Sort an Array with 0s, 1s and 2s.

Suppose [2,0,2,1,1,0,1,2,0,0]

Approaches
① Brute Force
② Optimized
③ Optimal

① Brute force:

sort(vec.begin, vec.end())

[O(n log n) T.C] [S.C O(1)]

② Optimized Approach: O(n) (2 passes)

Count of zeros, Ones, Twos ↓ ↓ ↓ 4 3 3 — can be done in single loop

[2,0,2,1,1,0,1,2,0,0] → [0,0,0,0,1,1,1,2,2,2] Overwriting in inner loop

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

O(1) - S.C

Optimal Approach O(n) → Single Pass

→ Dutch National Flag Algorithm

[2,0,2,1,1,0,1,2,0,0] high — Use 3 pointer (low, mid, high)

mid

[0000 111 222]

0 - l-1 | l - m-1 | h+1 - n-1

m — h

Swap — used for Unsorted Elements

0s → 0 to low-1

1s → low to mid-1

2s → high+1 to n-1

Unsorted = (mid - high)

0000 (1)(1)(1) | (2)(2)(2)

0 l-1 | l m-1 | h+1 n-1

m-h

0

~~0000~~

swap(A[l], A[m]) 0

low++

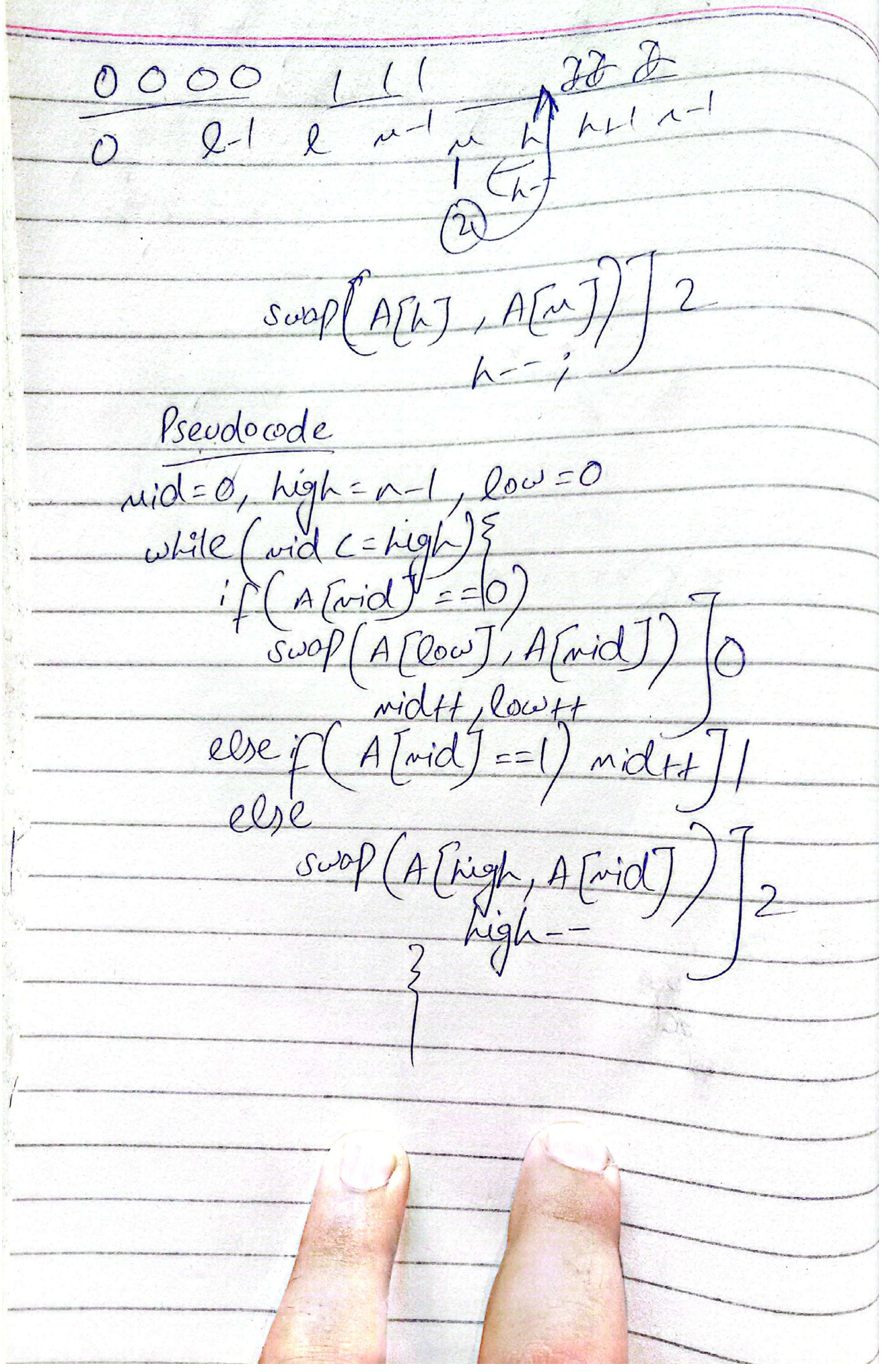
mid++

0000 111 | 222

0 l-1 | l m-1 | m-h h+1 n-1

1 - m++

mid++ 1


0 0 0 0
1 1 1
0
l-1
l
n-1
swap(A[h], A[m]) 2
h--;
Pseudocode
mid=0, high=n-1, low=0
while(mid <= high){
if(A[mid]==0)
swap(A[low], A[mid]) 0
mid++, low++
else if(A[mid]==1) mid++ 1
else
swap(A[high], A[mid]) 2
high--
}