# HOMEWORK 5 SOLUTIONS

1. No change to the array

| j | Action | Array Contents |
|---|---|---|
| 0 | A[0] = 1 ≤ 8 i = 0 Exchange A[0] with A[0] | { 1, 5, 3, 2, 0, 4, 6, 8 } |
| 1 | A[1] = 5 ≤ 8 i = 1 Exchange A[1] with A[1] | { 1, 5, 3, 2, 0, 4, 6, 8 } |
| 2 | A[2] = 3 ≤ 8 i = 2 Exchange A[2] with A[2] | { 1, 5, 3, 2, 0, 4, 6, 8 } |
| 3 | A[3] = 2 ≤ 8 i = 3 Exchange A[3] with A[3] | { 1, 5, 3, 2, 0, 4, 6, 8 } |
| 4 | A[4] = 0 ≤ 8 i = 4 Exchange A[4] with A[4] | { 1, 5, 3, 2, 0, 4, 6, 8 } |
| 5 | A[5] = 4 ≤ 8 i = 5 Exchange A[5] with A[5] | { 1, 5, 3, 2, 0, 4, 6, 8 } |
| 6 | A[6] = 6 ≤ 8 i = 6 Exchange A[6] with A[6] | { 1, 5, 3, 2, 0, 4, 6, 8 } |
| 7 | (Exit the loop) Exchange A[7] with A[7] Return 7 | { 1, 5, 3, 2, 0, 4, 6, 8} |

3.

One possible O(n) solution: Have two pointers, one starts from the front and the other starts from the back.

while front < back

- if front is -ve and back is +ve, increment front and decrement back
- If front is +ve and back is -ve, swap them, increment front and decrement back
- If front is -ve but back is also -ve, increment front
- If front is +ve and back is +ve, decrement back

Another possible O(n) solution is to start both pointers at the front. One pointer keeps moving forward, and when it finds a negative number, it swaps it with the other pointer and both pointers are incremented.

Another possible option is to use one round of QuickSort Partition using 0 as the pivot value

4.

| # | Solution/Points |
|---|---|
| 1 | Sorting is beneficial to find the mth smallest item. Total runtime would be O(nlogn) (to sort) + O(m) = O(nlogn). Without sorting, it could take O(mn) which is quadratic. |
| 2 | Can be done in O(nlogn) with sorting. Sort the array, then loop through it and keep track of the max count. Total run time would be O(nlogn) + O(n) = O(nlogn). Without sorting, it could take upto O(n^2).<br>(You may also do it without sorting, by using Hashing. In that case it would only take O(n) to find the mode) |
| 3 | Yes sorting helps. Sort both arrays merge to find the median.<br>Can be done in O(nlogn) with sorting. |
| 4 | No sorting doesn't help in this case. Can be done in O(n) without sorting, by just keeping track of a counter, looping through the array and incrementing the counter every time the particular item is seen. Sorting on the other hand would take O(nlogn) runtime, which is less efficient. |
| 5 | Nope no sorting here. Can be done in just one iteration i.e. O(n), by looping through the array and keeping track of the minimum item seen so far.<br>Sorting would take a minimum of O(nlogn) which is less efficient |