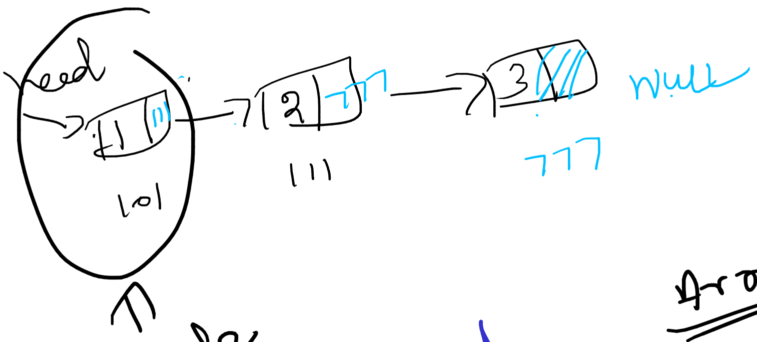
# Linked List

DS

Suitable container



node

int a=10;

double d= 17.5;

char s= 'a';

Array

Homogenous

Predefine,

class node{

int value;

Node next

};

1) value

2) Address of next element

"newton"

int [] ={10, 20}

char [] = {'N','o','t'}

## array

b.add → ar[3]

access → O(1)

5

[1, 2, 3]

7

[1, 2, ~~3~~]

Dynamic element ✗

insert at middle O(n)

deletion O(n)

Search → (log n)

## LL

1) head → 3 times

→ O(log n)

Dynamic elements

O(position)

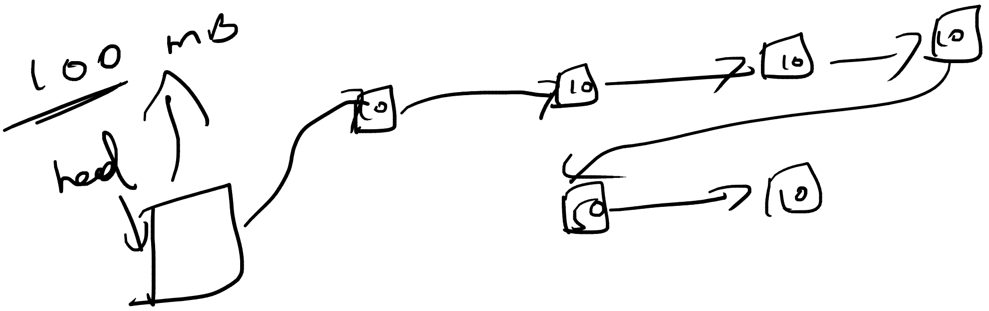
O(1)

O(position)

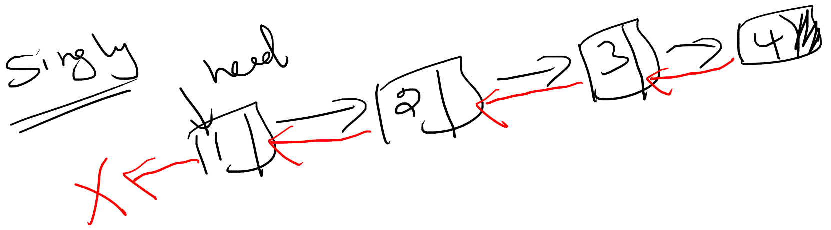
O(n)

