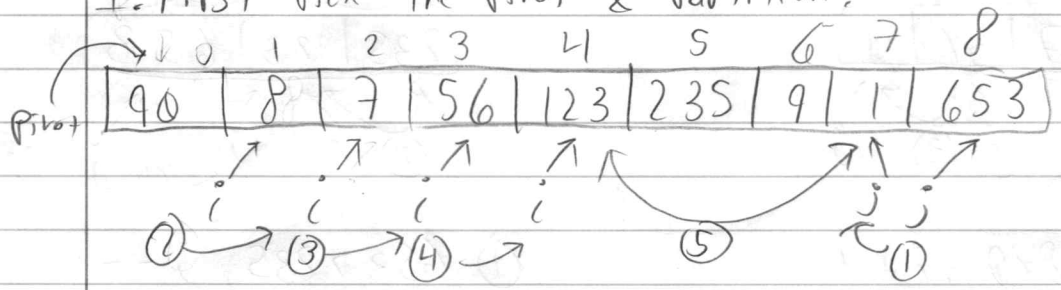
Problem #1 4/8/18

Name: Carlos Flores, PID: 5160328

a.i) Given: $A = \{90, 8, 7, 56, 123, 235, 9, 1, 653\}$

Sort the array A using QuickSort.

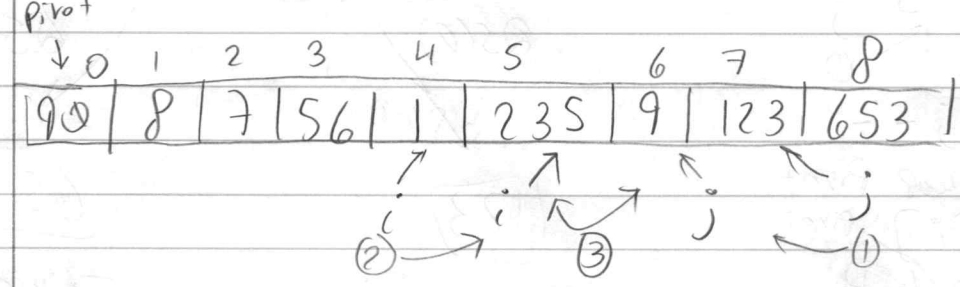
1. First pick the pivot & Partition.



① $653 > 90$, j-- ② $1 \not> 90$, $8 < 90$, i++

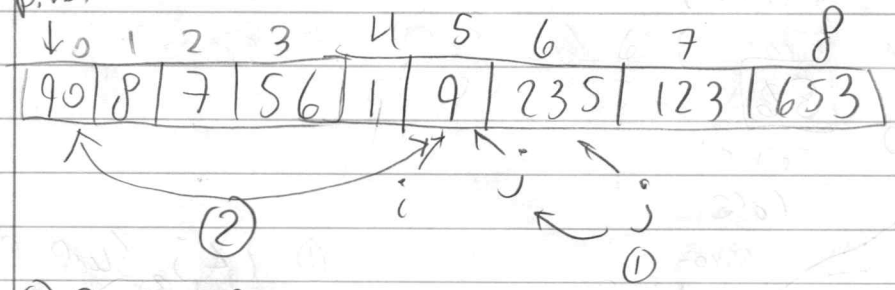
③ $1 \not> 90$, $7 < 90$, i++ ④ $1 \not> 90$, $56 < 90$, i++

⑤ $1 \not> 90$, $123 \not< 90$, Swap A[i] & A[j]



