Sorting II

Anh Huynh

1. Proof that, under the average-case scenario, the insertion sort has a time complexity of $O(N^2)$. Draw a clear figure and show all the operations clearly. 2 pts
    - **Drawn figure is on pg. 3 of this document**
    - **Explanation:** In an average-case scenario, sometimes an element will have to swap multiple times and move through already sorted portions of the array, thus making the time complexity $O(N^2)$. On pass i, the inspected element may need to shift past about half of the i elements that are already sorted.

      Early passes involve a few shifts but later passes may require the key element to move through many elements. This growing chain of comparisons and shifts leads to an $O(N^2)$ time complexity.
2. At the start of the insertion sort, the index of the inspected value is set to 1. Change the index of the inspected value and verify that the total number of operations equals 20. Consider the worst-case scenario. Use N=5, where N is the number of elements. 4 pts
    - **Answer is on pg 4 of this document**
3. The following function returns whether or not a capital “X” is present within a string. 4 pt

```
function containsX(string) {
  foundX = false;
  for(let i = 0; i < string.length; i++) {
    if (string[i] === "X") {
      foundX = true;
    }
  }
  return foundX;
}
```

(a) What is this function’s time complexity regarding Big O Notation?

- The function's time complexity is O(N) since the function checks each character in the string once, from index 0 up to string.length - 1. The for-loop continues to check the whole string even if 'X' is found before the last character of the string because it just sets the variable to true once string[i] === 'X' instead of exiting the for-loop once 'X' is found.
- The worst case is there is no 'X' present or 'X' is the very last character so it has to scan the entire string (N comparisons).
- The best case is that 'X' is the first character so it returns immediately after 1 comparison (O(1)).

(b) Then, modify the code to improve the algorithm’s efficiency for best- and average-case scenarios.

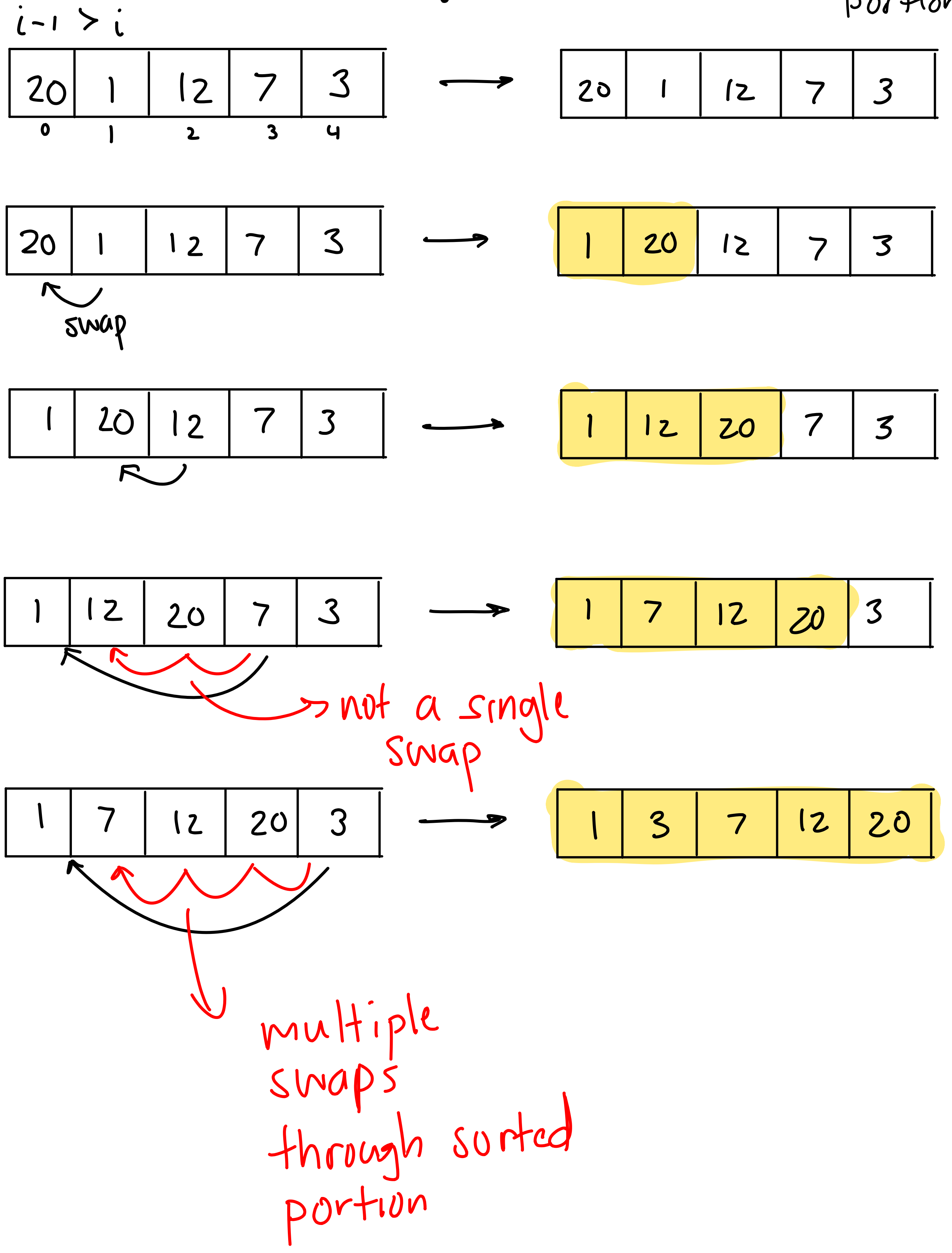
```
function containsX(string) {
  for(let i = 0; i < string.length; i++) {
    if (string[i] === "X") {
      return true;
    }
  }
  return false;
}
```

