

Sort an array of 0's, 1's & 2's

↳ Dutch National Flag Algorithm

Intuition

* This algorithm contains 3 pointers i.e, low, mid, high

* And, 3 main rules

- ① $arr[0 \dots low-1] \rightarrow$ contains 0 {Extreme left part}
- ② $arr[low \dots mid-1] \rightarrow$ contains 1
- ③ $arr[high+1 \dots n-1] \rightarrow$ contains 2 {Extreme right part}

* Whatever array you are given, make them into above structure

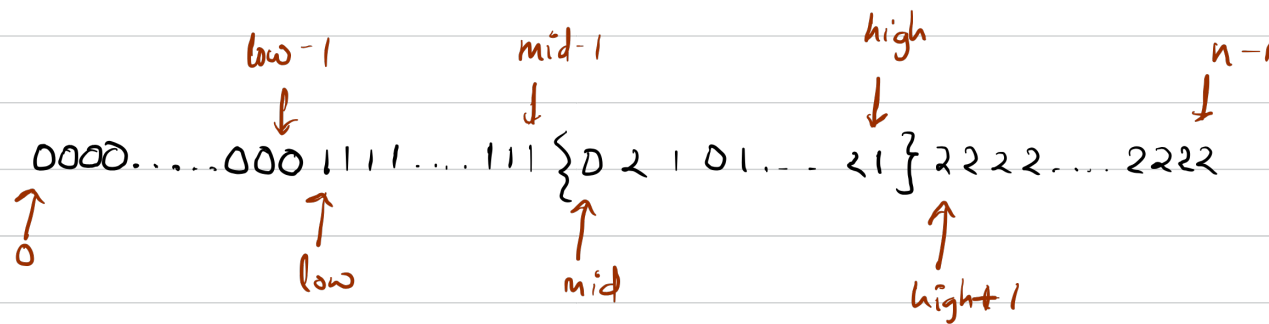
* input array: $\{1, 0, 2, 0, 0, 1, 2, 2, 1, 1, 1\}$

final array: $\{0, 0, 0, 1, 1, 1, 1, 1, 2, 2, 2\}$

Diagram illustrating the partitioning of the input array into three parts based on the Dutch National Flag Algorithm:

- low points to the first 0.
- mid-1 points to the last 1.
- high+1 points to the first 2.

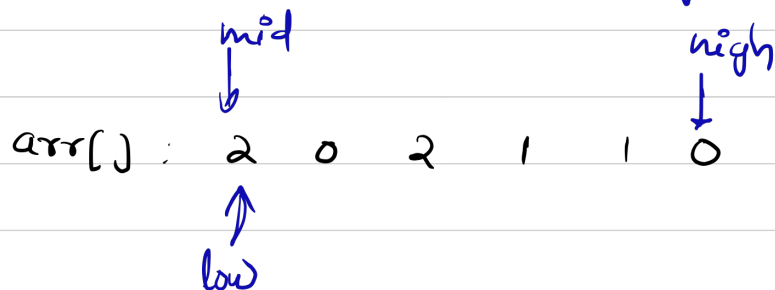
* If you observe carefully, we don't find (mid to high) portion, because this array is already sorted



// The Unsorted portion lies b/w (mid, high)

* We have to arrange these mid to high values so, that our top 3 rules won't break

* In our case, we assume whole array is unsorted, so we place pointers accordingly



* After all the values move to their Original places ,
└ mid & high will erase

Observations

① Case - I

if ($arr[mid] == 0$)

└ We will swap ($arr[low], arr[mid]$)

② increment both mid & low (mid++, low++)

② Case - II

if ($arr[mid] == 1$)

└ ① we will just increment mid (mid++)

Case - II

$\text{if } (\text{arr}[\text{mid}] == 2)$

↳ ① $\text{swap}(\text{arr}[\text{mid}], \text{arr}[\text{high}])$

② decrement high

(high --)

arr[]: [0 1 1 0 1 2 1 2 0 0 0]

↑ low ↑ low ↑ high ↑ high

arr[]: [0 1 1 0 1 2 1 2 0 0 0]

Iterations

① $\text{arr}[\text{mid}] == 0 \rightarrow \text{Case - I}$

arr[]: [0 1 1 0 1 2 1 2 0 0 0]

↑ low ↑ high

② $arr[mid] == 1 \rightarrow \text{Case - II}$

$arr[]: [0, 1, 0, 1, 2, 1, 2, 0, 0, 0]$
low ↑ mid ↓ high ↑

③ $arr[mid] == 1 \rightarrow \text{Case - II}$

$arr[]: [0, 1, 0, 1, 2, 1, 2, 0, 0, 0]$
low ↑ mid ↓ high ↑

④ $arr[mid] == 0 \rightarrow \text{Case - I}$

$arr[]: [0, 1, 0, 1, 2, 1, 2, 0, 0, 0] \Rightarrow arr[]: [0, 0, 1, 1, 2, 1, 2, 0, 0, 0]$
low ↑ mid ↓ high ↑

⑤ $arr[mid] == 1 \rightarrow \text{Case - II}$

$arr[]: [0, 0, 1, 1, 2, 1, 2, 0, 0, 0]$
low ↑ mid ↓ high ↑

⑥ $arr[mid] == 2 \rightarrow \text{Case - III}$

$arr[]: [0, 0, 1, 1, 2, 0, 0, 2]$
low ↑ mid ↓ high ↑

$arr[]: [0, 0, 1, 1, 0, 1, 2, 0, 0, 2]$
low ↑ mid ↓ high ↑

⑦ $arr[mid] == 0 \rightarrow \text{Case - I}$

arr[]: [0 0 1 1 0 1 2 0 0 2]

low points to the first 1, mid points to the first 0, high points to the last 0.

`arr[]`: [0 0 0 1 1 1 1 2 0 0 2]

↑ ↑ ↓
low mid high

Likewise, if you do \rightarrow Our array will be sorted

Time Complexity

 $O(n)$

Space Complexity

0(1)