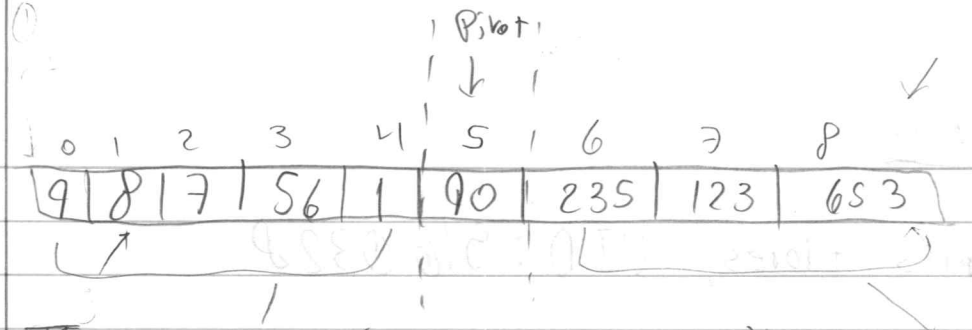
① $123 > 90$, j-- ② $9 \not> 90$, $1 < 90$, i++

③ $9 \not> 90$, $235 \not< 90$, Swap A[i] & A[j]

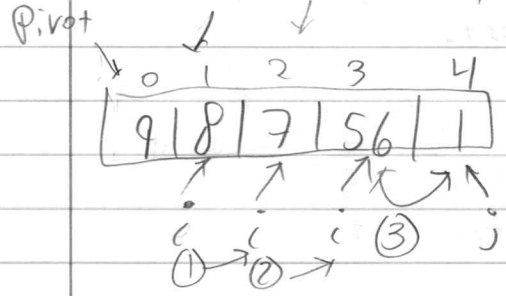


① $235 > 90$, j--

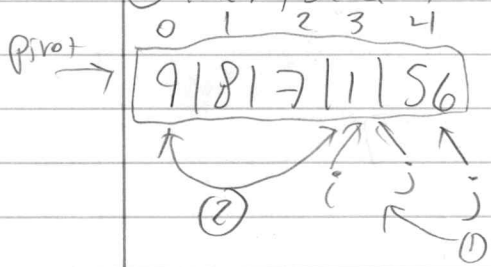
② $i \not< j$, Swap pivot & A[i]



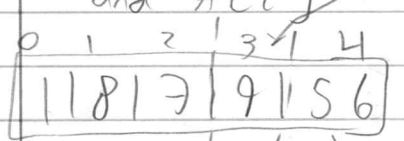
II. QuickSort (A, left, pivot-1), Quicksort (A, pivot+1, right)

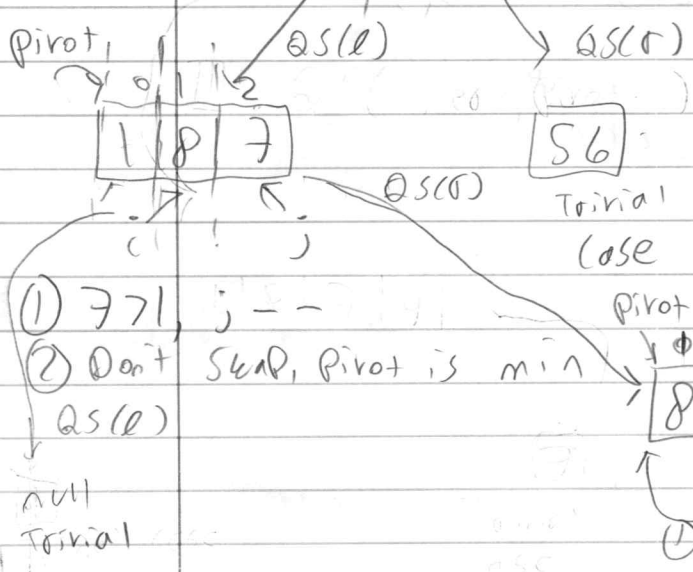


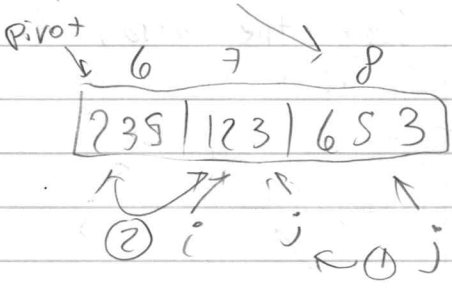
① 1 < 9, 8 < 9, i++
② 1 < 9, 7 < 9, i++
③ 1 < 9, 56 < 9, Swap A[i] & A[j]



