Node* traveller;
   traveller = head
if (traveller != NULL)
  { if (traveller -> value == 16)
    { newNod -> next = traveller -> next;
      traveller -> nextNode = newNode; return;
    }
  traveller = traveller -> nextNode;
}
```

Stacks

Last in, First out (LIFO) → Item remove in reversed.

First in First out (FIFO)
↳ removed in same order.

Push (জায়গা)

Pop (বাদ দেয়া/delete) → always delete from top

Top (last element যে point)

push(10), push(20), push(40), push(70), pop(), push(100)

10 → top	20 → Top	40 → top	70 → top	40 → top	100 → top
	10	20	40	20	40
		10	20	10	20
			10		10

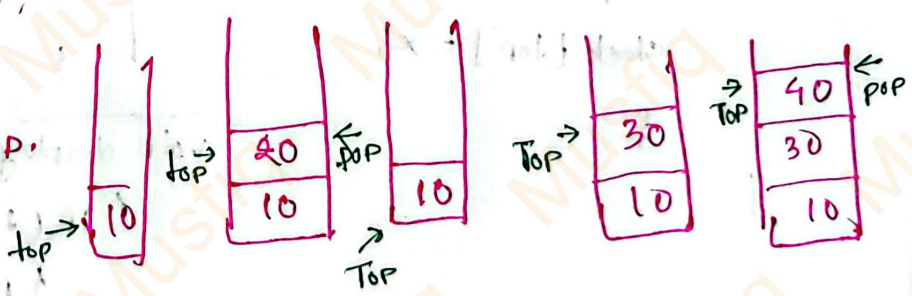
× push(10), push(20), pop(), push(30), push(40), pop(), pop()

pop()

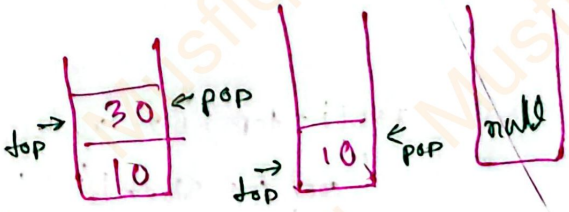
10	20	10	30	40	30	10
	10		10	30	10	
				10		

Underflow

Insert
Delete
} from top.

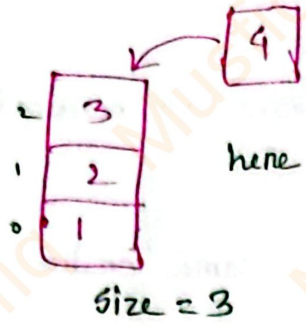


Visual



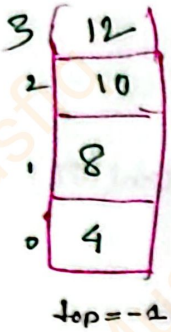
Underflow

Overflow

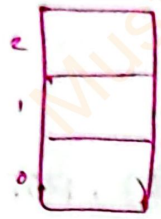


here i want to add 4. that means overflow

size = 3



top = -1



top = -1

↳ global declaration

called pop(). but there is no element here it's called Underflow.

print → 12, 10, 8, 4

top, print → underflow

Push

```
Void Push(int x) {
    if (top == maxStack-1)
    {
        printf ("Overflow");
    }
    else
    {
        cin >> x → 4, → 8, → 10, → 12
        top++; → -1+1, 0+1, 1+1, 2+1
        Stack [top] = x;
    }
}
```

Pop

```
void Pop ( ) {
    int y;
    if (top <= -1)
    {
        printf ("Underflow");
    }
    else
    {
        printf("%d", Stack[top]);
        top--; 3-1, 2-1, 1-1, 0-1
    }
```

```
void display ( )
{   if (top <= -1)
    {   printf ("Empty");
    }
    else
    {   for(i=0; i<=top; i++)
        {   cout << Stack[i];
        }
    }
```

9, 8, 10, 12