Tree   DLL → explain TC

Memory   Linked List



DLL



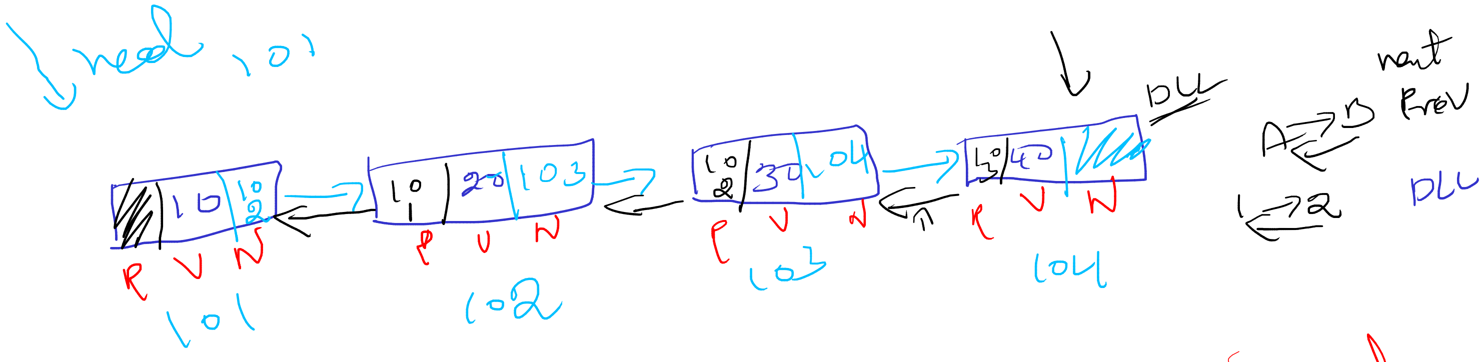
1) Reverse

SLL   next

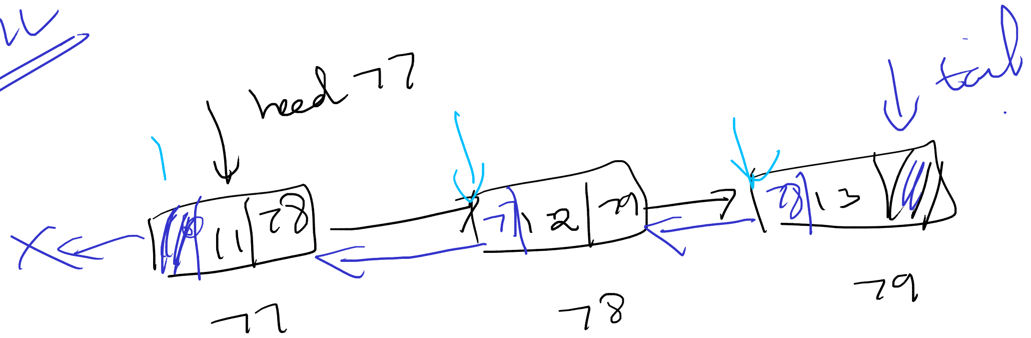
A→B   singly

1→2   LL

tail



DLL



{ int val

Node prev

Node next

3

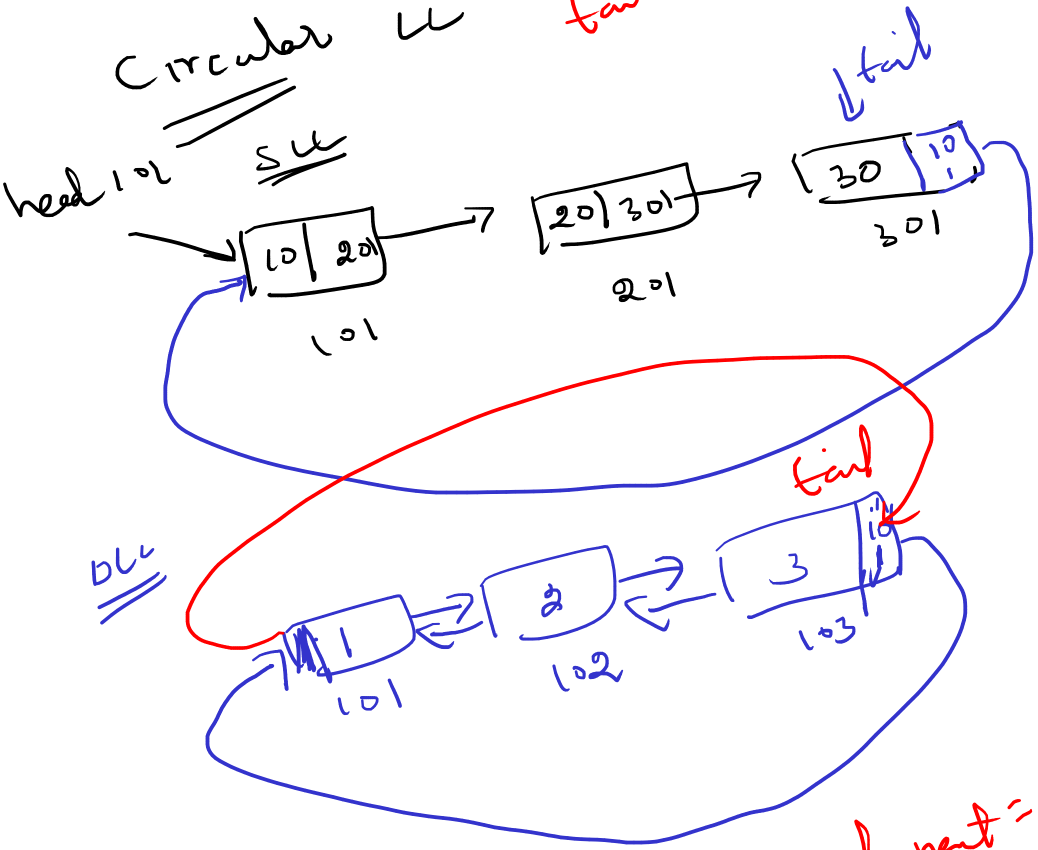
Node next ] 2

Node cre

prefer

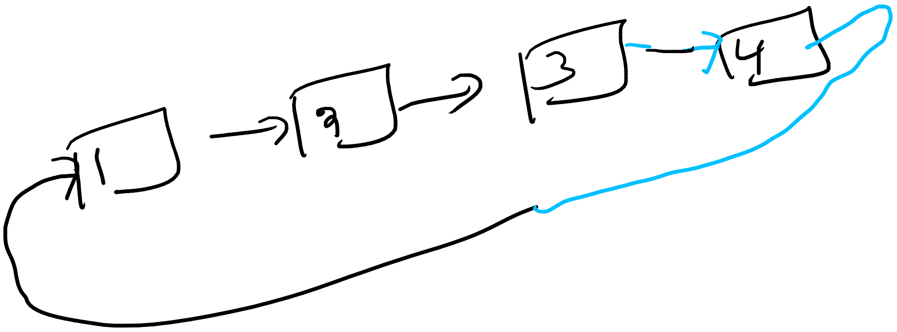
# Circular LL

tail.next = head



tail.next = head

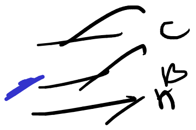
head.prev = tail



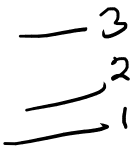
# Stack

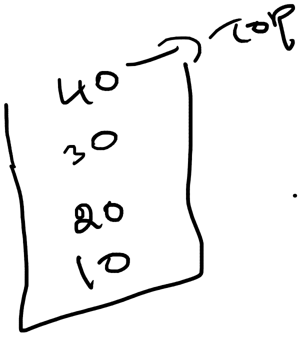
