Agenda (TnS)

→ Searching

→ Linear Search

→ Binary Search

questions + portal + Extra.

→ Sorting-

→ Bubble

→ Selection

→ Insertion

→ Merge

→ Quick

→ Count

→ Radix.

Complexity Analysis

Expectation → searching(arr, lo, hi, data) → true / false.

faith → searching(arr, lo, mid-1, data)
searching(arr, mid+1, hi, data)

mid = $lo + \frac{hi - lo}{2}$

$lo + \frac{(hi - lo)}{2}$

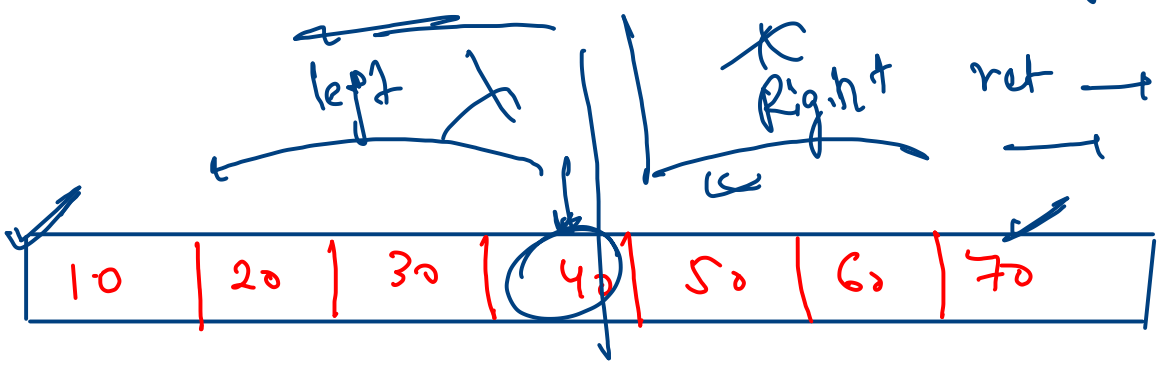
$\Rightarrow lo + \frac{hi}{2} - \frac{lo}{2}$

