

## \* Sort binary array

→ Given an array which only contains 0s & 1s in a random shuffled order, rearrange the array such that all the 0s are placed before 1s.

I/p:- [1, 0, 0, 1, 0]

O/p:- [0, 0, 0, 1, 1]

## \* Conditions:-

- 1) You can iterate over the array only once.
- 2) You cannot create any additional arrays. Do it in-place.

## \* Solution:-

### \* Approach 1:- Bruteforce

→ Consider if we don't have to follow the mentioned conditions, then we can just iterate over the array, count the no. of 0s & update the first count elements of the array with 0 & the next count elements to 1.

function foo(arr) {

let countZero = 0;

for (let i = 0; i < arr.length; i++) {

if (arr[i] === 0) countZero++;

}

for (let i = 0; i < countZero; i++) {

arr[i] = 0;

}

for (let i = countZero; i < arr.length; i++) {

arr[i] = 1;

}

}

TC :-  $O(n)$

SC :-  $O(1)$

## \* Approach 02 :- Simplest approach

$[1, 0, 1, 1, 0]$

→ The element at index 0 is not at its correct position.

→ We don't know how many 0s exist in the array, but we are sure that 0 should be placed at index 0.

→ We create a variable `currentNonZero`, that keeps a track on where we need to place 0.

$\begin{matrix} 0 & 1 & 2 & 3 & 4 \\ [1, & 0, & 1, & 1, & 0] \\ \uparrow \\ c \end{matrix}$       `currentNonZero = 0`

→ We iterate over the array & check if the current element is 0.

$\begin{matrix} 0 & 1 & 2 & 3 & 4 \\ [1, & 0, & 1, & 1, & 0] \\ \uparrow \uparrow \\ c \quad i \end{matrix}$       `currentNonZero = 0`  
`i = 0`

→ Check if `arr[i] == 0` → false, then do `i++`

$\begin{matrix} 0 & 1 & 2 & 3 & 4 \\ [1, & 0, & 1, & 1, & 0] \\ \uparrow \quad \uparrow \\ c \quad i \end{matrix}$       `currentNonZero = 0`  
`i = 1`

→ Check if `arr[i] == 0` → true, swap (`arr[i]`, `arr[currentNonZero]`) & do `currentNonZero++` & `i++`

$\begin{matrix} [0, 1, 1, 1, 0] \\ \uparrow \quad \uparrow \\ c \quad i \end{matrix}$       `currentNonZero = 1`  
`i = 2 3 4 5`

$\begin{matrix} [0, 1, 1, 1, 0] \\ \uparrow \quad \uparrow \\ c \quad i \end{matrix} \Rightarrow \begin{matrix} [0, 1, 1, 1, 0] \\ \uparrow \quad \uparrow \\ c \quad i \end{matrix} \Rightarrow \begin{matrix} [0, 0, 1, 1, 1] \\ \uparrow \\ c \end{matrix}$

## \* Code :-

```
function foo(arr) {
```

```
    let currentNonZero = 0;
```

```
    for (let i = 0; i < arr.length; i++) {
```

```
        if (arr[i] == 0) {
```

```
            [arr[i], arr[currentNonZero]] = [arr[currentNonZero], arr[i]];
        }
```

```
        currentNonZero++;  
    }  
}  
}
```

\* Approach 3:- Two pointer approach:-

```
function foo(arr) {  
    let i=0, j=arr.length-1;  
    while (i<=j) {  
        if (arr[i]==1) {  
            swap(arr[i], arr[j]);  
            j--;  
        } else {  
            i++;  
        }  
    }  
}
```