Date
22/12/22

# § Data Structures §

Last Class :: Complexity notations.

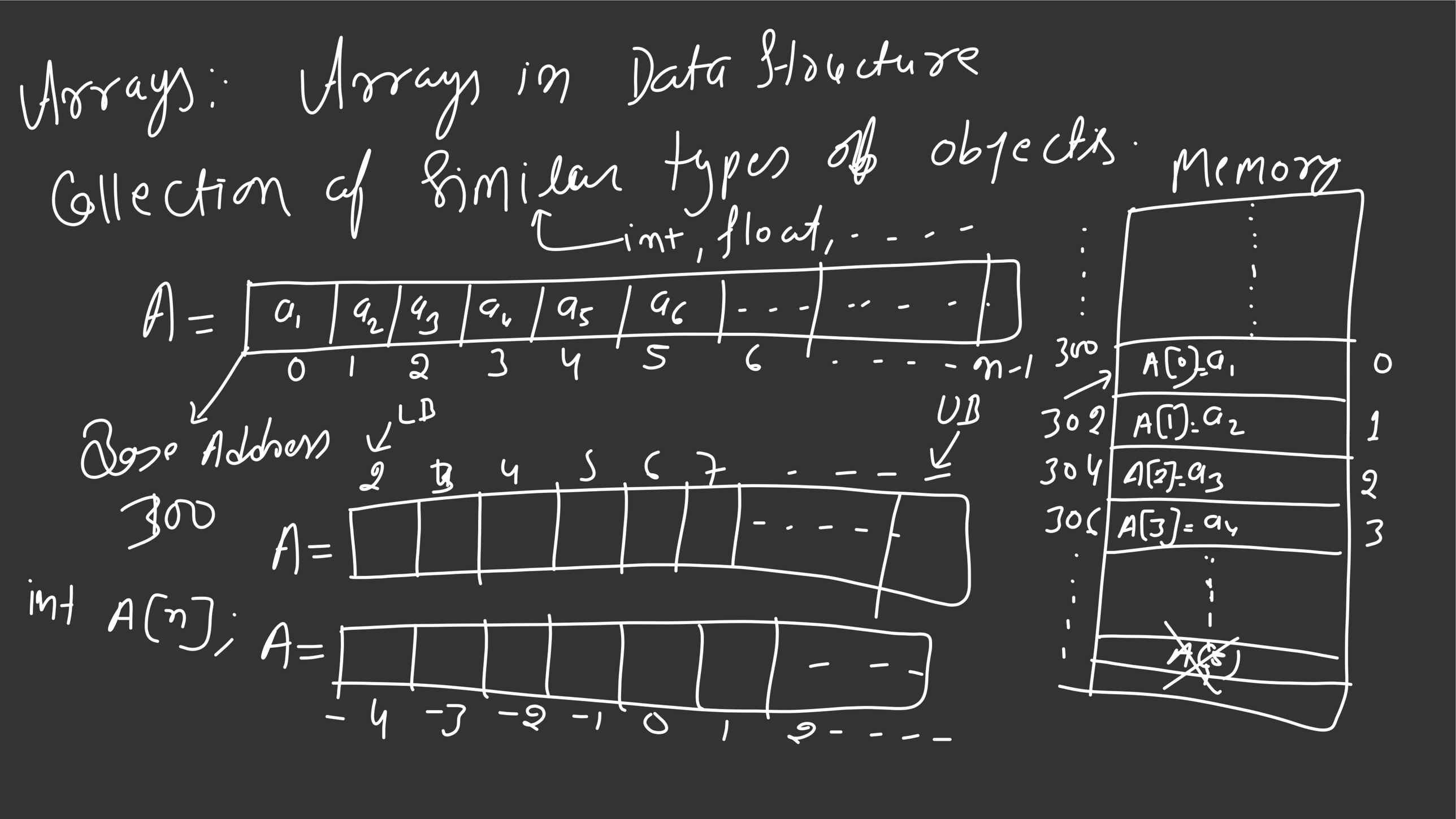
i) Big oh (O) notation → Worst Case

ii) Big omega (Ω) notation → Best Case

iii) Big theta (Θ) notation → Average Case

# Arrays: Arrays in Data Structure

Collection of similar types of objects.



