Problem Set 1 - General

Question F

This question was a weird ride - at first, I had thought it was WAY easier than expected. I thought the parameters for the list of swaps was purely that it needed to end up in sorted order by the end - and given the n^2 time constraint, it would just be as simple as finding the max element in the positions 1-n, sorting that to the last position. Honestly, I had a feeling this was way too simple to be the solution, but my friends had said it was quite an easy question, so maybe?... But nah, it turns out I missed a pretty key sentence, which was the fact that it needed to have all of the pairs that formed inversions inside the final list.

So the next approach I thought of was to explore was trying to find a way to identify all of the inversions in a list, and then potentially order them in a way that gave us a final answer? Finding all of the inversions was just as simple as, for every element, checking if there are any elements proceeding that are smaller - if so, counting them. My attempt of implementing that into the algo I already has was to, for every largest element, look at all elements to the right (as inversions can only happen for smaller elements to the right) and then note down all of the inversions for that largest element in the order they needed to be applied to push the largest element to the end, apply them, and then attempt that for the second largest, and so on… However, from some testing, I found that this would omit some results - namely, see the test case below and how we lose the original 6 7 swap.

Ind	1	2	3	4	5	6	7
	5	2	5	7	10	8	1
Step							
A		10					
		5	7				
		5	6				
Ind	1	2	3	4	5	6	7
	5	2	5	7	1	8	10
B		8					
	Ah but we've lost the 6 7 swap in the original						

Ultimately, I started to explore the idea of using some sort of bubble sort approach for this question. I intuitively had a feeling that Bubble Sort would account for every inversion in the array as, for every element largest to smallest - it would bubble (I knew that bubble sort inherently would fix inversions one at a time by swapping adjacent elements, which would probably ensure that every single inversion in the array is explicitly resolved at some step.

In this case though, a normal bubble sort, combined with noting down the index swaps that happen wouldn't work - as even though it correctly sorts the array, it wouldn't record the original index position of the inversions that happen - to fix this, when reading in the elements, I would store their original indexes alongside in an array of pii's.

While running bubble sort, I would then note down the original indexes of the elements being swapped, and then print out the list at the end (in reverse order) in order to get our result.

Not too bad of a question, good value F.