Approach 1

Use count sort method. Count the number of 0s and 1s in an array. Replace required no of 0s in the left side and 1s in the right side of the array

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

In [3]:
```
def sep_01(arr):
    count_0 = 0
    count_1 = 0
    for i in arr:
        if i==0:
            count_0+=1
        else:
            count_1+=1
    for i in range(len(arr)):
        if count_0 > 0:
            arr[i] = 0
            count_0 -=1
        else:
            arr[i] = 1
    return arr
```

In [5]:
```
arr = [0,0,1,1,1,0,1,0,1]
sep_01(arr)
#ans [0, 0, 0, 0, 1, 1, 1, 1, 1]
```

Out[5]:
```
[0, 0, 0, 0, 1, 1, 1, 1, 1]
```

In [6]:
```
arr = [0,0,1,1,0]
sep_01(arr)
#ans [0, 0, 0, 1, 1]
```

Out[6]:
```
[0, 0, 0, 1, 1]
```

Approach 2

Use 2 pointer technique. place l pointer to left side and r pointer to right side. Increment l pointer until you find 0 and decrement r pointer until you find 1. Swap elements if l ==0 and r ==1 Repeat the process until l>r

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

In [13]:
```
def sep_0_1(arr):
    l = 0
    r = len(arr)-1
    while(l<=r):
        if arr[l] == 1 and arr[r]==0:
            arr[l] = 0
            arr[r] = 1
            l+=1
            r-=1
        if arr[l] == 0:
            l+=1
        if arr[r] == 1:
            r-=1
    return arr
# Not good idea to increment only one at a time. We can increment until we find requi
```

```
In [14]: arr = [0,0,1,1,1,0,1,0,1]
         sep_0_1(arr)
         #ans [0, 0, 0, 0, 1, 1, 1, 1, 1]

Out[14]: [0, 0, 0, 0, 1, 1, 1, 1, 1]
```

```
In [15]: arr = [0,0,1,1,0]
         sep_0_1(arr)
         #ans [0, 0, 0, 1, 1]

Out[15]: [0, 0, 0, 1, 1]
```

```
In [16]: arr = [0,0,1,1]
         sep_0_1(arr)
         #ans [0, 0, 1, 1]

Out[16]: [0, 0, 1, 1]
```

Approach 2 can also be used to separate even and odd numbers . In this case we have to just see Least Significant bit. LSB can be found using %2. Approach 1 cannot be used for this purpose.

```
In [17]: def sep_0_1_enhance(arr):
             l = 0
             r = len(arr)-1
             while(l<r):
                 while(arr[l] == 0 and l < r ):
                     l +=1
                 while(arr[r] == 1 and l < r):
                     r -=1
                 if(l < r):
                     arr[l] = 0
                     arr[r] = 1
                     l +=1
                     r -=1
             return arr
```

```
In [18]: arr = [0,0,1,1,1,0,1,0,1]
         sep_0_1_enhance(arr)
         #ans [0, 0, 0, 0, 1, 1, 1, 1, 1]

Out[18]: [0, 0, 0, 0, 1, 1, 1, 1, 1]
```

```
In [19]: arr = [0,0,1,1,0]
         sep_0_1_enhance(arr)
         #ans [0, 0, 0, 1, 1]

Out[19]: [0, 0, 0, 1, 1]
```

```
In [20]: arr = [0,0,1,1]
         sep_0_1_enhance(arr)
         #ans [0, 0, 1, 1]

Out[20]: [0, 0, 1, 1]
```

```
In [ ]:
```