Array in DS :-

Name [LB : UB]

Eg: A [1 : 10] ⇒ UB − LB + 1

10 − 1 + 1 = 10

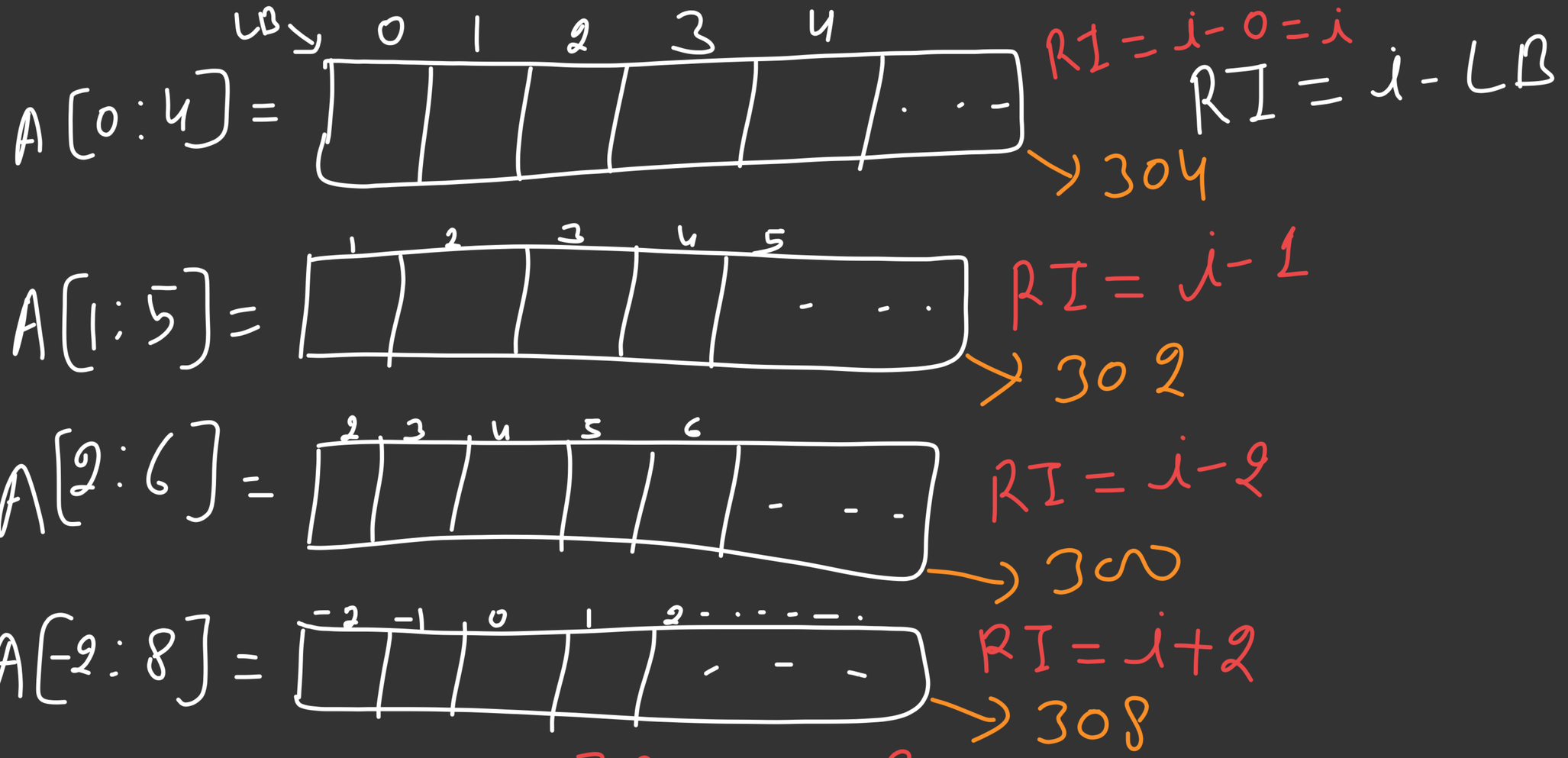
A [2 : 15] ✓

A [−2 : 10] ⇒ 10 − (−2) + 1 = 13

$$Loc(A[i]) = \text{Base address} + W \times (\text{No. of Elements before } A[i])$$

(No. of Elements before A[i]) — Relative Index

W — Size of Each Element.




$$loc(A[i]) = \text{Base address} + w * (i - LB)$$

$$300 \quad 2$$

location of $A[2]$ = ?

