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

↳ Dutch National Flag Algorithm

Intution

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

* And, 3 main rules

① arr[0 .... low − 1] → contains 0 {Extreme left part}

② arr[low .... mid − 1] → contains 1

③ arr[high+1 .... n−1] → contains 2 {Extreme right part}

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

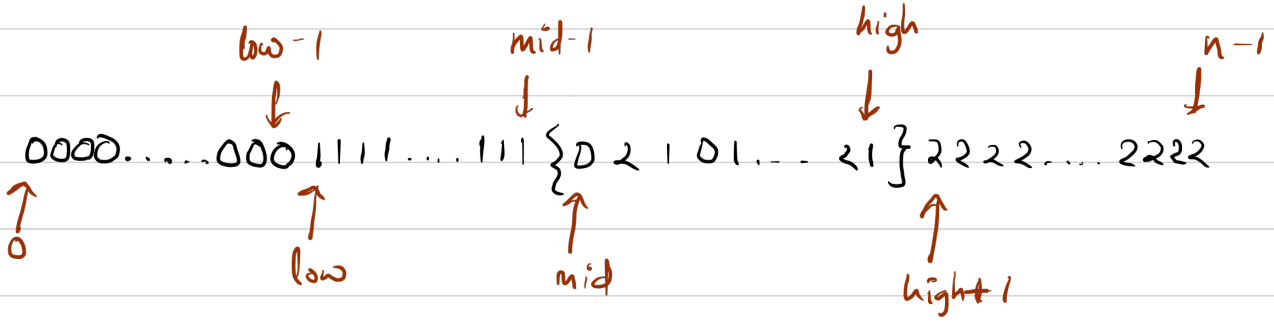
* input array : {1, 0, 2, 0, 0, 1, 2, 2, 1, 1, 1}

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

low, mid−1, n−1

0, low − 1, high+1

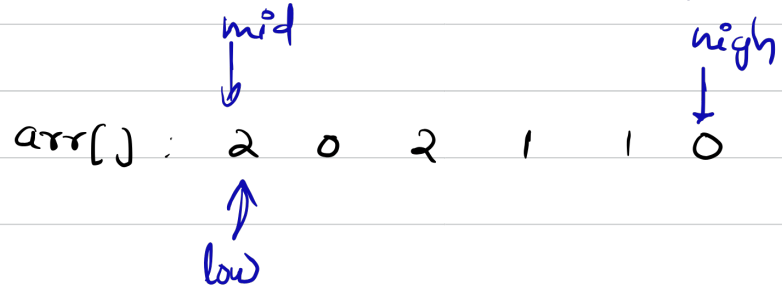
* if you observe carefully, we don't find (mid to high) portion, because this array is already sorted



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

* We have to arange 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)

↳ ① we will swap (arr[low], arr[mid])
② increment both mid & low (mid++, low++)

② Case - II

if (arr[mid] == 1)

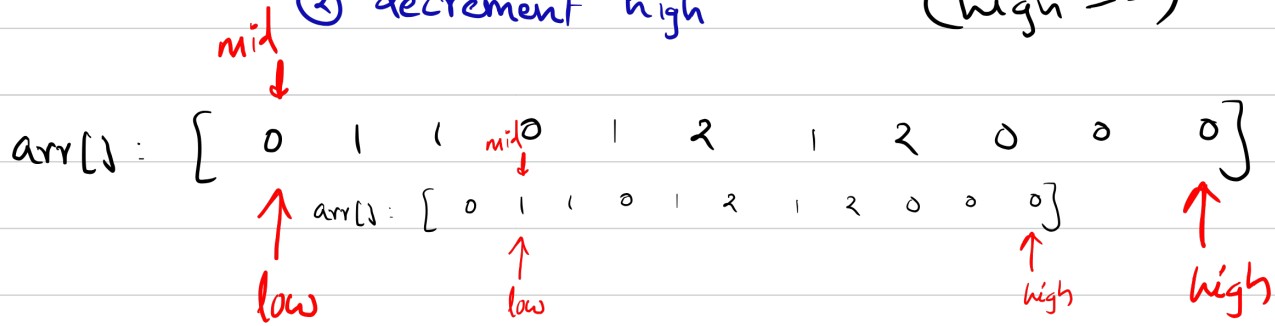
↳ ① we will just increment mid (mid++)

Case - III

if (arr[mid] == 2)