LIFO FILO

Last -> First



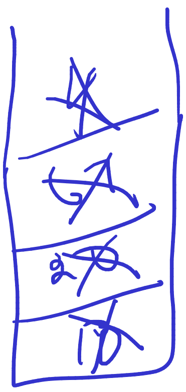
Last -> First

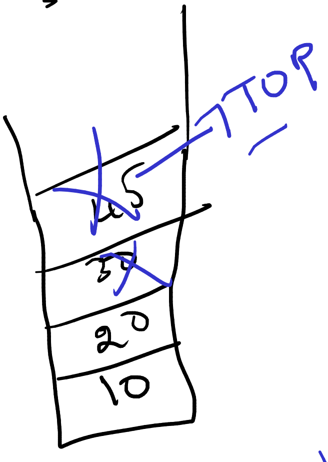




or top el

10 20 30 45 67 4

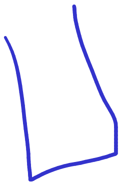




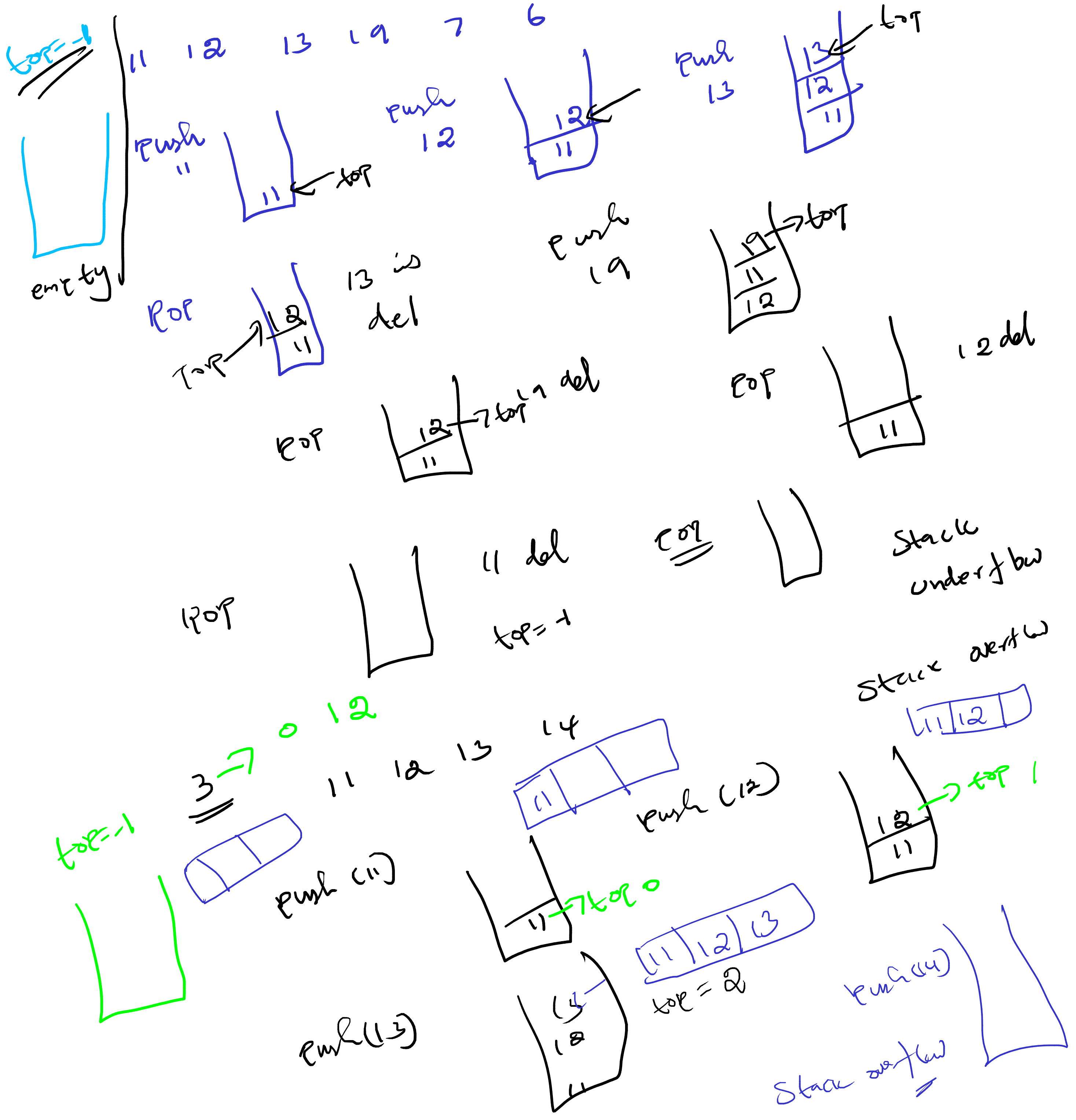
insert -> push

delete -> pop

pop -> 4 X



top -> last element stack



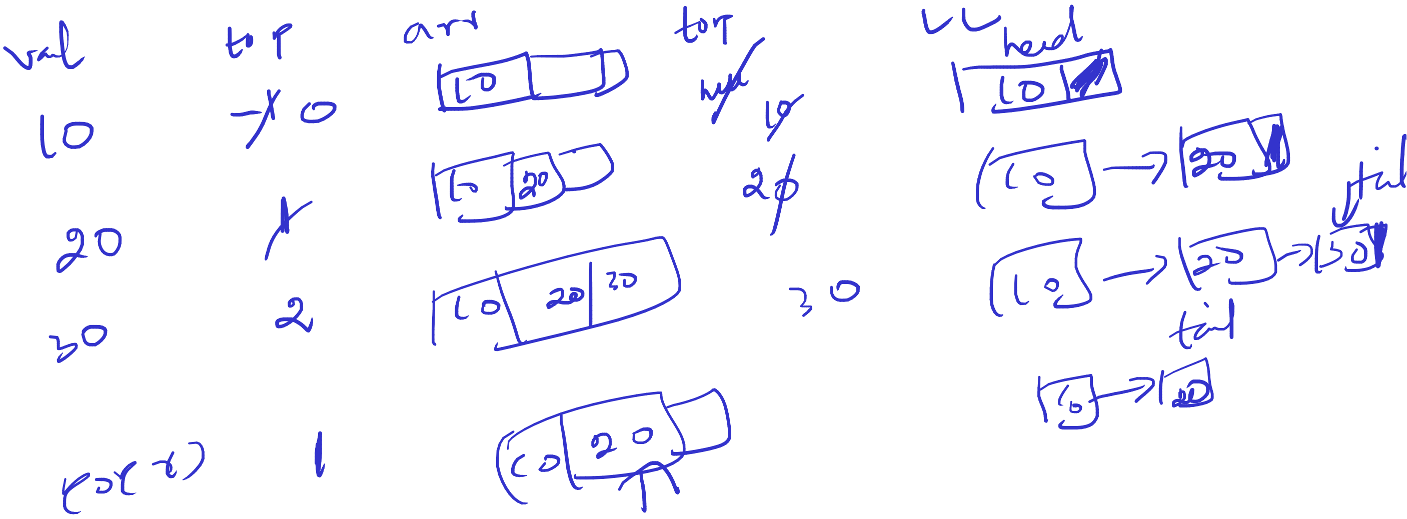
Peek | top

current element || last element

13

arr → stack

LL → stack


val
top
arr
top
LL
head
10
-1 0
10
~~null~~
10
10
20
1
10 20
20
10
20
30
2
10 20 20
30
10
20
30
tail
tail
pop()
1
10 20
10
20

