In [ ]:

```
# 7+ yrs. IT. 4 C++, JAVA, Sen Backend (go, py, c++, js)
# 3 Prod companies.
# 2 Linkedin
#
```

Min, Max

C++: min(10,20), max(1.4,5.6)
Java: min(10,20), max(1.4,5.6)
JS:

In [ ]:

```
t
```

Array Data type

1. Why
2. Creating an array

```
// C++
int arr[10];
for (int i =0; i < 10; i++)
   arr[i] = i*100;


int arr[] = {10,20,30};

int *arr;
arr = new int(10);


delete []arr;


// Java
int arr[] = new int[n];
```

```
# Python
arr = []
```

1. Accessing Data
2. Memory
3. Language wise variations
4. Dynamic arrays

In [ ]:

```
```

Insertion, Deletion in array

In [ ]:
```
a [ , , , , ]
   0 1 2 3 4
curr=-1


add 10
curr=0
a [10,   ,   ,   ]
    0 1  2  3  4

add at end  20,30,40,50
curr=4
a [10, 20, 30, 40 ,50 ]
    0  1   2   3   4
O(1)

Delete last element
a [10, 20, 30, 40 ,  ]
    0  1   2   3   4
curr=3
O(1)


Delete first element
a [20, 30, 40 ,  ,  ]
    0  1   2   3   4
curr=2
O(N)

Add 10 in beginning
a [10, 20, 30, 40 ,  ]
    0  1   2   3   4
curr=3
O(N)

Update value at index  2 to 300
a [10, 20, 300, 40 ,  ]
    0  1   2   3   4
curr=3
O(1)


Update value 40 to  value of 400
a [10, 20, 300, 400 ,  ]
    0  1   2    3   4
curr=3
O(N)


Delete value 400
a [10, 20, 300, 400 ,  ]
    0  1   2    3    4
curr=3
find 400 -> O(N)
delete 400(index 3) -> O(N)

Add ->  O(N) + O(N) = 2O(N) ~ O(N)
Multiply -> O(N) * O(N) = O(N^2)

overall -> O(N)
```

Simple array problems

1. Find max element is an array

```
// C++
int find_max(int arr[], int n) {
 // return maximum element from array
}
```

```
# Python
def find_max(arr):
    # return maximum element from array

def find_max(arr):
    max = arr[0]
    for el in arr:
        if el > max:
            max = el
    return max

arr =[1,2,5,7,34,9,0]
maximum= arr[0]
for i in range(len(arr)):
    if arr[i]> maximum:
        maximum=arr[i]
print(maximum)


// Java
public int find_max(int arr[], int n) {
        int max = arr[0];
        for(int i=1; i<arr.length; i++){
            if(arr[i] > max)
                max = arr[i];
        }

        return max;
    }

public static int find_max(int[] arr){
        int max = -1;
        for(int i=0;i<arr.length;i++){
            if(arr[i]>max){
                max = arr[i];
            }
        }
        return max;
    }

int findMax(int arr[], int n) {
    if(n >0) {
        int max = arr[0];
    }

    for (int i=1; i<n; i++) {
        if( arr[i] > max)
            max = arr[i];

    }

    return max;
}


int findMax(int[] arr, int size) {
    // sort the array in ascending order
    arr.sort()
    largest_element = arr[size-1]
    return largest_element
}
```

O(N)

1. Find max element and min element in a single pass

public int[] find_max(int arr[], int n) { if(n == 0) throw new RunTimeException("Arrays is Empty");

```
    int[] result = new int[2];
    int min = arr[0];
    int max = arr[0];
    for(int i=1; i<arr.length; i++){
        if(arr[i] < min)
            min = arr[i];
        if(arr[i] > max)
            max = arr[i];
    }

    result[0] = min;
    result[1] = max;

    return result;

}
```

10, 20,40, 1, 50, 3
0 1 2 3 4 5
N = 6

min 10 10 10 1 1 1
max 10 20 40 40 50 50

i 1 2 3 4 5 6X

~~~
```Javascript
// Akram (Javascript)
function findMaxMin(arr) {
  let max;
  let min;
  if (arr.length <= 0) {
    return [];
  }

  if (arr.length == 1) {
    max = arr[0];
    max = arr[0];

  }

  max = arr[0];
  min = arr[0];

  for (let i = 1; i < arr.length; i++) {
    if (arr[i] > max) {
      max = arr[i]
    }
    if (arr[i] < min) {
      min = arr[i]
    }
  }
  return { max, min };
}
~~~

TC: O(N)

1. Reverse an array in place
2. Built-in min, max, reverse

In [1]:

In [ ]:

```
# swap value of 2 variables

int x = 10, y = 20;
int temp;
temp = x;
x = y;
y = temp;
```

In [ ]:

In [6]:

```

def reverse(arr):
    i = 0
    j = len(arr) - 1
    while i < j:
        arr[i],arr[j] = arr[j],arr[i]
        i+=1
        j-=1

a = [1,2,3,4,5]
a.reverse()
print(a)
reverse(a)
print(a)
reverse(a)
print(a)

# TC: O(N)
```

```
[5, 4, 3, 2, 1]
[1, 2, 3, 4, 5]
[5, 4, 3, 2, 1]
```

In [3]:

```
a=[1,2,3,4]
print(a[::-1])
print(a)
```

```
[4, 3, 2, 1]
[1, 2, 3, 4]
```

In [ ]:

- Array
- Dynamic array

In [ ]:

Iterator

In [ ]:

In [ ]:

In [ ]: