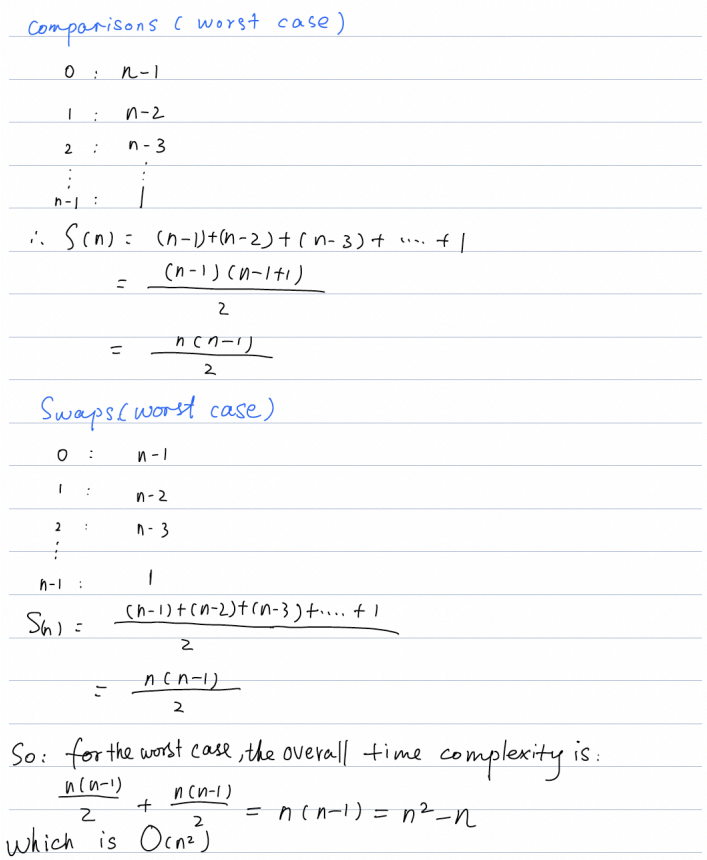
1. Bubble sort is O(n^2). Looking at the algorithm can you explain to your group why that is?

- Outer Loop: The outer loop runs n times, where n is the size of the array. This loop ensures that each element is bubbled to its correct position.
- Inner Loop: For each iteration of the outer loop, the inner loop runs up to n−1−i times, where i is the current iteration of the outer loop. This loop compares adjacent elements and swaps them if they are in the wrong order.
- Number of Comparisons: In the worst-case scenario, the inner loop performs approximately n(n-1)/2 comparisons.
- Swaps:In the worst-case scenario, the inner loop performs approximately n(n-1)/2 comparisons.



2. Can you come up with an array that will generate the "worst case"?

worst case: [5, 4, 3, 2, 1]

3. Can you come up with an array that will generate the "best case"?

best case:[1, 2, 3, 4, 5]

* By default, the best case is the same, but can you optimize it?

Yes

* Given the example above, optimize it so it stops running after the array is sorted?

We can add a flag(in my code, I used is_swapped) to detect whether any swaps were made.
At the start of each outer loop iterate, I set is_swapped to 0.
Then, as j iterates through the inner loop, if any swap occurs, I set is_swapped to 1.
After completing the inner loop, check the is_swapped flag. If it remains 0, it means the array is already sorted, and we can break out of the loop early.

* The optimized version is more common, and gives us the "best case".

best case:[1, 2, 3, 4, 5]

After the first pass, the is_swapped flag remains 0 because the array is sorted and no swaps are needed. The algorithm terminates early, and it performs O(n) comparisons.