1) Datatype

2) Integer overflow

3) operator

4) conditions

5) loops

6) return from fn

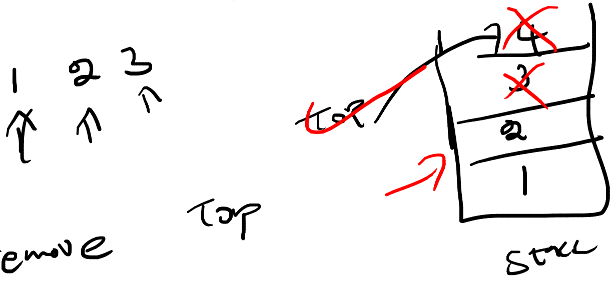
1) swap    3) reverse a number

2) a?b

# Stack

Linear DS

LIFO | FILO



remove Top

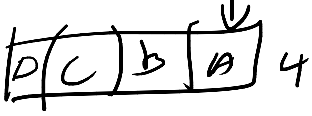
peek/ top() ->
return top

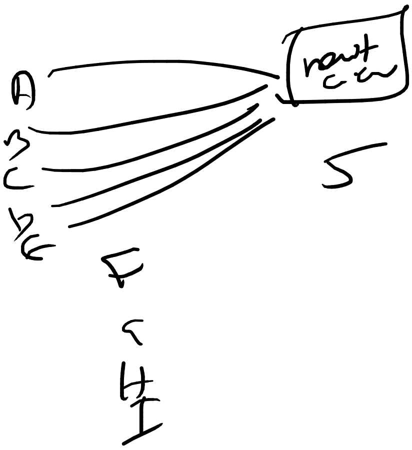
pop()

FILO -> Stack
FIFO -> Queue

# Queue

2001
Flipkart
3
2
1
100



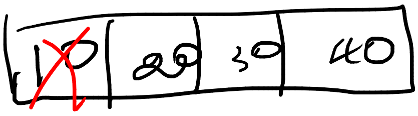


enqueue → insert

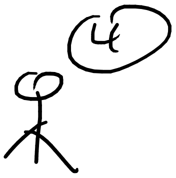
dequeue → delete / exit

front / peek()

10 20 30



Stack → top = -1



front = 0

rear = 0

full → (r >= s2)

deq() 10

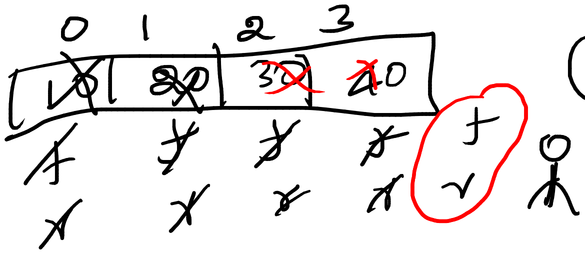
d → 20

d → 30

d → 40

d → ? empty

arr



(4)

50

enque(10)

enq(20)

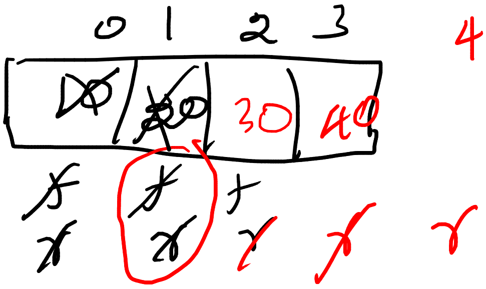
enq(30)

40

d → 10

d → () (f == r)

empty



4

(4)

(rear >= size)

(4 >= 4)

d()

(f == r) X

else

head

→ front → exit / de

rear → insert / enq

tail

e → 10

e → 20

e → 30

e → 40

e - 50

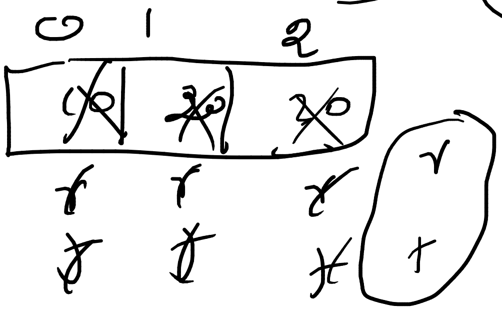
r → for → insert

f → for → delete

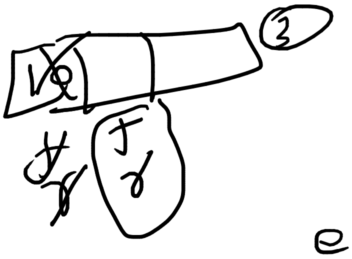
cur

(r >= capct) full

full



(3)