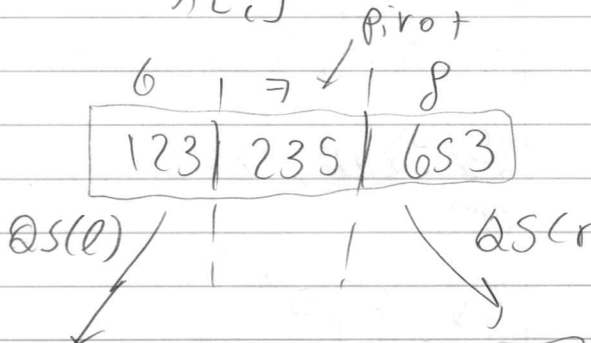
① 56 > 9, j--
② i < j, Swap Pivot and A[i]

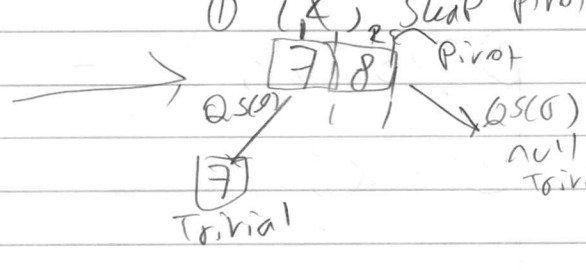






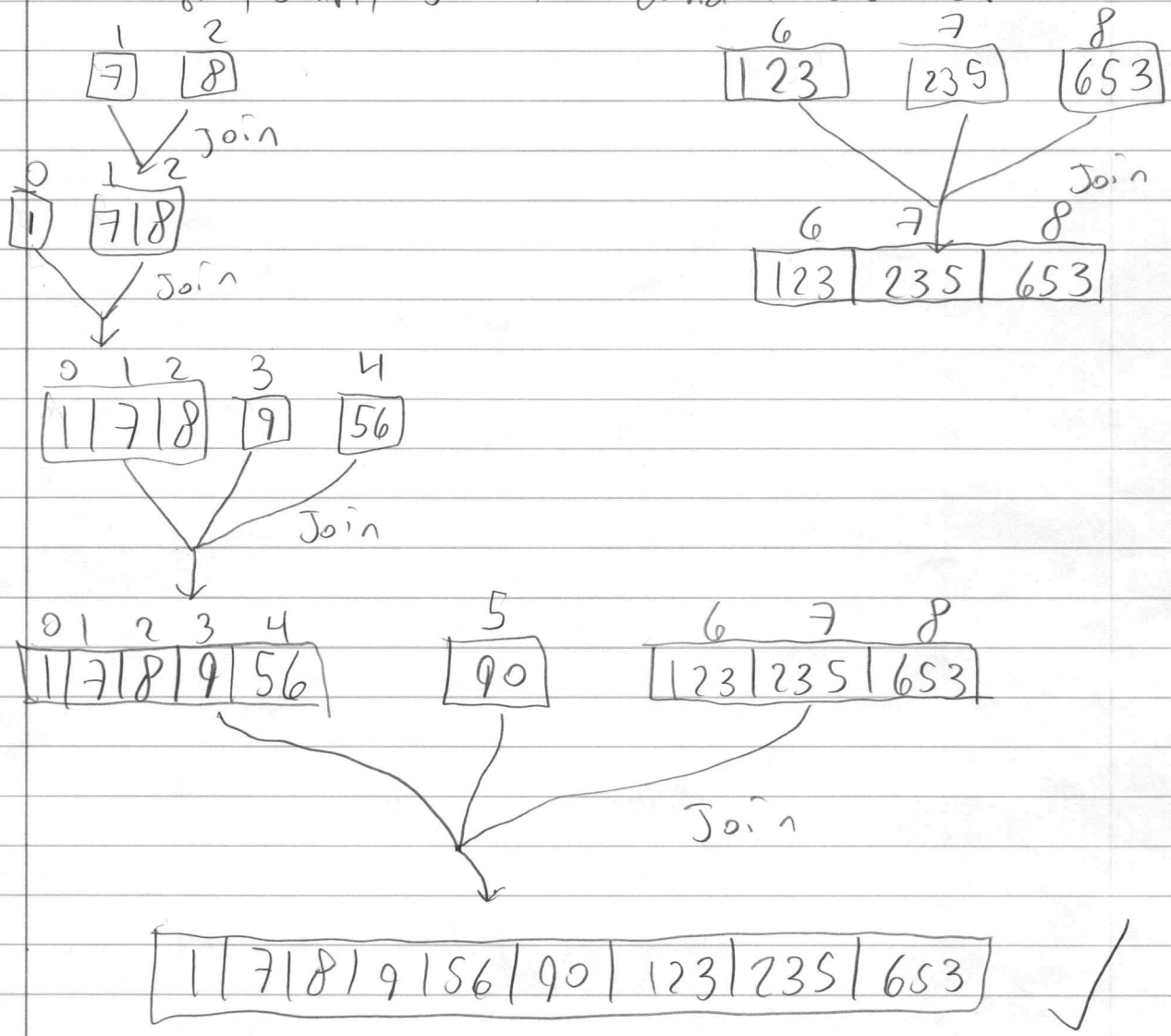
① 653 > 235, j--
② i < j, Swap Pivot & A[i]