$= \frac{lo}{2} + \frac{hi}{2}$

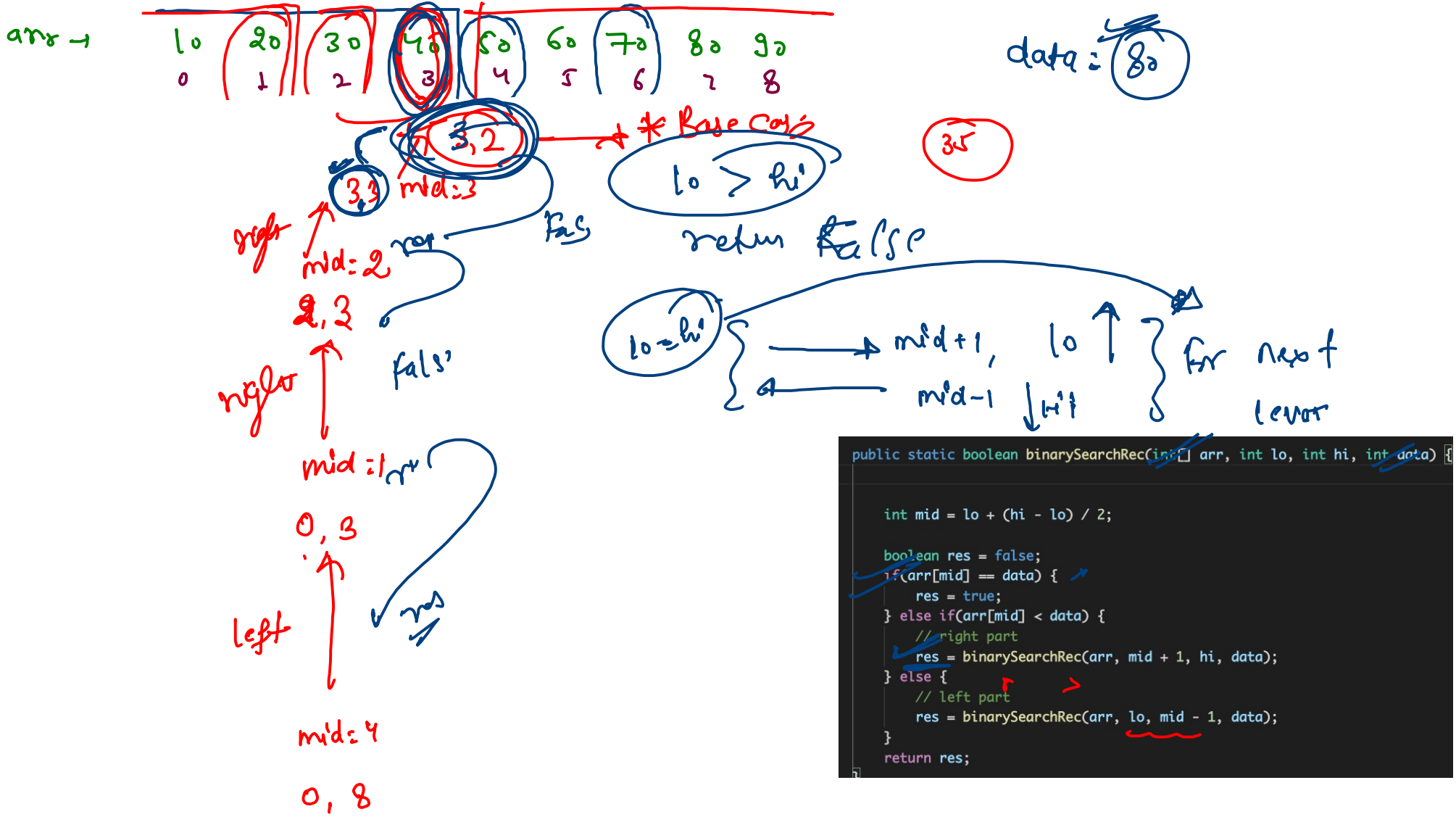
$= \frac{lo + hi}{2}$

data. if(arr[mid] == data) return;

Merging →



ret → left
Right

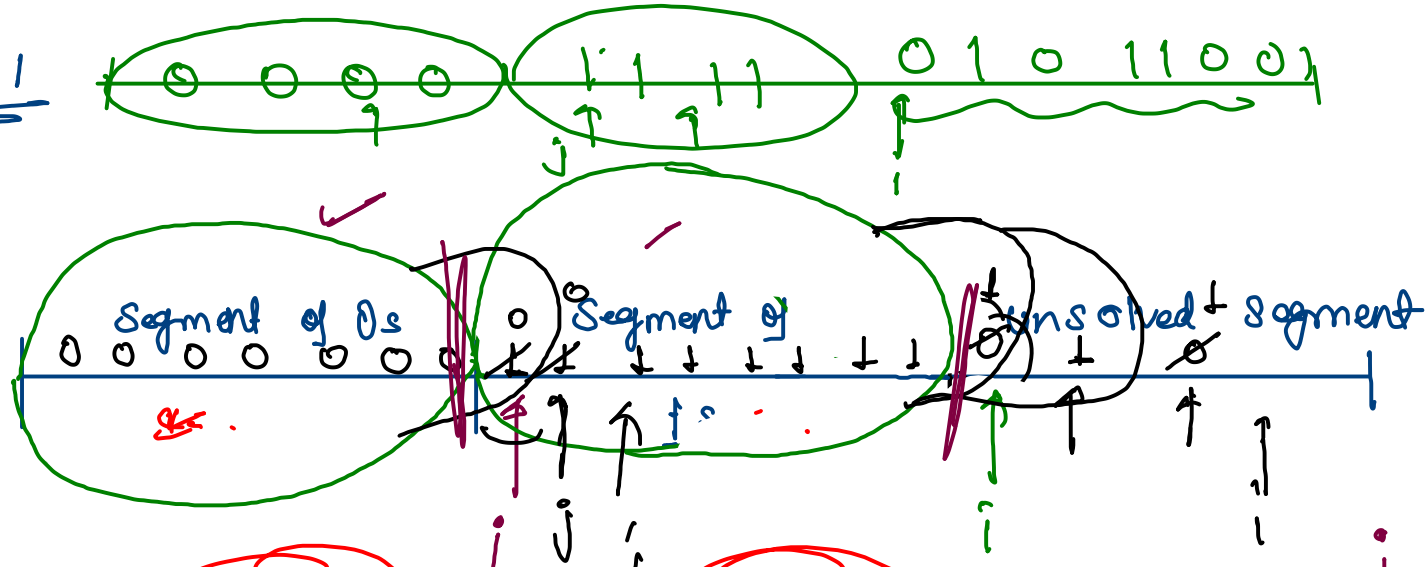



```java
public static boolean binarySearchRec(int[] arr, int lo, int hi, int data) {

    int mid = lo + (hi - lo) / 2;

    boolean res = false;
    if(arr[mid] == data) {
        res = true;
    } else if(arr[mid] < data) {
        // right part
        res = binarySearchRec(arr, mid + 1, hi, data);
    } else {
        // left part
        res = binarySearchRec(arr, lo, mid - 1, data);
    }
    return res;
}
```

