Greater Than Me (Permutation with Repetation)

arr =

0	1	2	3	4	5	6	7
5	3	-2	7	6	-2	4	3

↑ i

logic:- count no. of elements in entire array which are strickly greater than myself

```
for ( int i=0 ; i<n ; i++){
    for ( int j=0; j<n; j++){
        check if jth element > ith element
        then count ++;
    }
}
```

Code

```java
public static void main(String[] args) {
    Scanner scn = new Scanner(System.in);
    int n = scn.nextInt();
    int[] arr = new int[n];
    for (int i = 0; i < n; i++) {
        arr[i] = scn.nextInt();
    }
    greaterThanMyself(arr, n);
}
public static void greaterThanMyself(int[] arr, int n) {

    // permutation with repetation
    for (int i = 0; i < n; i++) {
        int count = 0;
        for (int j = 0; j < n; j++) {
            if ( arr[i] < arr[j] ) {
                count++;
            }
        }
        System.out.print(count + " ");
    }

}
```

Greater At Right (Combination without Repetation)

$i = 0, \; j = i+1$

arr =

0	1	2	3	4	5	6	7
5	3	-2	7	6	-2	4	3

i j

(arr[j] > arr[i])

dry run

i = 0, count = ~~0~~ ~~1~~ 2 ✓

i = 1, count = ~~0~~ ~~1~~ ~~2~~ 3 ✓

i = 2, count = ~~0~~ ~~1~~ ~~2~~ ~~3~~ 4 ✓

i = 3, count = 0 ✓

i = 4, count = 0 ✓

i = 5, count = ~~0~~ ~~1~~ 2 ✓

i = 6, count = 0 ✓

i = 7, count = 0 ✓

psudo code

1) input array

2) traverse in array for i^{th} from 0 to n

   2.1) declare count = 0

   2.2) traverse in array for j^{th} from (i+1) to n

      2.2.1) check if j^{th} element > i^{th} element
      then count++;

   2.3) update i^{th} element with count

Code

```java
public static void main(String[] args) {
    Scanner scn = new Scanner(System.in);
    int n = scn.nextInt();
    int[] arr = new int[n];
    for (int i = 0; i < n; i++) {
        arr[i] = scn.nextInt();
    }

    int[] ans = greaterInRight(arr, n);
    for (int i = 0; i < ans.length; i++) {
        System.out.print( ans[i] + " " );
    }
}
public static int[] greaterInRight(int[] arr, int n) {
    // comb without repe
    for (int i = 0; i < n; i++) {
        int count = 0;
        for (int j = i + 1; j < n; j++) {
            if ( arr[j] > arr[i] ) {
                count++;
            }
        }
        arr[i] = count;
    }
    return arr;
}
```

maximum difference between the two elements

→ find maximum diff. b/w any 2 element

→ larger element is on right side

n = 7

arr =

2	3	10	6	4	8	1
0	1	2	3	4	5	6

arr[i] = myself (smaller element)

arr[j] = other (larger element)

if

Combination without Repetation

Code

```
for( i = 0 ⟶ n ){
    for( j = (i+1) ⟶ n ){
        if( arr[j] > arr[i] ){
            max diff. here
        }
    }
}
```

Code

```java
public static void main(String[] args) {
    Scanner scn = new Scanner(System.in);
    int n = scn.nextInt();
    int[] arr = new int[n];
    for (int i = 0; i < n; i++) {
        arr[i] = scn.nextInt();
    }
    int ans = maxDiff(arr, n);
    System.out.println(ans);
}
public static int maxDiff(int[] arr, int n) {
    int ans = 0;
    for (int i = 0; i < n; i++) {
        for (int j = i + 1; j < n; j++) {
            if ( arr[j] > arr[i] ) {
                int diff = arr[j] - arr[i];
                if ( diff > ans ) {
                    ans = diff;
                }
            }
        }
    }
    return ans;
}
```

n = 7

arr =

2	3	10	6	4	8	1
0	1	2	3	4	5	6

i, j

ans = ~~0~~ ~~1~~ 8

i = 0, (2, 3) diff = 1
(2, 10) diff = 8
(2, 6) diff = 4
(2, 4) diff = 2
(2, 8) diff = 6

i = 1, (3, 10) diff = 7
(3, 6) diff = 3
(3, 4) diff = 1
(3, 8) diff = 5

i = 2, ✗

i = 3, (6, 8) diff = 2

i = 4, (4, 8) diff = 4

i = 5, ✗

i = 6, ✗

Find Duplicate 3

arr =

0	1	2	3	4	5	6
5	3	2	1	7	2	4

(Combination without repetation)

```
for ( int i = 0 ⟶ n ){
    for ( int j = i+1 ⟶ n ) {
        if ( arr[i] == arr[j] ){
            return true;
        }
    }
}
return false;
```

```java
public static void main(String[] args) {
    Scanner scn = new Scanner(System.in);
    int n = scn.nextInt();
    int[] arr = new int[n];
    for (int i = 0; i < n; i++) {
        arr[i] = scn.nextInt();
    }
    boolean ans = findDuplicate(arr, n);
    System.out.println(ans);
}
public static boolean findDuplicate(int[] arr, int n) {
    // comb without repe
    for (int i = 0; i < n; i++) {
        for (int j = i + 1; j < n; j++) {
            if ( arr[i] == arr[j] ) {
                return true;
            }
        }
    }
    return false;
}
```