e → 10

d → 10

empty

e → 10

e → 20

d → 10

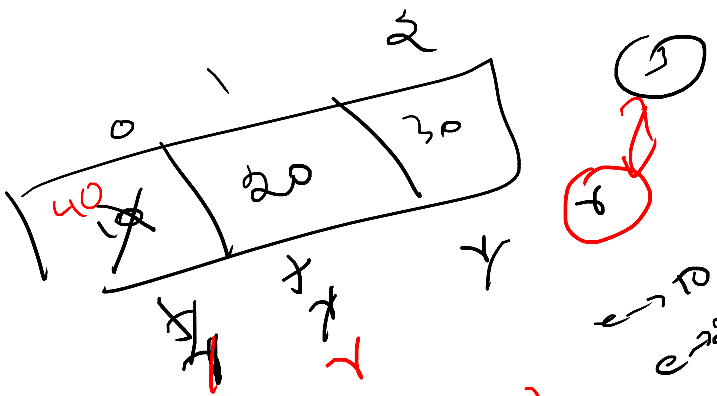
e → 30

d -

occupy front

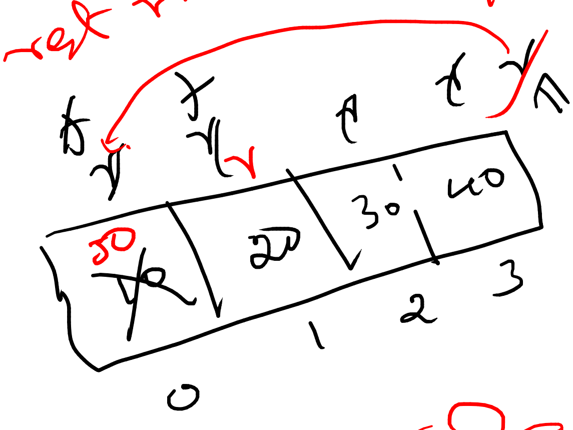
size

deq



e->10
e->20
e->30
e->40

rear r=0



56->

size

3
4

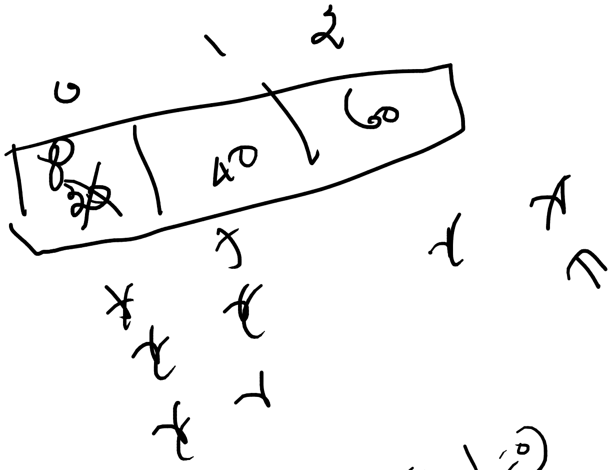
e(10)
e(20)
d()
e->30
e->40
e->50

r=-1
f=0

4 => ae

0 => empty

size++
size--;



3

capacity=3

e->20
e->40
d->
e->50
e->80

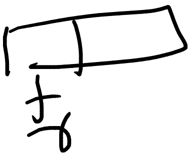
front=0
rear=0
size=0

(f!=0)

full -> size==capacity
empty size==0

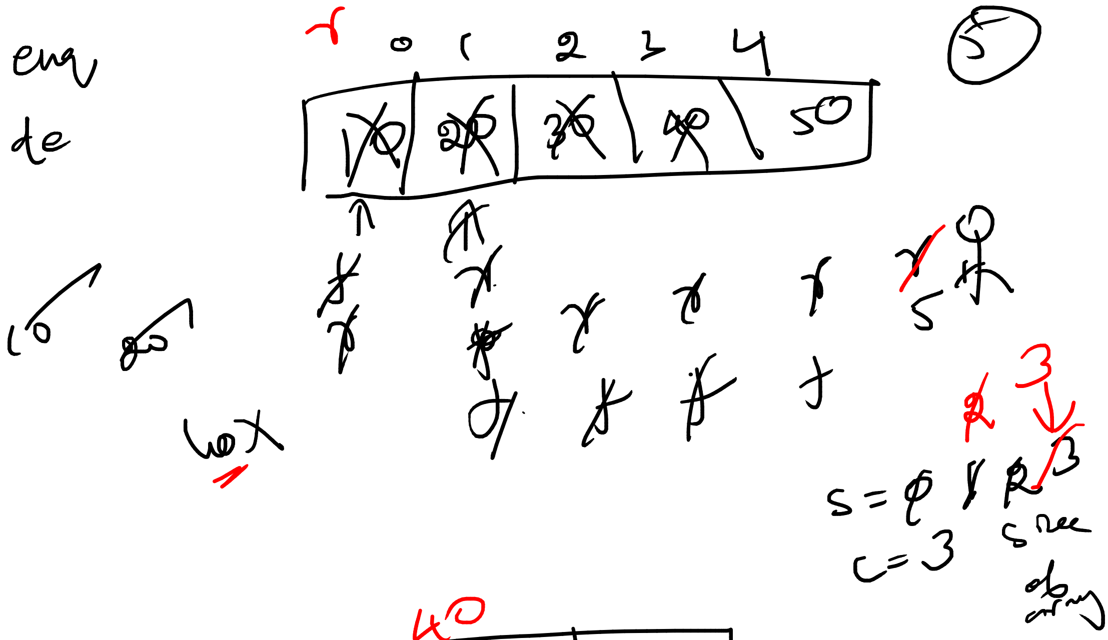
## Que

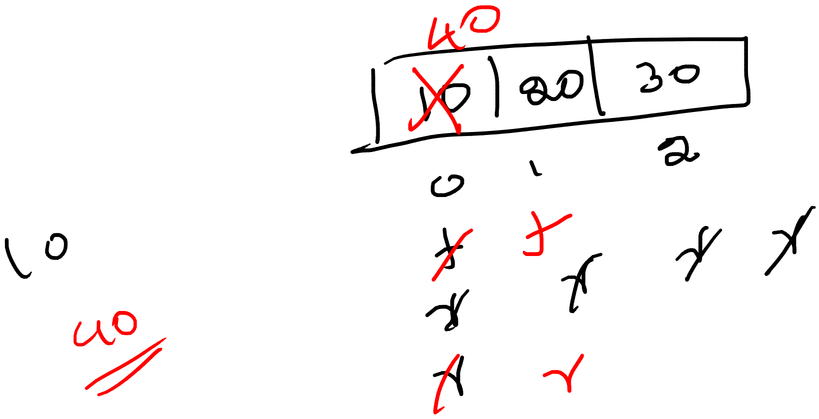
FIFO



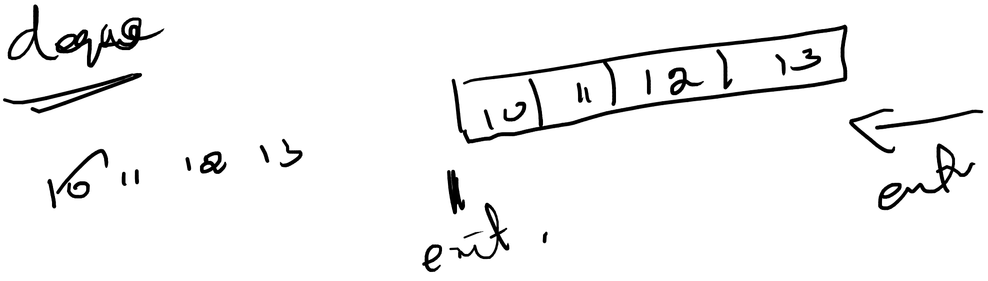
Simple que

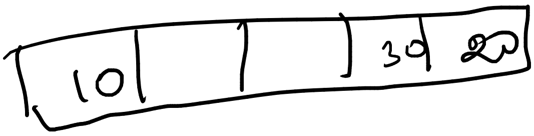
ins → enq
del → de



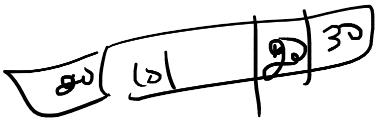


## deque





addfront (10)
add end (20)
add end (30)