```
for(i=top; i>=0, i--)
{   cout << Stack[i];
}
```

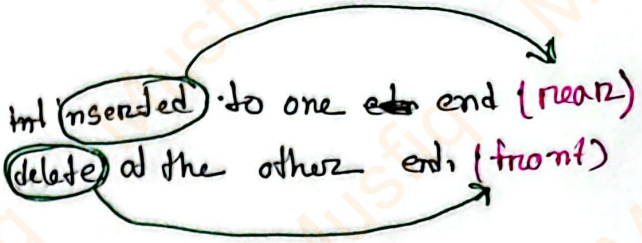
12, 10, 8, 4

Queue

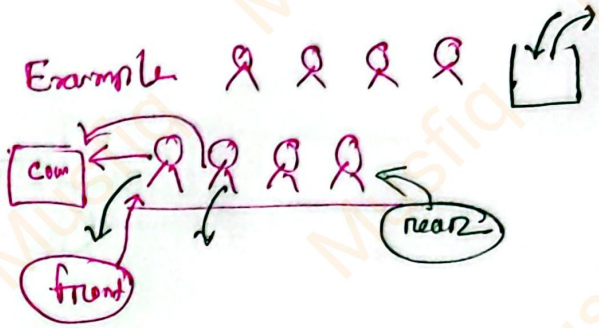
- enqueue (insert)
- dequeue (delete)

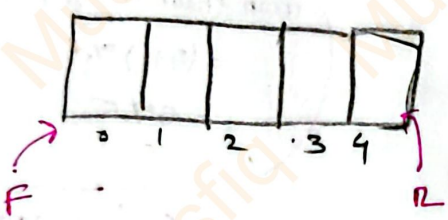
special kind of list, where, item one end (insert) to one end (rear)
delete at the other end (front)

First in, First out (FIFO)

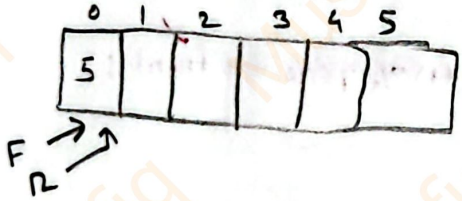


Example

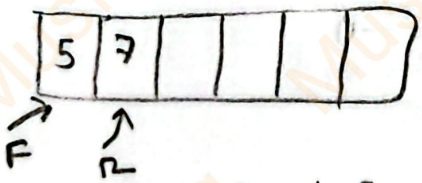




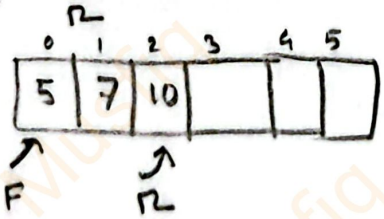
f = rz - 1 (Queue empty)



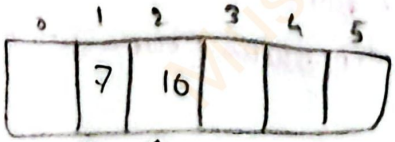
enq(5)



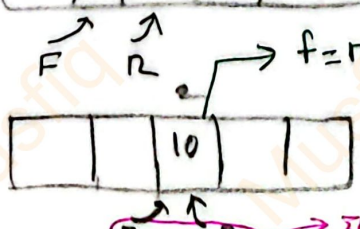
enq(7)



enq(10)



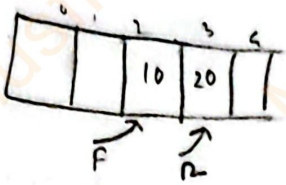
deq()



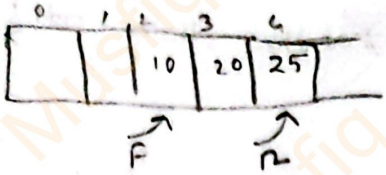
deq()

F R → যখন একটা Index কে point করে থাকে তখন array তে ঐ ele আছে

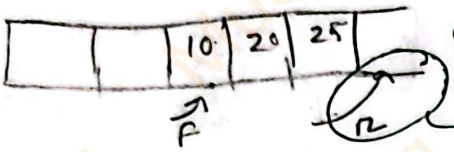
enq(5)
enq(7)
enq(10)
dequeue() → ফাংশন call করার সময় কোন Argument pass করা লাগে না Always 1st element delete হয়ে যায়।
dequeue()



enq(20)



enq(25)



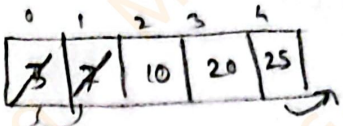
enq(30) → Overflow.

enque(5)
enqueue(7)
enqueue(10)
dequeue()
dequeue()
dequeue()
enqueue(10)
enqueue(25)
enqueue(30)

dequeue() → যদি এমন call করা হয় তাহলে q[2]=10 print হবে কারণ array empty

n=5

f=r=0 এ 20 add হবে



f=r=-1

2x dequeue()

r = r-1

q = q

f=r=-1

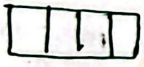
dequeue()
underflow

rear=(rear+1)%qsize
=(4+1)%5
=5%5
=0

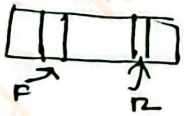
from aadeeb sir

case 1: (overflow) → enq(20)

case 2: (Empty)



case 3: (normal)




```
Void enqueue(int x){          → r = n-1, q = q overflow.
if (rare == n-1)              // (rare+1)%qsize == front;
  {printf("Overflow");
  }
else if (front == -1 && rare == -1)
  { front = rare = 0;         → q[0] = 5
    que[rare] = x
  }
else
  {
    rare++;                   // rare = (rare+1)%qsize;
    queue[rare] = x;          // queue[rare] = x;
  }
```

q[1] = 7
q[2] = 10
q[3] = 20
q[4] = 25

Dequeue

```
void dequeue() {
    int y;
    if (front == -1 && rear == -1)
    { printf("Empty");
    }
    else if (front == rear)
    { printf("%d", queue[front]);
      front = rear = -1;
    }
    else
    {
      front++ // front = (front+1) % qsize;
    }
}
void desplay() {
    int i;
    if (front==-1 && rear == -1)
    { printf("Empty");
    }
    else {
        for(i= front ; i <= rear ; i++)
        { printf("%d", queue[i])
        }
    }
}
```

q[0] = 5
q[1] = 7