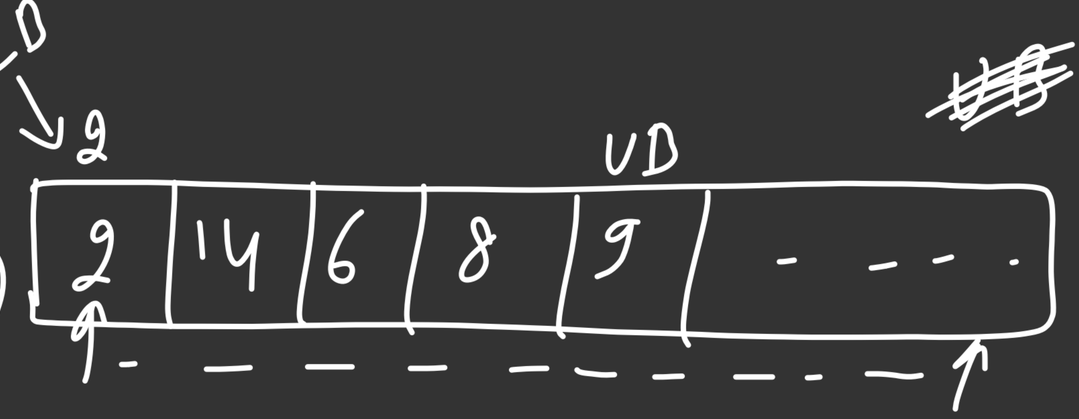
Consider an array A[-5:25], which is stored in an array starting from location 1000. Each element takes 4 bytes in memory. Find the location of A[5]?

1000

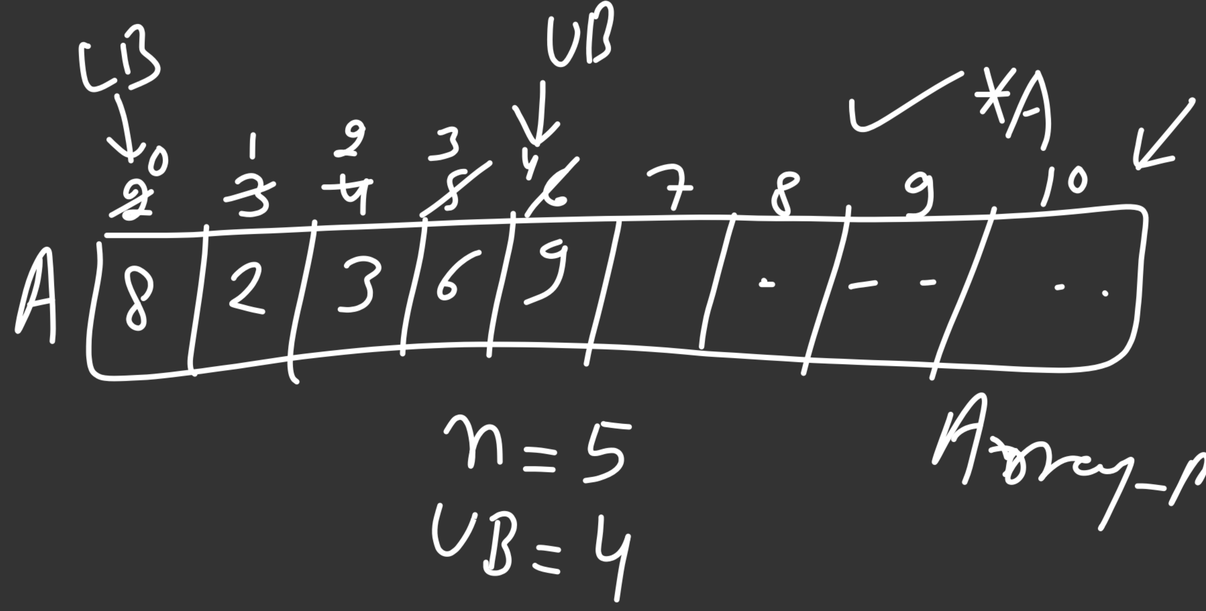
$i = 5$, $LB = -5$, $BA = 1000$

§ Traversing in array:

```
for(i=LB; i<=UB; i++)
{
    pf("%d", A[i]);
}
```




function f(A[], ...