Sort 01



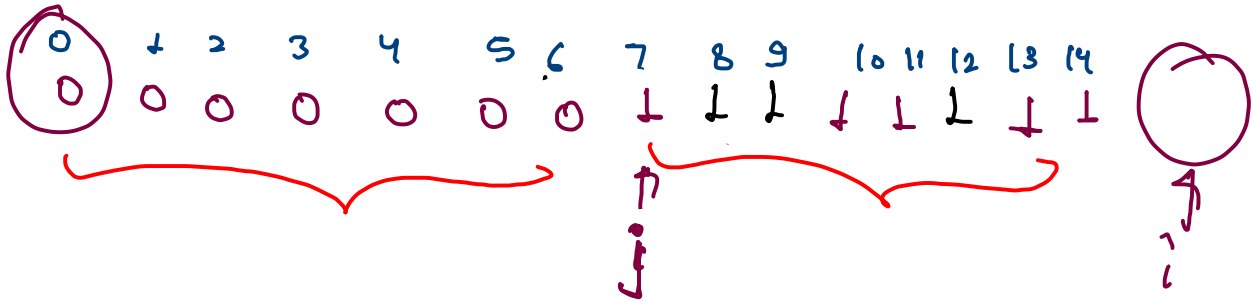
i → first unsolved

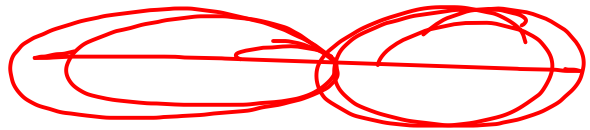
j → first 1

prime	Not
odd	Even
arr[i] = 1	arr[i] = 0

// increment in segment of 1.
i++;
	swap(arr, i, j);
	i++;
	j++;

Segregate Parity



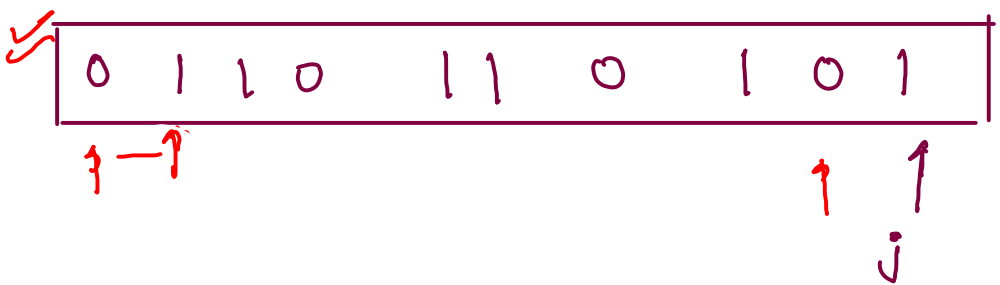


i → First Unsolved i

j → first 1

arr[i] == 0
i++

arr[i] == 1
swap(i, j)
j--;




```
if(arr[i] == 1){
    i++; //Increment in
         segment of 1
} else {
    swap(arr, i, j);
    i++;
    j++;
}
```