Que

insert end

remove beg

peek()

Dequeue

add First

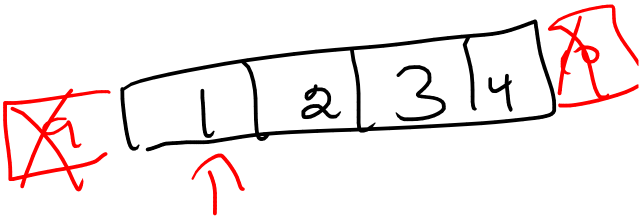
add Last

remove First

remove Last

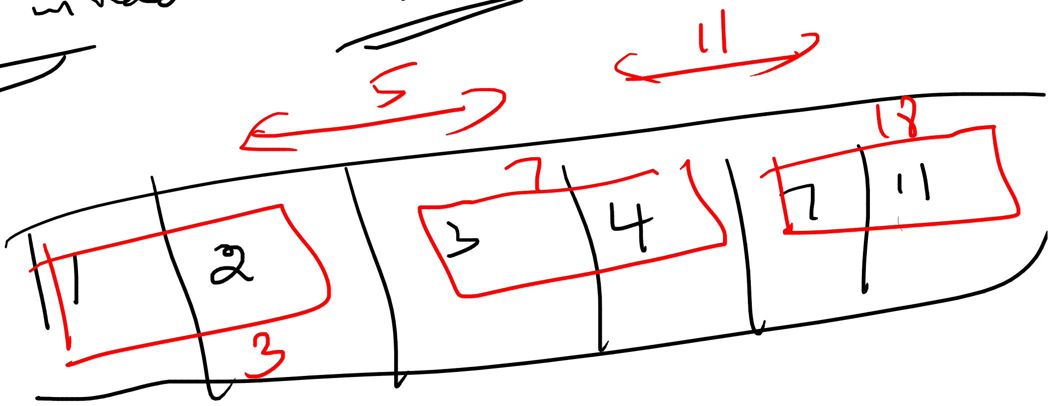
set First

set Last



Sliding window

k=2



Subarray size k maximum sum

(1 2)

2 3

3 4

4 7

7 11 -> 18

length of sum

| -7 | 6 | 13 | 9 | 14 | 2 |
|---|---|---|---|---|---|

k=2   23

(-7 6) =-1   13 9 =22   14 2 =16

(6 13) =19   9 14 = 23

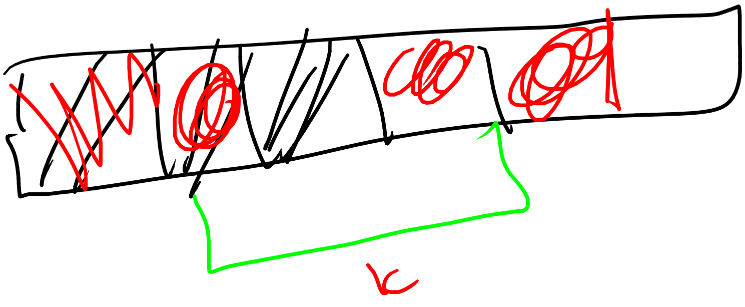
3

-7 6 13 → 12

6 13 9 → 28

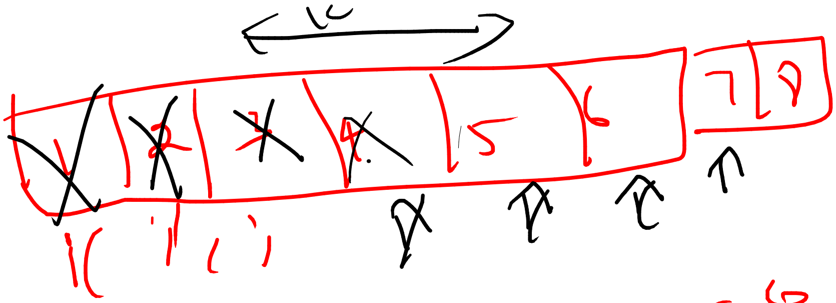
13 9 14 → 36

9 14 2 → 25



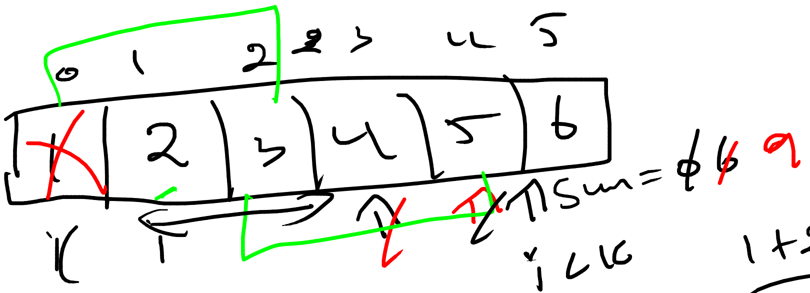
rejection

k=3



1+2+3
2+3+4
3+4+5

1+2+3 = 6
6-1+4 = 9
9-2+5=12
12-3+6= 15
15-4+7= 18



1+2+3 =6

i=k → n
i-k
i

k=3

n*k

1+2+3
2+3+4

3-3=0

6-1=5+4=9 remove
add

12-3=9
9+6=15

9-2=7+5=12

## SLIDING

approach → nested / $n^2$ → linear $O(n)$

window size | subarray size of k

|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|---|
| arr | 1 | 3 | 5 | 4 | 7 | 9 | 3 | 11 | 4 |

