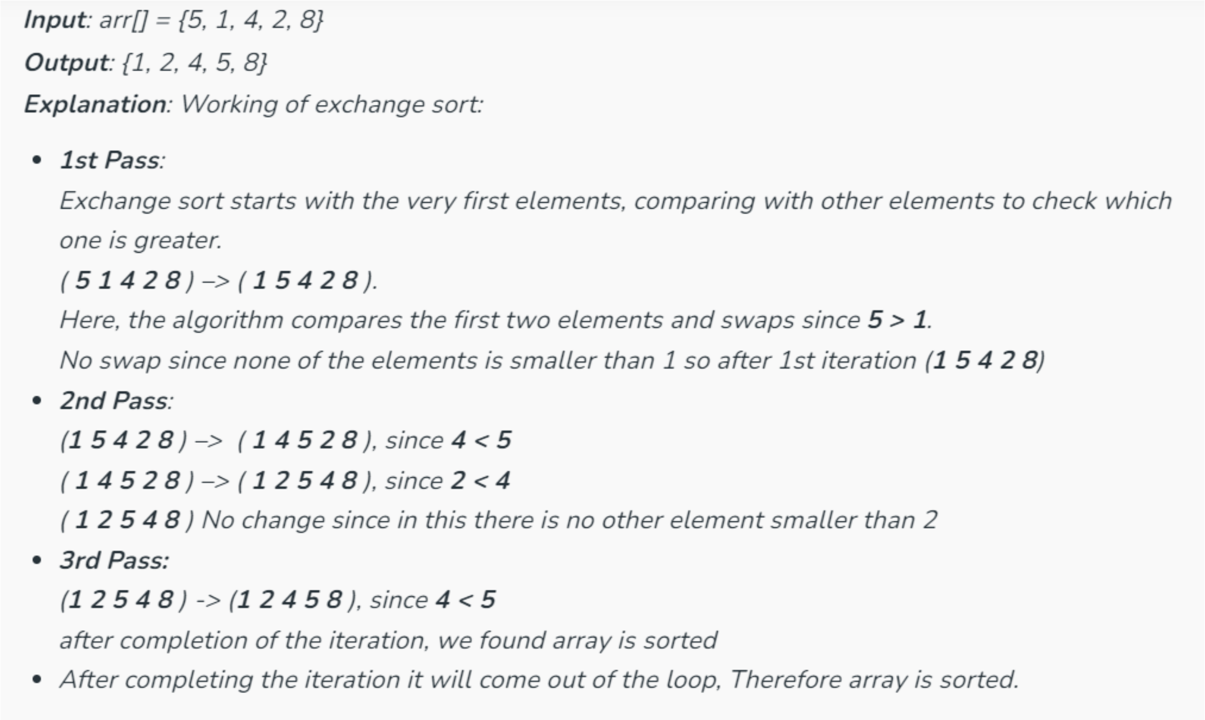
Exchange sort



Input: *arr[] = {5, 1, 4, 2, 8}*

Output: *{1, 2, 4, 5, 8}*

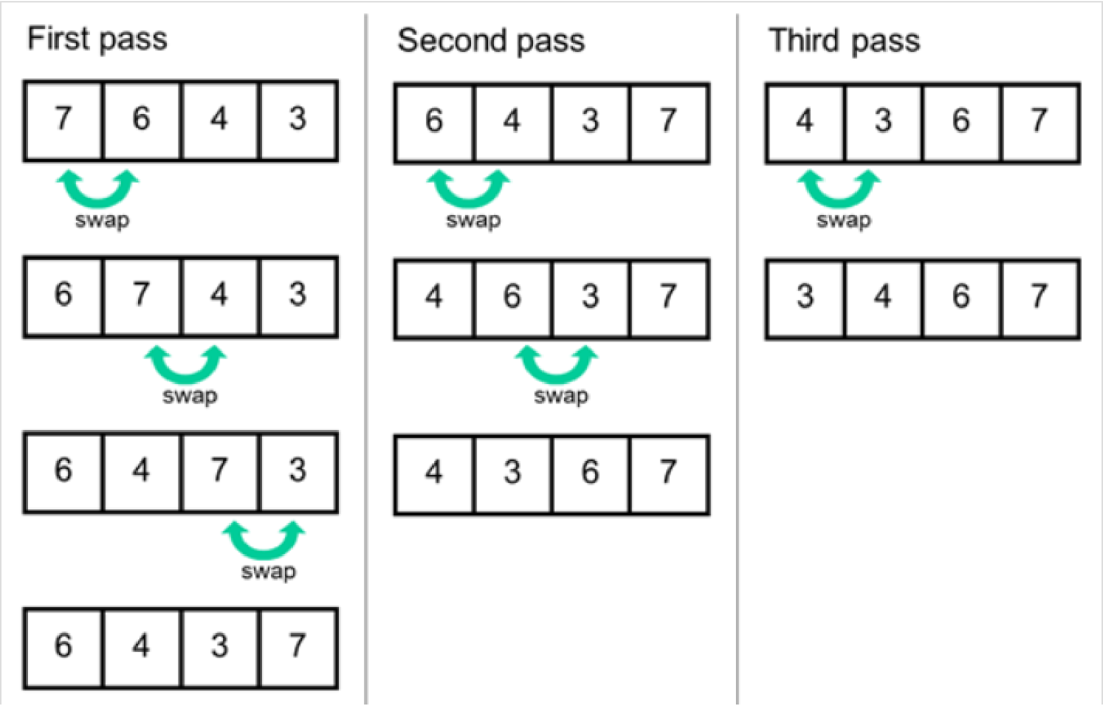
Explanation: *Working of exchange sort:*

- ***1st Pass***:
  Exchange sort starts with the very first elements, comparing with other elements to check which one is greater.
  *(**5 1 4 2 8**) –> (**1 5 4 2 8**).*
  *Here, the algorithm compares the first two elements and swaps since **5 > 1**.*
  *No swap since none of the elements is smaller than 1 so after 1st iteration (**1 5 4 2 8**)*
- ***2nd Pass***:
  *(**1 5 4 2 8**) –> (**1 4 5 2 8**), since **4 < 5***
  *(**1 4 5 2 8**) –> (**1 2 5 4 8**), since **2 < 4***
  *(**1 2 5 4 8**) No change since in this there is no other element smaller than 2*
- ***3rd Pass:***
  *(**1 2 5 4 8**) -> (**1 2 4 5 8**), since **4 < 5***
  after completion of the iteration, we found array is sorted
- *After completing the iteration it will come out of the loop, Therefore array is sorted.*

Bubble Sort

The bubble sort, also known as the ripple sort, is one of the least efficient sorting algorithms. However, it is probably the simplest to understand. At each step, if two adjacent elements of a list are not in order, they will be swapped. Thus, larger elements will “bubble” to the end, (or smaller elements will be “bubbled” to the front, depending on implementation) and hence the name.

The principle of a bubble sort is illustrated below: Compare the first two values and swap if necessary. Then compare the next pair of values and swap if necessary. This process is repeated n-1 times, where n is the number of values being sorted.


First pass
7 6 4 3
swap
6 7 4 3
swap
6 4 7 3
swap
6 4 3 7
Second pass
6 4 3 7
swap
4 6 3 7
swap
4 3 6 7
Third pass
4 3 6 7
swap
3 4 6 7