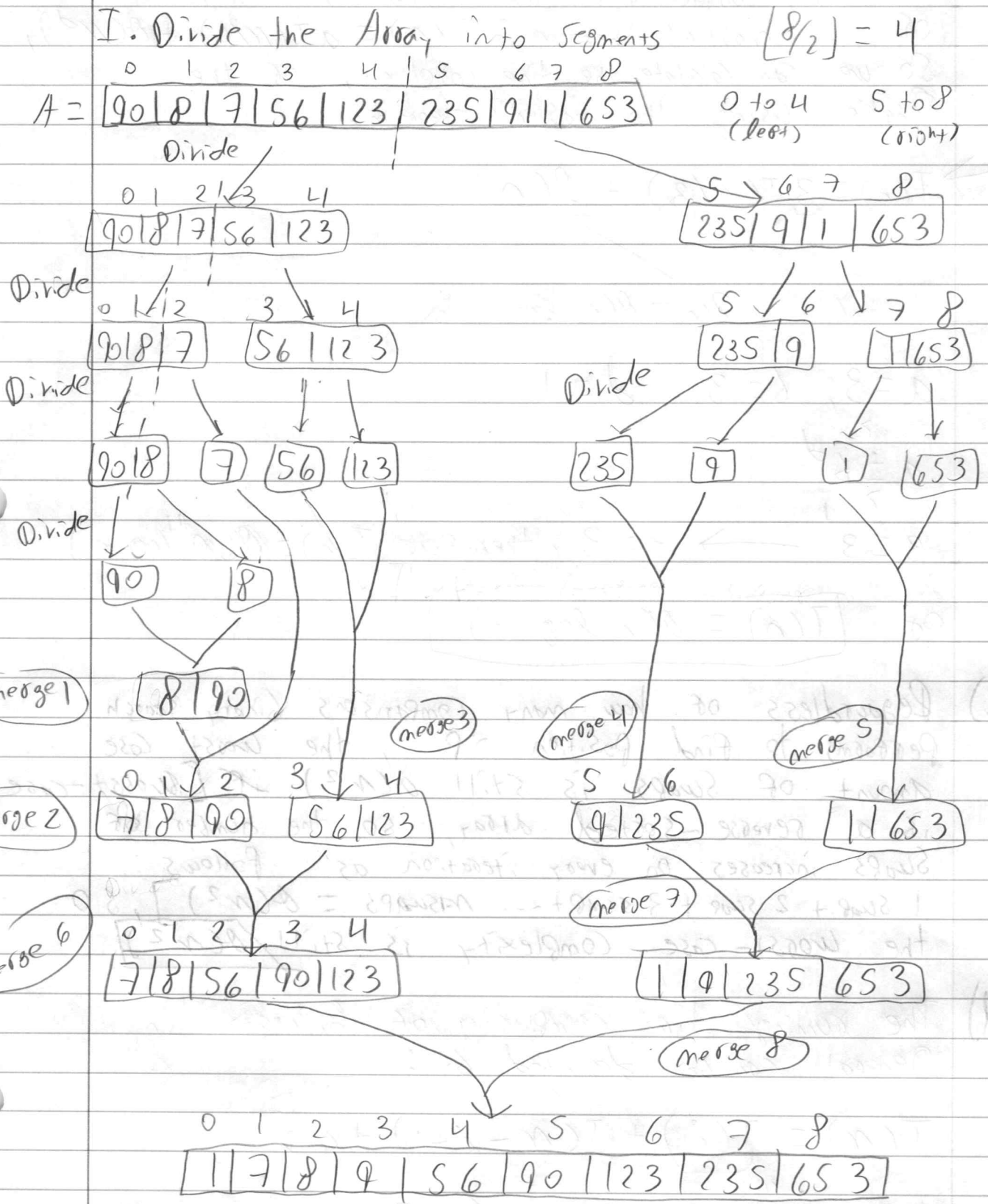


III. We have reached the toirial cases and have divided as much as we can. Time to conquer...

To conquer, Simply join the divided segments.



a.2) Sort the array A using MergeSort



b) $T(n) = 3T(n/3) + 2n; \quad T(1) = 0$

This time complexity function follows the signature of the master's theorem: $T(n) = aT(n/b) + \theta(n^d)$, so we can calculate the time complexity of the function using the master's theorem.

$T(n) = \underline{3}\,T(n/\underline{3}) + \theta(n^{\underline{1}})$

$2n = \theta(n)$.

$a = 3, \quad b = 3, \quad d = 1$

$a \overset{?}{=} b^d$

$3 \overset{?}{=} 3^1 \longrightarrow 3 = 3$, therefore $T(n) = \theta(n^1 \log n)$

or $\boxed{T(n) = \theta(n \log n)}$

c) Regardless of how many comparisons binary search performs to find position "p", the worst case amount of swaps is still $\theta(n^2)$ if [worst-case is a reverse-sorted array, so the number of swaps increases on every iteration as follows: 1 swap + 2 swaps + 3 swaps + ... n swaps $= \theta(n^2)$], so the worst-case complexity is still $\boxed{\theta(n^2)}$