k=3

of k ith maximum sum

su

k=3

sum 6 1 2

1) $1+\boxed{3+5} = 9$

2) $9-1+4=12$

3) $12-3+7=16$

4) $16-5+9=20$

$20-4+3=19$

$19-7+11=23$

$23-9+4=18$

1) remove i-k

2) Add i

$3-3=0$

$4-3=1$

$5-3=2$

n*k

O(n)

n k

5 3

arr 1 2 3 4 5

$1/1$ $\frac{1+2}{2}$ $\frac{1+2+3}{3}$ $\frac{2+3+4}{3}$

$\frac{3+4+5}{3}$

>k

$(1-k)$ .. $\frac{(1-2)+(1-1)+(4)}{k}$

n=5 k=3

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 1 | 3/2 | 6/3 | 9/3 | 12/3 |
|  |  | 2 | 3 | 4 |
| 1 | 1 |  |  |  |

n=5 k=3

s= 01010

(0(2)>=(0(1)

|  |  | cont zero | cont Ones |  |
|---|---|---|---|---|
| 0-2 | 010 | 2 | 1 | (2>=1) |
| 1-3 | 101 | 1 | 2 | (1>=2) ✗ ✗ |
| 2-4 | 010 | 2 | 1 | (2>=1) |

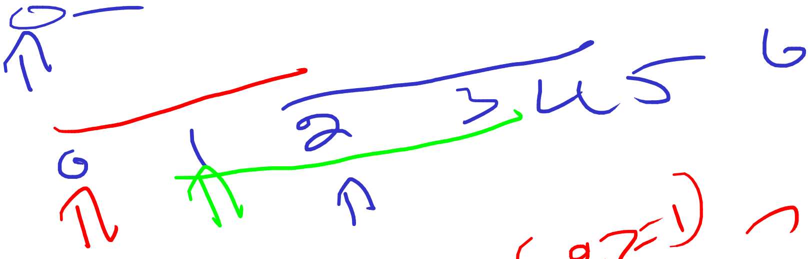
2 2

0 1 0 0 0 1 1 0   2 5   one 3   (5>=3) ->

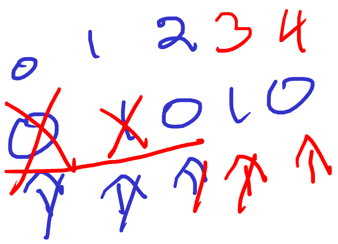
generet subarray of size k

k=3

(n≥k)   O(n)



(2>=1) -> ans 0 1 2



c2 - 0 1 2 1 2

c0 - 0 1 2 1

(1>=2) X

if (c2>=c0)
cout

k=3   4-3=1

k=1   k=3

2>=1 ->

ans=0 1 2

0 1 2 3 4
0 1 0 1 0

c2 = 0 1 2 1 2

c0 = 0 1 2 1

(i>=k)  (4>=3)

i-k   4-3=1

(4>=2)

0 -> 48

'0'-'0'=0   (2>=1) ->

(i>=k-1)

(c2>=c0)
ans++

(1>=2) X

subarray length

ω

1+3+5

1 3 5 → 9

3 5 4 → 12  3+5+4

5 4 7 → 16  5+4+7

4 7 9 → 20  4+7+9

7 9 3 → 19

9 3 11

3 11 4

repetion

k=3

48 - 48

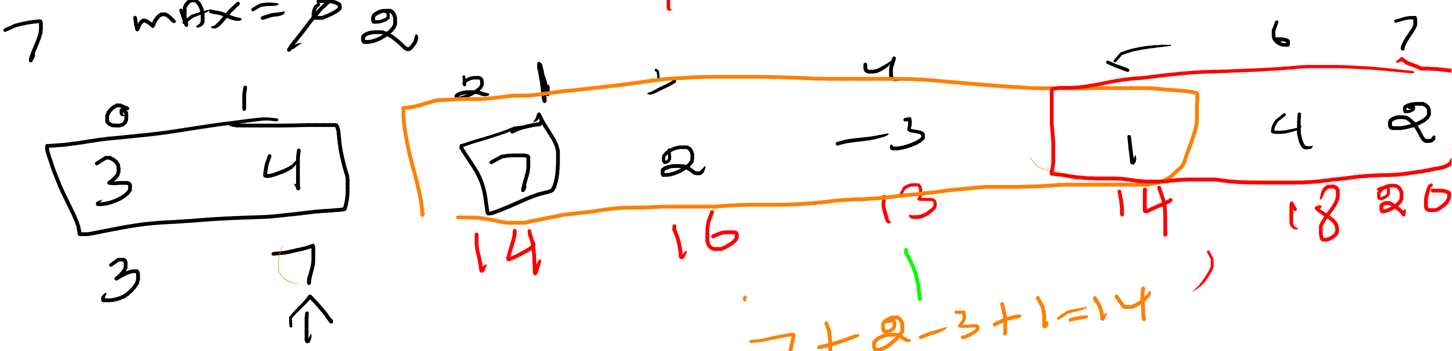
0 - 0 = 0

1 - 0 = 1

49 - 48

| | |
|---|---|
| 0 | 48 |
| 1 | 49 |
| 2 | 50 |
| 3 | 51 |
| 4 | 52 |
| 5 | |
| 6 | |
| 7 | |

k=7  max = ~~0~~ 2



7+2-3+1=14

14-7=7

18-7=11

20-7=13

sum-k

14-7 = 7

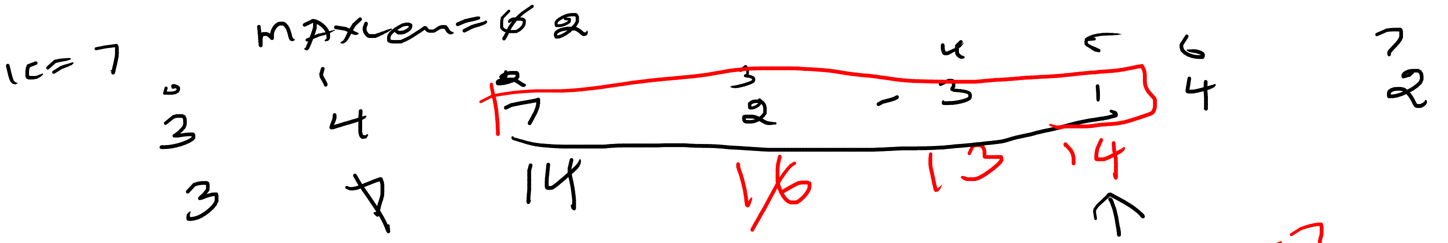
2-1=1

16-7=9

13-7=6

14-7=7

5-1=4



| | |
|---|---|
| ~~13, 4~~ | |
| 16, 3 | |
| 14, 2 | |
| 7, 1 | |
| 3, 0 | |

14-7=7.

