Sort array with 0s, 1s & 2s

[2 0 2 1 1 0 1 2 0 0]

↓

[0 0 0 0 1 1 1 2 2 2]

① Optimized approach

```
count0 = 0 , count1 = 0 , count2 = 0
for (i = 0; i < n; i++) {
    if (nums[i] == 0)  count0++;
    else if (nums[i] == 1)  count1++;
    else count2++;
}
idx = 0;
for ( i = 0; i < count0; i++) {
    nums[idx++] = 0;
}
for (i = 0; i < count1; i++) {
    nums[idx++] = 1;
}
for (i = 0; i < count2; i++) {
    nums[idx++] = 2;
}
```

② Most optimal / DNF (Dutch National Flag)

O(n) TC with single pass

O(1) SC

3 pointers → low, mid, high

[2 0 2 1 1 0 1 2 0 0]

mid, low ↑ (at index 0) high ↑ (at last index)

0 0 0 | 1 1 1 | unsorted elements | 2 2 2

0 … l-1 | l … m-1 | m … h | h+1 … n-1

gradually it will close up & atp it will be m = h and there are no unsorted elements anymore

0s → 0 to low-1

1s → low to mid-1

2s → high+1 to n-1

unsorted → mid-high

now

A[mid]

- 0 → swap(A[l], A[m]) low++, mid++
- 1 → mid++
- 2 → swap(A[h], A[m]) high--

taking this example

0 0 0 0 1 1 1 0 1 2 2 0 2 2 2

(0 … l-1, l … m-1, m … h, h+1 … n-1)

A[mid] = 0 ⇒ swap(A[l], A[m]) low++, mid++

[0 0 0 0 0] [1 1 1] 1 2 2 0 2 2 2

A[mid] = 1 ⇒ mid++

0 0 0 0 0 1 1 1 1 2 2 0 2 2 2

A[mid] = 2 ⇒ swap(A[h], A[m]) high--

Pseudo code

```
mid = 0, high = n-1, low = 0
while (mid <= high) {
    if (A[mid] == 0) {
        swap (A[low], A[mid])
        mid++, low++
    }
    else if (A[mid == 1)  mid++
    else {
        swap (A[mid], A[high])
        high--;
    }
}
```

Dry run

2 0 2 1 1 0 (m, l at index 0; h at last)

mid = 2 ∴ swap mid & high, high--

0 0 2 1 1 2 (m, l at index 0; h at index 4)

mid = 0 ∴ swap mid & low, mid++, low++

0 0 2 1 1 2 (m, l at index 1; h at index 4)

mid = 0 ∴ swap mid & low, mid++, low++

0 0 2 1 1 2 (m, l at index 2; h at index 4)

mid = 2 ∴ swap h & m, high--

0 0 1 1 2 2 (m, l at index 2; h at index 3)

mid = 1 ∴ mid++

0 0 1 1 2 2 (l at index 2; m, h at index 3)

mid = 1 ∴ mid++

0 0 1 1 2 2 (l at index 2; h at index 3; m at index 4)

now m > h so exit loop and we got sorted array