★ Max difference b/w any 2 elements :

eg: [16, 24, 89, 35]
0 1 2 3

(16, 24) (24, 89) (89, 35)
(16, 89) (24, 35)
(16, 35)

i j i j i j

(0, 1) (1, 2) (2, 3)
(0, 2) (1, 3)
(0, 3)

2 index pairs → 0

n=4

i=0, j=1 → 3 (i+1 → n-1)
i=1, j=2 → 3
i=2, j=3 → 3

→ How generate pair of 2 elements ?

```
for (let i = 0; i < n; i++) {
    for (let j = i+1; j < n; j++) {
        console.log(arr[i], arr[j]);
    }
}
```

① i = 0
→ j = i+1 = 0+1 = 1 → arr[0], arr[1]
→ j = 2 → arr[0], arr[2]
→ j = 3 → arr[0], arr[3]
→ j = 4 (4<4) ✗

② i = 1
→ j = i+1 = 1+1 = 2 → arr[1], arr[2]
→ j = 3 → arr[1], arr[3]
→ j = 4 ✗

```
let maxDiff = -Infinity;
for (let i = 0; i < n; i++) {
    for (let j = i+1; j < n; j++) {
        const diff = Math.abs(arr[i] - arr[j]);
        if (diff > maxDiff) {
            maxDiff = diff
        }
    }
}
```

* Total iterations (instructions)

$i = 0,\ j = 1, 2, 3 \rightarrow (3)$

$i = 1,\ j = 2, 3 \rightarrow (2)$

$i = 2,\ j = 3 \rightarrow (1)$

$i = 3,\ j \times \rightarrow (0)$

$n = n \Rightarrow 3 + 2 + 1 = 6$

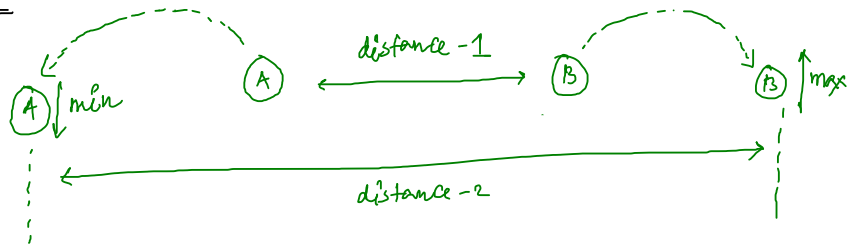
$1 + 2 + 3 + \ldots - + n-3 + n-2 + n-1$

= Sum of $n-1$ natural nums

$= n(n-1)/2$

$= \frac{n^2 - n}{2} \simeq n^2$

Improve :



diff b/w 2 ele = the gap b/w 2 numbers

distance2 > distance1

* maxDiff = maxEle - minEle

⇒ n iterations

* 2nd largest element :

```
        0  1  2  3  4  5  6  7
eg:   [ 3, 1, 2, 5, 5, 4, 5, 3 ]

op: 4
```

[3, 1, 2, 4, 3]

4

① find maxEle (1st max) → 5

② find maxEle by ignoring 1st max elements

maxEle = -∞

1. 3 == 5	1. 1 == 5	1. 2 == 5	1. 4 == 5	1. 3 == 5
2. 3 > -∞ ✓	2. 1 > 3 ✗	2. 2 > 3 ✗	2. 4 > 3 ✓	2. 3 > 4 ✗

3

4

2nd max

```
/* find maxEle */

let secondMax = -Infinity;
for (let i=0; i<n; i++) {
    if (arr[i] != maxEle &&
        arr[i] > secondMax)
    {
        secondMax = arr[i];
    }
}
```

```
685 function SecondLargest(arr, n) {
686   // Write code here
687   let firstMax = -Infinity;
688   for (let i = 0; i < n; i++) {
689     if (arr[i] > firstMax) {
690       firstMax = arr[i];
691     }
692   }
693
694   let secondMax = -Infinity;
695   for (let i = 0; i < n; i++) {
696     if (arr[i] != firstMax && arr[i] > secondMax) {
697       secondMax = arr[i];
698     }
699   }
700
701   console.log(secondMax);
702 }
```

0 1 2 3 4 5 6 7

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

① i = 0

3 != 5 && 3 > -∞

② i = 1

1 != 5 && 1 > 3

③ i = 2

2 != 5 && 2 > 3

④ i = 3

5 != 5 (F) && ~~&&~~

Short circuit

⑤ i = 4

5 != 5 (F)

⑥ i = 5

4 != 5 && 4 > 3

⑦ i = 6

5 != 5 (F)

⑧ i = 7

8 != 5 && 3 > 4

firstMax = 5

secondMax = ~~-∞~~ 3

④

9:20 – 9:35 PM BREAK

~~Complete~~ ~~Syllabus~~ 1st

3 oranges

2 mangoes

1 Software

2nd

oranges

★ Reverse an Array :