n = 5

UB = 4

Array-Mem

# § Insertion in Array ::

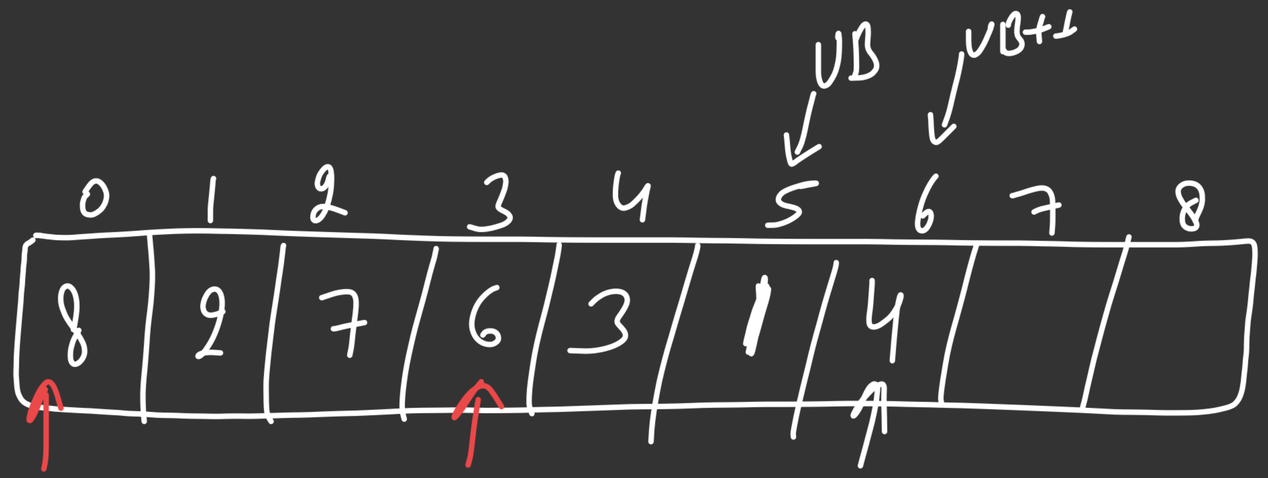
int Insert(A, n, Key)

Insert an element in last

1. if(UB == Array_max)
   overflow & Return
2. UB++
3. A[UB] = Key
4. n++

~~O(n)~~

O(1)




n = 6 = n++

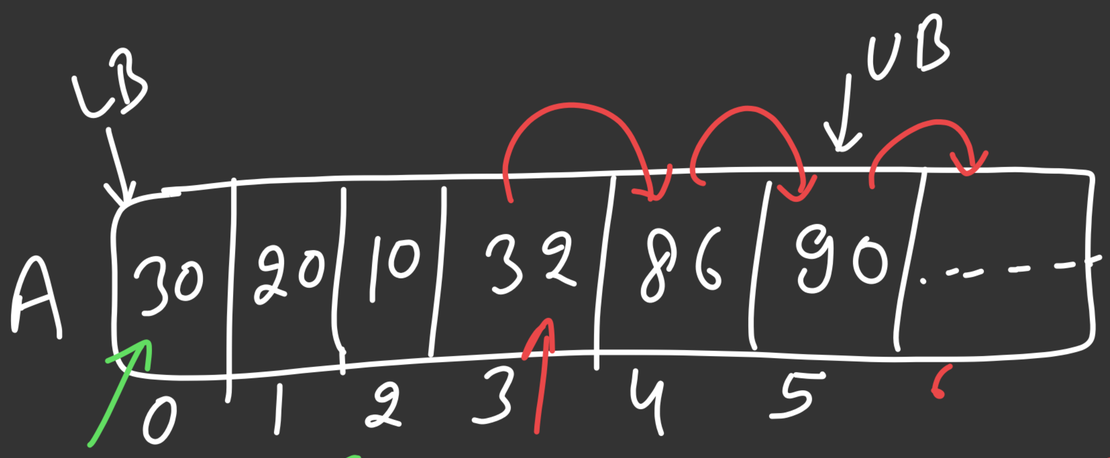
LB = 0

UB = 5 = UB++

Array_max = 8

key = 4

```
Insert(A, LB, UB, n, i, Key)
{
    if (UB == Array-max)
    {
        for(K = UB; K >= i; K--)
            A[K+1] = A[K]
    }
    A[i] = Key;
    UB++;
    n++;
}
```




O(n)

Key = 25

A[6] = A[5]