i → first unsolved

j = last unsolved

Segregate Odd Even → odd - Even

order of odd & closest in it creg

4 7 6 9 3 8 10 11 14 16 15 20

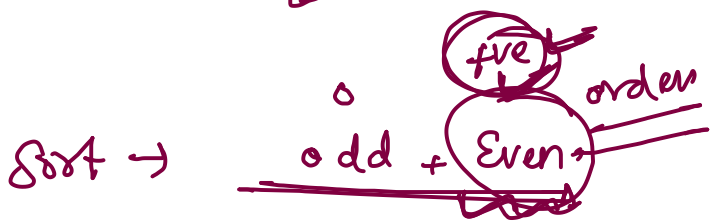
order maintain

→ 7 9 3 11 13 | 4 6 8 10 14 16 20

odd | Even

E_1 O_1 E_2 E_3 O_2 O_3 E_4 O_4

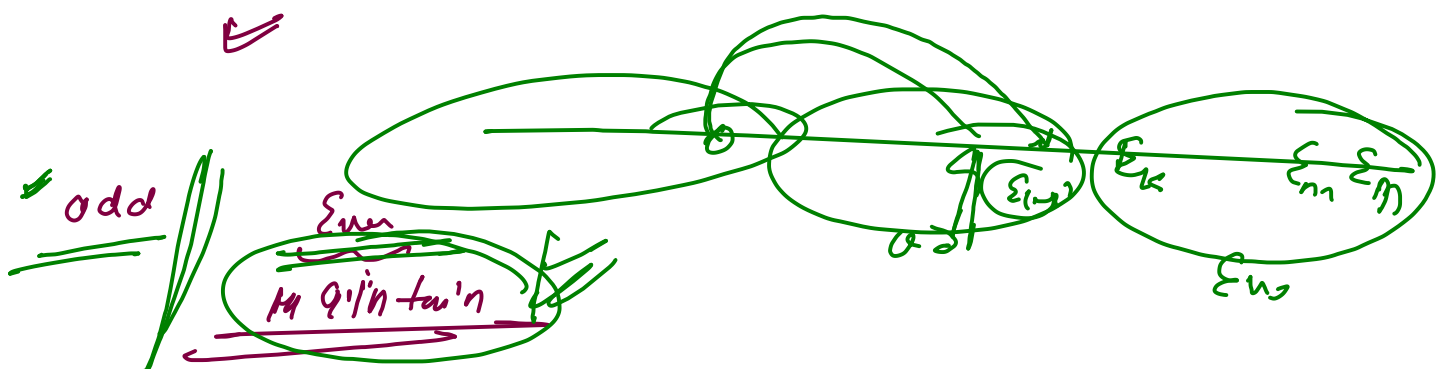
10 1 14 8 5 3 18 7



Even may be arrange in Random order

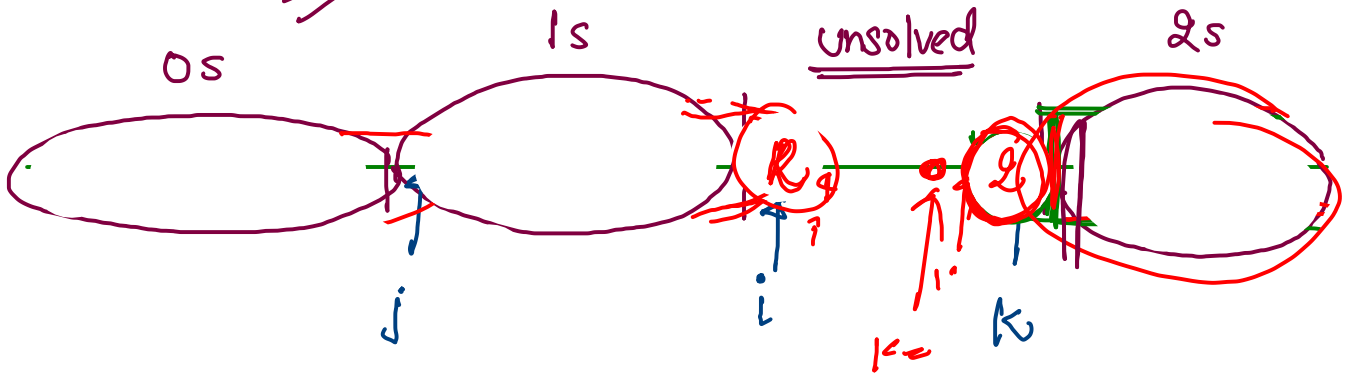
1 5 3 7 { 10 14 8 18 }

order Maintain



Sort 0 1 2 (Dutch Flag)

Algorithm



Quick Sort

i ⇒ first unsolved

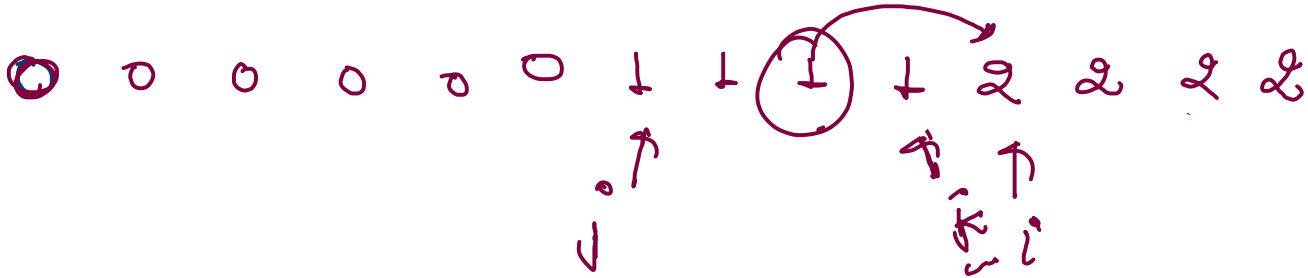
j ⇒ First 1

k ⇒ Last unsolved

} decision factor → 'i'

arr[i] == 1	arr[i] == 0	arr[i] == 2

i++;
 	
