Sort Array By Parity

You are given an array of 0s and 1s in random order. Segregate 0s on left side and 1s on the right side of the array

Input:

1	0	1	0	1	0	1

Output:

0	0	0	1	1	1	1

Approach 1

Sort the array in increasing order using STL library or using merge sort

```
sort(arr, arr+n)
```

Input:

1	0	1	0	1	0	1

Output:

0	0	0	1	1	1	1

Code:

```
class Solution{
public:
    void segregate0and1(int arr[], int n) {
        sort (arr, arr + n);
        for(int i=0;i<n;i++){
            cout<<arr[i]<<" ";
        }
    }
};
```

Time Complexity : O(nlogn)
Space Complexity: O(1)

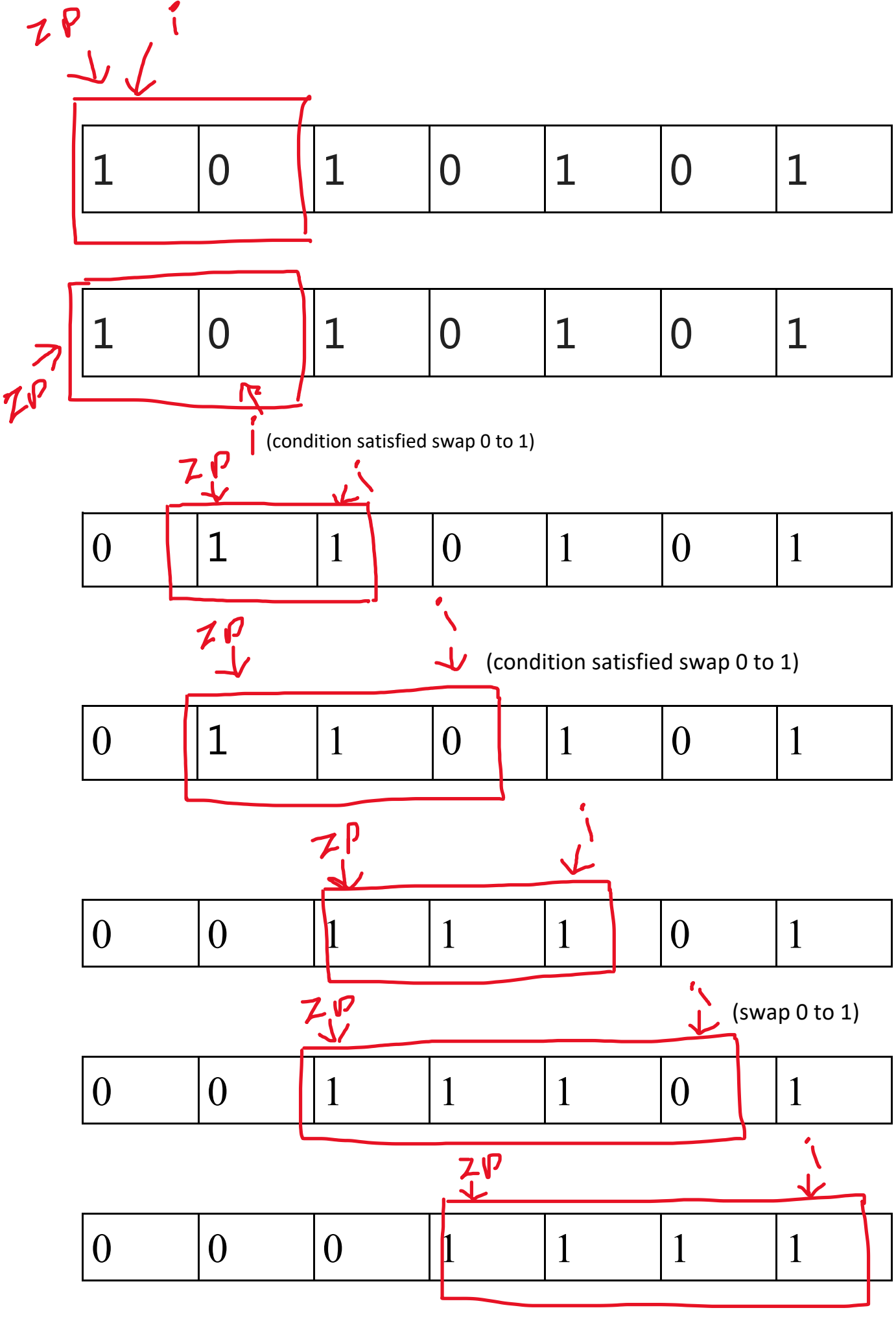
Approach 2 (optimal)

a) Take two pointers zero_pointer and ith pointer, initialize them to 0.
b) Keep incrementing ith pointer to n.
c) Keep checking if arr[i]th equal to 0. If this condition is satisfied swap that number to arr[zero_pointer] and increment zero_pointer.

Intuition Behind the Approach

When we find 0 in the loop we move that 0 to the left part of the array and increment zero_pointer. Through this, we can ensure that before zero_pointer all the elements would be 0 and after that all the elements would be 1.

Dry Run


zp
i
1 0 1 0 1 0 1
zp
1 0 1 0 1 0 1
i
(condition satisfied swap 0 to 1)
zp
i
0 1 1 0 1 0 1
zp
i
(condition satisfied swap 0 to 1)
0 1 1 0 1 0 1
zp
i
0 0 1 1 1 0 1
zp
i
(swap 0 to 1)
0 0 1 1 1 0 1
zp
i
0 0 0 1 1 1 1

Code:

```
class Solution{
public:
    void segregate0and1(int arr[], int n) {
        int zero_pointer = 0;
        for(int i = 0;i<n;i++){
            if(arr[i] == 0){
                swap(arr[zero_pointer], arr[i]);
                zero_pointer++;
            }
        }
    }
};
```

Time Complexity : O(n)
Space Complexity: O(1)

SDE 2 EXPEDIA | Ex - ARCESIUM

lalit kumar

codewithlalit

YouTube
lalit kumar