Sum-k =

7+2-3+1=7

5-~~1~~=~~2~~ 4

14-7=7

Sum-k

2-1=1

14-7=7

16-7=9

Sum=0)

max < curr

(2<1) X

| | |
|---|---|
| 13 | 4 |
| 16 | 3 |
| 14 | 2 |
| 7 | 1 |
| 3 | 0 |

7-4=3

1) sum == k

maxLen = i+1

if (hm.contains(s-k))

len = hm.get(7)

5-1=4

K=7    n=8

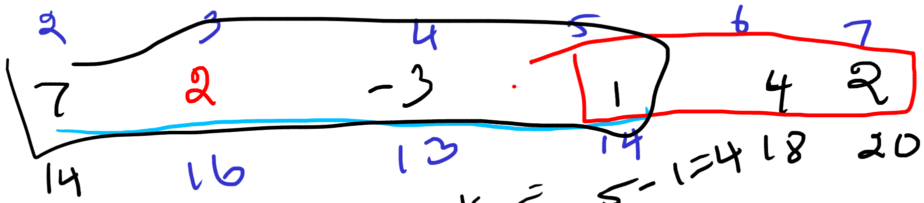
0 1

arr 3 4

3 7

20-7=13

13



sum-K = 5-1=4

14-7=7

14-7=7

14-7=7

3x3

j 0 1 2

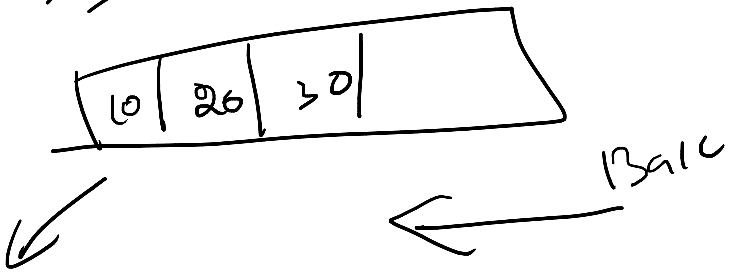
i 0 0+0%2 0+1%2 0+2%2

1 1+0%2 1+1%2 1+2%2

2 2+0%2 2+1%2 2+2%2

0
1 1
0 0 0
1 1
0

FIFO

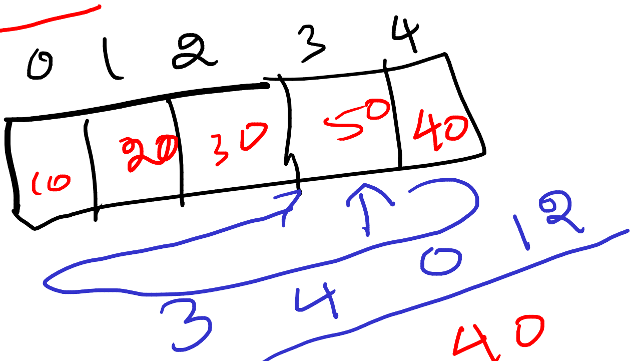


Double endue

insert→end, begin

remove→begin, end

50 40 10 20 30



end 10

end 20

end 30

beg 40

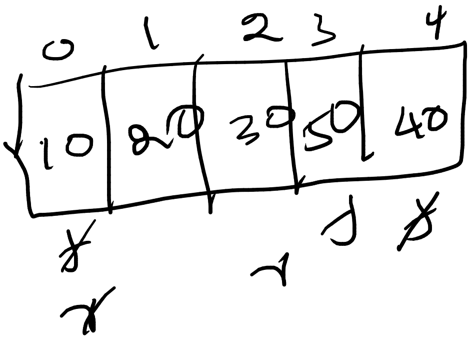
beg 50

50→40→10→20→30

-1

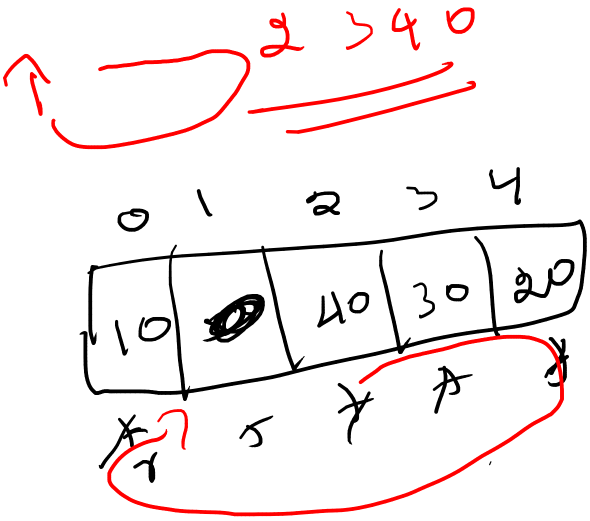
f→beg

f=size-1



e→10
e→20
e→30

b→40
b→50



50 40 30 20 10

-1

b→10
b→20
b→30

b→40
b→50

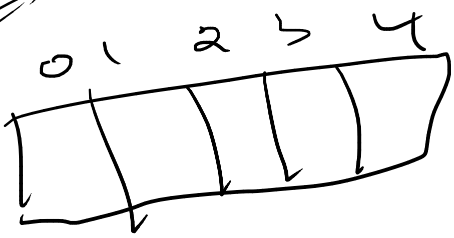
60

f--

(f-1==-1) X

$n^2$

$O(n)$



④

1) → 7

2) → 3,4

4) 7 2 -3 1

3 → 1 4 2

④