// Increment the Region 1
	swap(i, j):
	i++;
	j++;
		swap(i, k):
		k--;



i = 0

j = 0

k = arr.length - 1

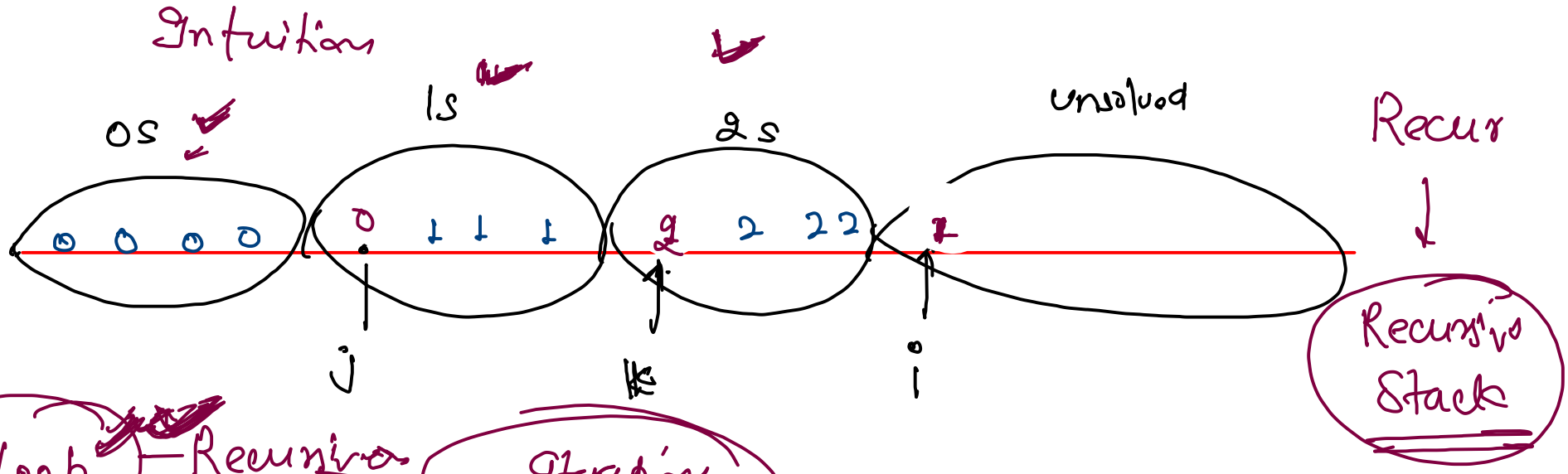
k > i → Stopping.

i == k

Both are unvalued

arr[i] == 0	arr[i] == 1	arr[i] == 2.

swap(i, j);
i++;
j++;
	i++;	swap(i, k);
		k--;



arr[i] == 0

swap(i, j)

swap(i, k)

i++;
j++;
k++;

arr[i] == 1

swap(i, k);

i++;

k++;

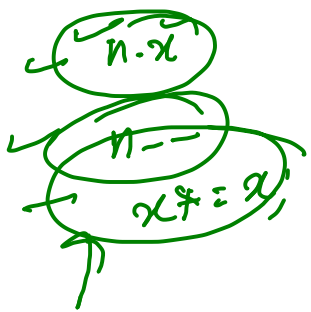
arr[i] == 2.

i++;

Solve Polynomial

$f(x, n) \rightarrow 1 \cdot x^n + 2 \cdot x^{n-1} + 3 \cdot x^{n-2} + \ldots + n \cdot x^1$

$f(x, n) \rightarrow n \cdot x^1 + (n-1) \cdot x^2 + (n-2) \cdot x^3 + \ldots + 1 \cdot x^n$



$O(1)$

n times

$O(n)$

$n(n+1)$

Merge Two Sorted Arrays

```
if(arr1[i] > arr2[j]){
    res[k] = arr2[j]
    j++;
} else {
    res[k] = arr1[i];
    i++;
}
k++
```