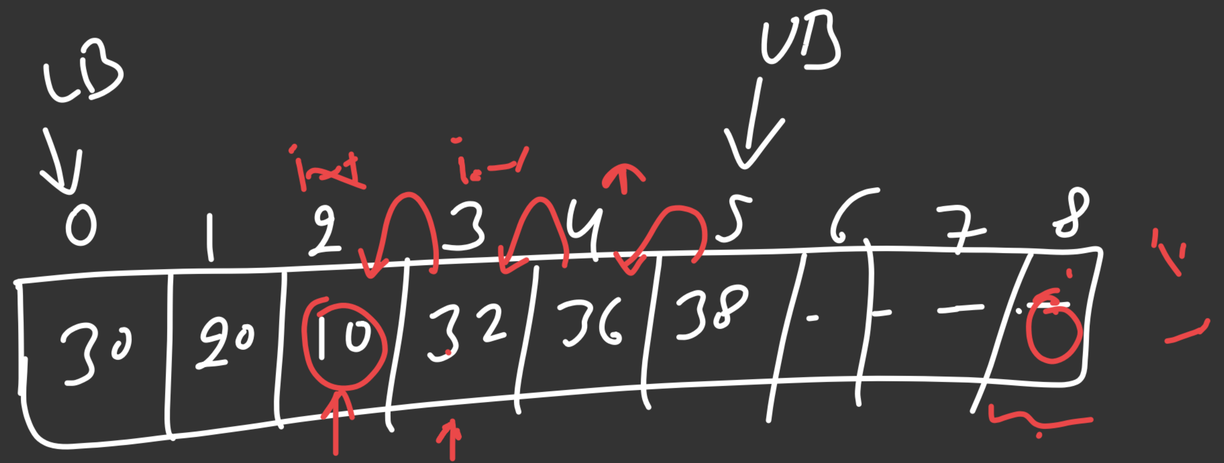
A[5] = A[4]

A[4] = A[3]

A[3] = key

# § Deletion in Array :-

```
temp=0
for(int i=3 ; i<arr.len ;i++ ){
    arr[i-1]= arr[i] ;
    if (i==arr.len-1){
        arr[i]=0
    }
}
```




LB = 0
UB = 5
n = 6
UB--
n--

O(1)

LB ≤ i ≤ UB

Delete.

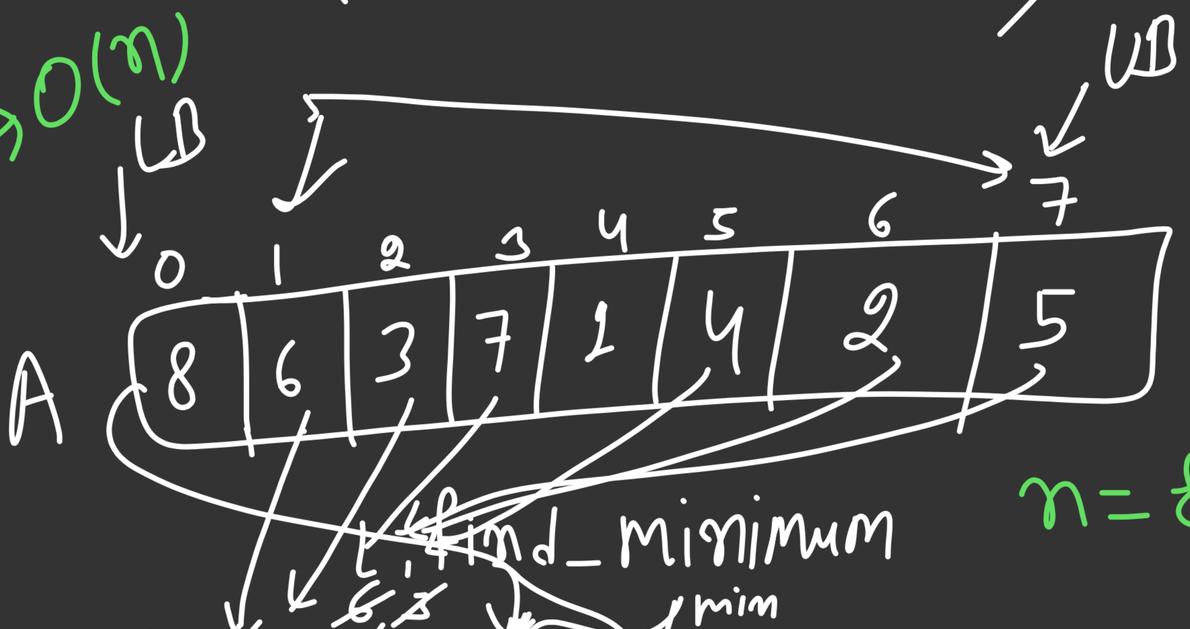
# Applications of Array::

## find a minimum and maximum elements in an array

```
find_min(A)
{
    min = A[LB]
    for (i=LB+1; i≤UB; i++)
    {
        if(A[i] < min)
            min = A[i]
    }
    return min;
}
```

O(n)

O(n)



Another way to find Min & max

A: 2, 8, 3, 7, 5, 1, 4, 3

B: 2 3 1 3

A: 2 1

B: 1

Extra Space.