d) The running time complexity of Quicksort in general can be described by:

$T(n) = T(i) + T(n - i - 1) + cn$

where "i" is the size of S_1 partition, and "$n - i - 1$" is the S_2 partition (-1 excludes the Pivot)

↳ "cn" represents the linear time spent creating the partitions.

Because all of our elements are equal, we can consider that the pivot is virtually the smallest element, and therefore $i = |S_1| = 0$ since no other element is "smaller" than the pivot.

This reduces our complexity to:

$$T(n) = T(0) + T(n-1) + cn \quad , n > 1,$$

but we can consider $T(0) = 1$ because it is a base-case and it simply returns, and can therefore be ignored, and the equation further simplifies:

$$T(n) = T(n-1) + cn \quad , n > 1.$$

↳ Because this is a recursive function, we can telescope.

$$T(n-1) = T(n-2) + c(n-1)$$
$$T(n-2) = T(n-3) + c(n-2)$$
$$T(n-3) = T(n-3) + c(n-3)$$
$$\vdots$$
$$T(2) = T(1) + c(2)$$

↳ Plugging them back into $T(n)$ results in:

$$T(n) = T(1) + (c(2) + c(3) + c(4) + \ldots c(n-1) + c(n) = \theta(n^2)$$

Therefore our worst-case is still

$$\theta(n^2)$$