Best case: O(1) if the first character is 'X'
Average case (X is a random position in string): ~N/2 checks which is O(N) but fewer operations than scanning all of string (all of N).
Worst case: O(N)

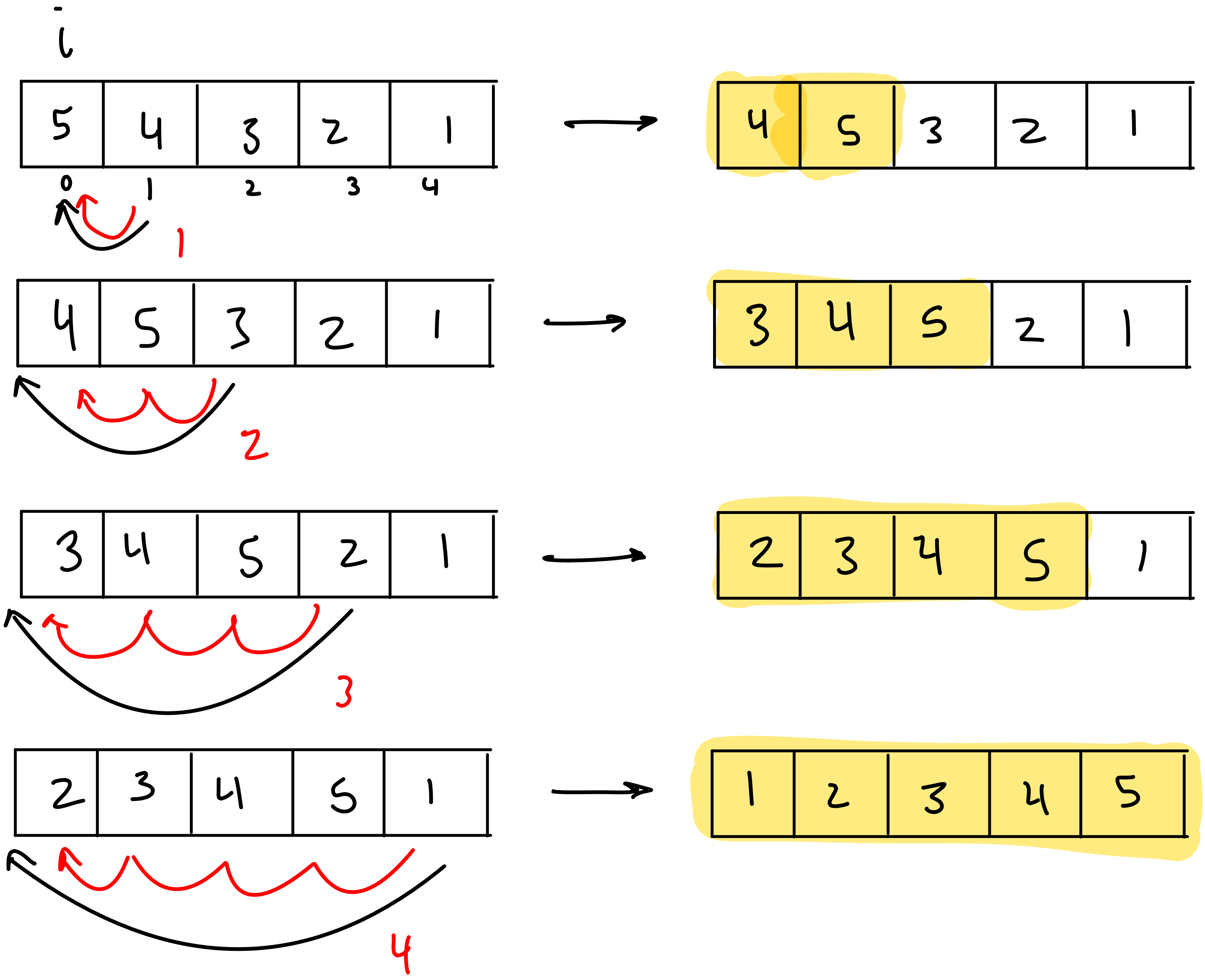
Video Link: https://youtu.be/_vlP8055n2M

Average Case

🟨 = sorted portion



Worst Case (n = 5)



Totals:

Comparisons: 1 + 2 + 3 + 4 = 10

Swaps: 1 + 2 + 3 + 4 = 10

Total operations: 10 + 10 = 20