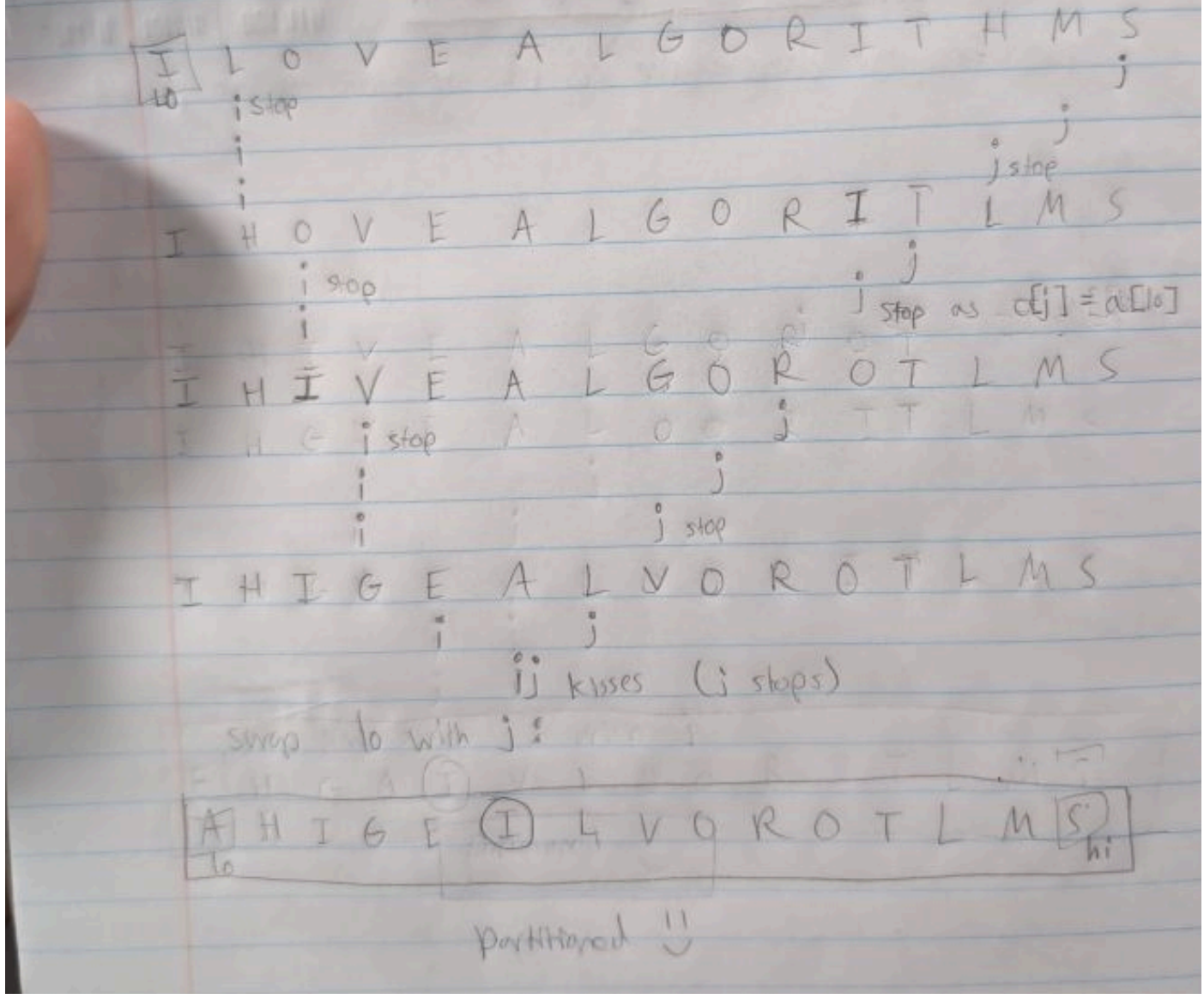
1) Quicksort

a) Show, in the style of the trace given with partition(), how that method partitions the array I, L, O, V, E, A, L, G, O, R, I, T, H, M, S

VVV

a) For partition method, we pick a pivot and utilize two increments. I will use first element as a pivot.

Repeat until i and j pointers cross
- scan i from left to right so long as (a[i] < a[lo])
- scan j from right to left " " (a[i] > a[lo])
- exchange a[i] with a[j]



b) Show, in the style of the quicksort trace discussed in the book, how quicksort sorts the array I, L, O, V, E, A, L, G, O, R, I, T, H, M, S (for the purpose of this exercise, ignore the initial shuffle.) Compare the best, worst, and average case scenarios for the quicksort algorithm.

VVV

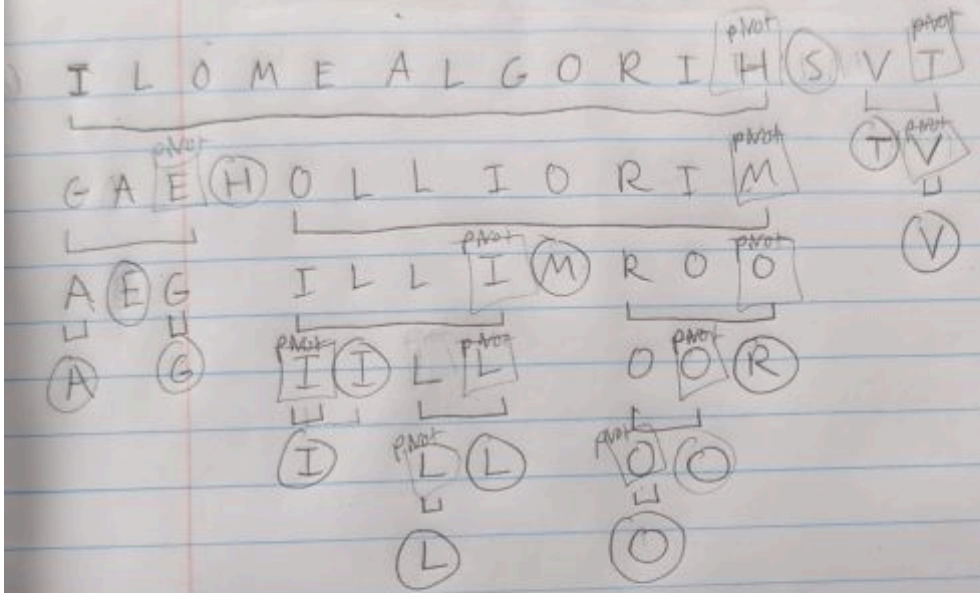
b) For quicksort, we will select the last value as our pivot

I	L	O	V	E	A	L	G	O	R	I	T	H	M	S
														pivot

if i > pivot → pass

if i ≤ pivot → j++ → if i > j → swap(i, j)

↳ if i == j → pass



So the final indices become:

A E G H I I L L M O O R S T V

Best case: $O(n \log(n))$ each subarray must still be sorted after partitioning

Worst case: $O(n^2)$ occurs when largest or smallest element is picked as the pivot for each run

Avg case: $O(n \log(n))$ same reason as best case