Best Sorting

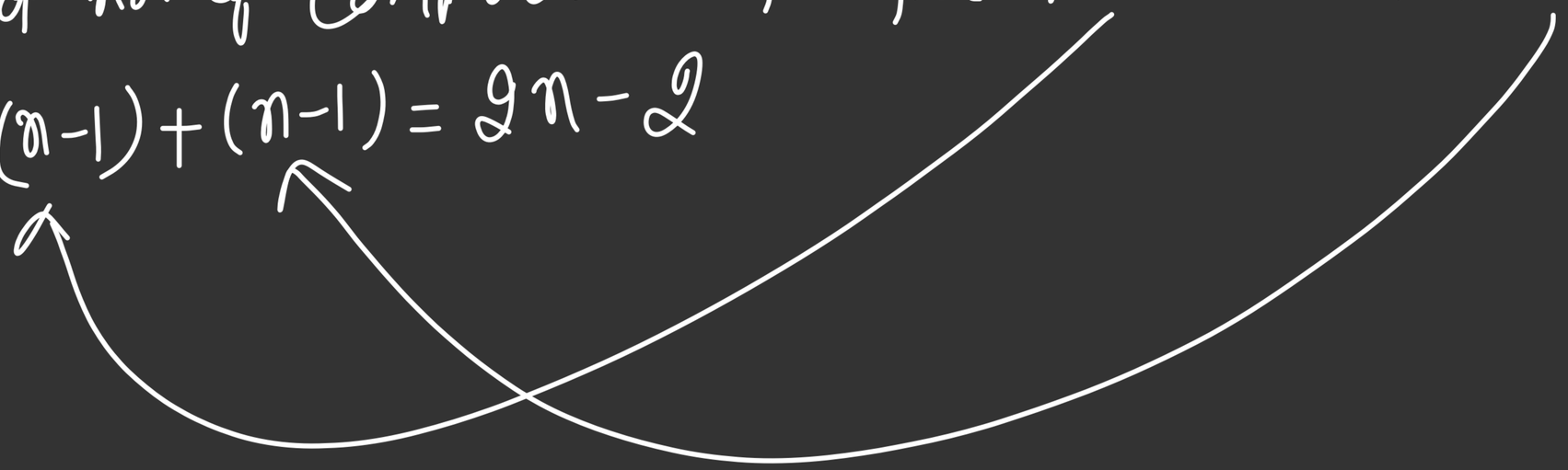
$O(n \lg n)$ ✗

$(n-1)$

$n + \lg n + O(1)$

Exact no. of Comparisons to find minimum & maximum ?

$(n-1)+(n-1)=2n-2$



Can we Improve the above Complexity / no. of Comparisons.

```
Small = A[0]
{
  for(i=1; i<n; i=i+2)
  {
    ① if(A[i] ≤ A[i+1])
      {
        ① if(A[i] < Small)
            Small = A[i]
        ① if(A[i+1] > large)
            large = A[i+1]
      }
    Else
    {
```

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|---|
| A = | 22 | 8 | 40 | 24 | 12 | 18 | 2 | 32 | 25 |

n = 9

Small = ~~22~~ 8

large = ~~22~~ 40

| A[i] | A[i+1] |
|---|---|
| 8 | 40 |

8 ≤ 40

| | |
|---|---|
| 24 | 12 |

24 ≤ 12

| | |
|---|---|
| 18 | 2 |

18 < 2

| | |
|---|---|
| 32 | 25 |

32 < 25

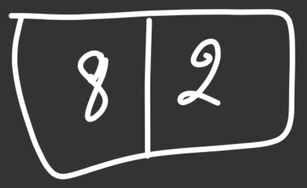
19

2n - 2

2*9 - 2 = 16

$\frac{n}{2}$

3 Comparisons.




$(n-1)$ or $(n-2)$

When $n$ is Even

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

$$= \frac{3n}{2} - \frac{3*2}{2} + 1 \Rightarrow \frac{3n}{2} - 2$$