Sort an array of 0’s and 1’s

Approach 1:

```
#include <bits/stdc++.h>
void sort012(int *arr, int n)
{
   //   Write your code here
   sort(arr,arr+n);
}
```

But this is not expected by the problem.

- Time complexity :O(NlogN)
- Space Complexity : O(N) considering merge sort temporary vector

Approach 2:

Using bucket sort , creating buckets of 0 ,1, 2 and putting the values.

Maintaining 3 variables cnt0,cnt1,cn2 and incrementing the variables when 0,1,2 are found in the array respectively.

Overwrite the values in the array based on these values.

```
#include <bits/stdc++.h>
void sort012(int *arr, int n)
{
   //   Write your code here
   int cnt0=0,cnt1=0,cnt2=0,i;
   for(int i=0;i<n;i++)
   {
      if(arr[i]==0)
         cnt0++;
      else if(arr[i]==1)
         cnt1++;
      else
      {
         cnt2++;
      }
   }
   for(i=0;i<cnt0;i++)
   {
      arr[i]=0;
```

```
    }
    for(;i<cnt0+cnt1;i++)
    {
       arr[i]=1;
    }
    for(;i<cnt0+cnt1+cnt2;i++)
    {
       arr[i]=2;
    }
}
```

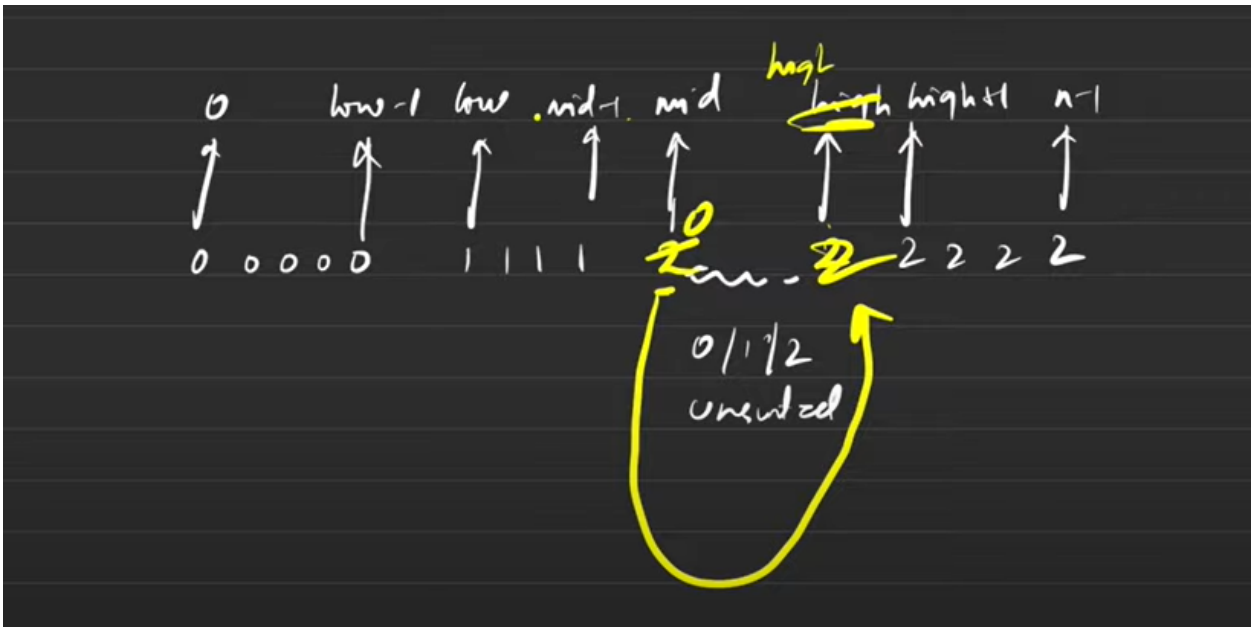
- Time Complexity : O(N)
- Space Complexity: O(1)

Approach 3:

Dutch National Flag Algorithm/Three pointer algorithm

This algo states that we have 3 pointer low,mid and high. Everything from 0 to low-1 is sorted and contains 0 , from low to mid-1 will be 1 , mid to high is the unsorted part of the array and high+1 to n will also be sorted and contains 2.

This rule will follow after each and every iteration of the array.



Our aim is to sort the array in range mid to high -1 so considering we start from mid there can be three condition

arr[mid]==0

In our array 0 lies in the range 0- low so we swap arr[low] and arr[mid] and increment low and mid because now the array will be sorted before low.

arr[mid]==1

low to mid contains 1 so we just increment mid because we found the already sorted elem.

arr[mid]==2

swap arr[mid] and arr[high] decrement high but not mid because the elem present at high can be either 0 or 1 which is not sorted and to sort it we need to keep the mid pointer pointing to it.

```
#include <bits/stdc++.h>
void sort012(int *arr, int n)
{
   //   Write your code here
   int low=0,mid=0,high=n-1;
   while(mid<=high)
   {
      if(arr[mid]==0)
      {
         swap(arr[mid],arr[low]);
         low++;
         mid++;
      }
      else if(arr[mid]==2)
      {
         swap(arr[mid],arr[high]);
         high--;
      }
      else{
         mid++;
      }
   }
}
```

- Time complexity : O(N)
- Space Complexity : O(1)