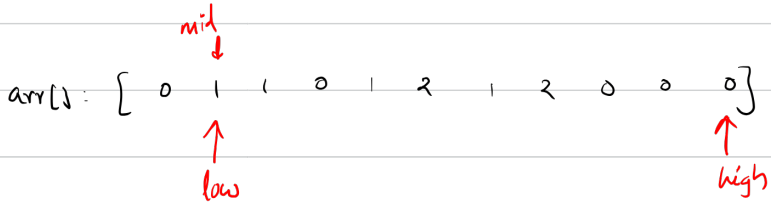
↳ ① swap(arr[mid], arr[high])

② decrement high (high--)

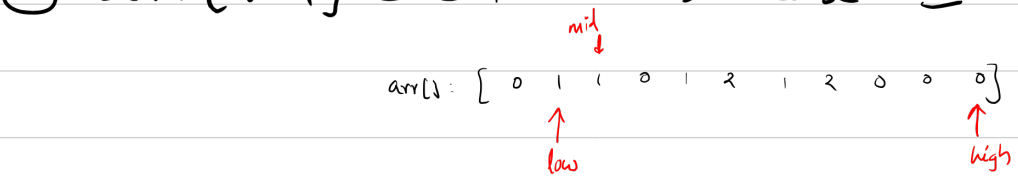


Iterations

① arr[mid] == 0 ⟶ Case - I

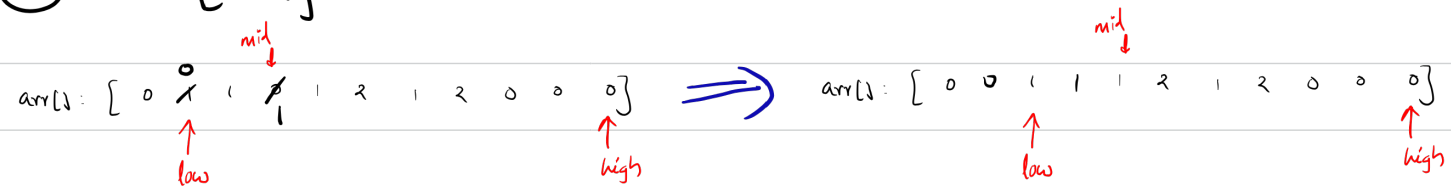


② arr[mid] == 1 ⟶ Case - II

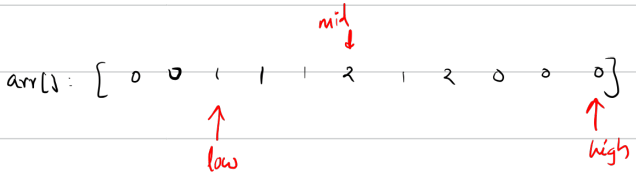


③ arr[mid] == 1 ⟶ Case - II

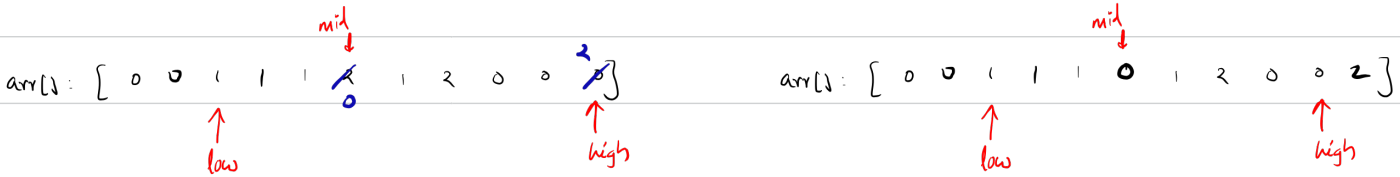
④ arr[mid] == 0 ⟶ Case - I



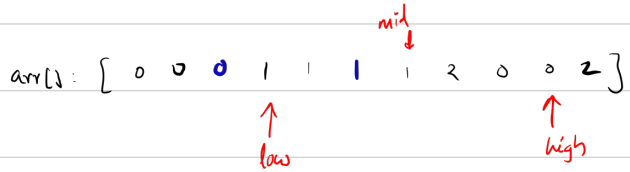
⑤ arr[mid] == 1 ⟶ Case - II



⑥ arr[mid] == 2 ⟶ Case - III



⑦ arr[mid] == 0 → Case - I



↓

Likewise, if you do → Our array will be Sorted

Time Complexity

$O(n)$

Space Complexity

$O(1)$