Eg: [1, 2, 3, 4, 5]

op: [5, 4, 3, 2, 1]

```
rev_arr = [];
for (let i = n-1; i >= 0; i--) {
    rev_arr.push(arr[i]);
}
/* copy rev_arr → arr
*/
for (let i = 0; i < n; i++) {
    arr[i] = rev_arr[i];
}
```

rev_arr = [] ~~[5]~~ ~~[5,4]~~ ~~[5,4,3]~~ ~~[5,4,3,2]~~

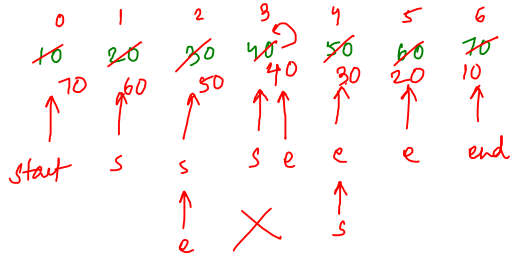
	0	1	2	3	4
	1	2	3	4	5

[5, 4, 3, 2, 1]

★ you should not use an extra array, do it inplace (with original array);

```
arr = [1,2,3,4,5]
Console.log(arr)
/*
    ode
*/
Console.log(arr) → [5,4,3,2,1]
```

Two pointer Approach :




```
let start = 0; 1
let end = 6; 5
swap(arr[start], arr[end])
start++;
end--;
(only perform when s<=e)
```

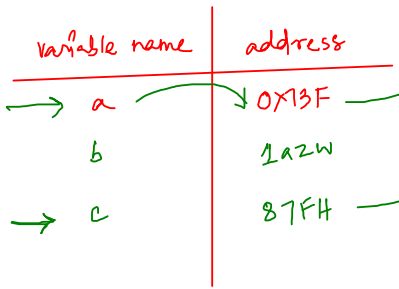
```
let temp = arr[start];
arr[start] = arr[end];
arr[end] = temp;
```

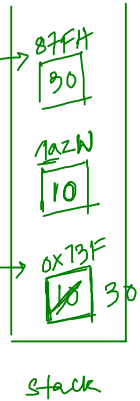
* Inbuilt JS method)

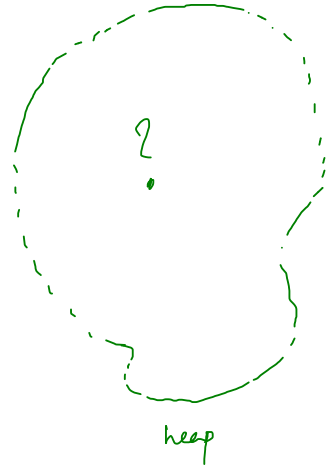
```
const arr = [1,2,3,4,5];
const rev_arr = arr.reverse();
console.log(rev_arr);
```

✱ How arrays work internally ?

```
let a = 10;
console.log(a);
let b = a;
let c = 30;
a = c;
```







→ Internally addresses follow hexadecimal system. (base-16)

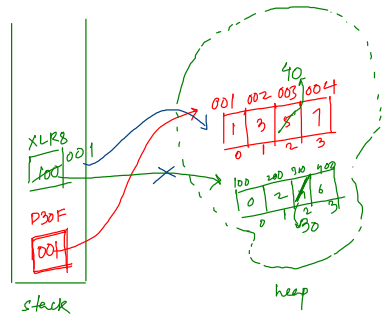
```
const arr = [1, 3, 5, 7];
```

→ arrays are created in heap.

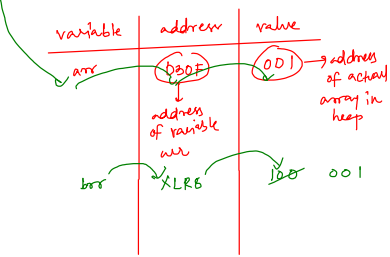
```
console.log(arr);
```

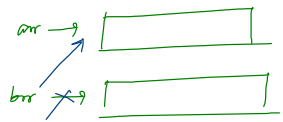
→ arr will store address of 1st element.

```
const brr = [0, 2, 4, 6]
brr[2] = 90; // => 100[2] = 90;
brr = arr; // => brr = 001;
arr[2] = 40; // => 001[2] = 40;
console.log(brr); // => [1, 3, 40, 7]
```



→ arrays use contagious / continuous memory locations.





brr = arr;

~~(all elements of arr are copied to brr → This is wrong)~~

(brr changes its address/direction to arr)

Q1;

```
function increment(a) {
    a++;
}

let a = 2;
increment(a);
console.log(a);
```

≠ why/how you are getting that output ?

Q2;

```
function increment(arr) {
    arr[0]++;
}

let arr = [1, 3, 5, 7];
increment(arr);
console.log(arr);
```