Reversing an Array

Reversing an array means reversing the order of elements in the given array.

Problem: Given an array of N elements. The task is to reverse the order of elements in the given array.

For Example

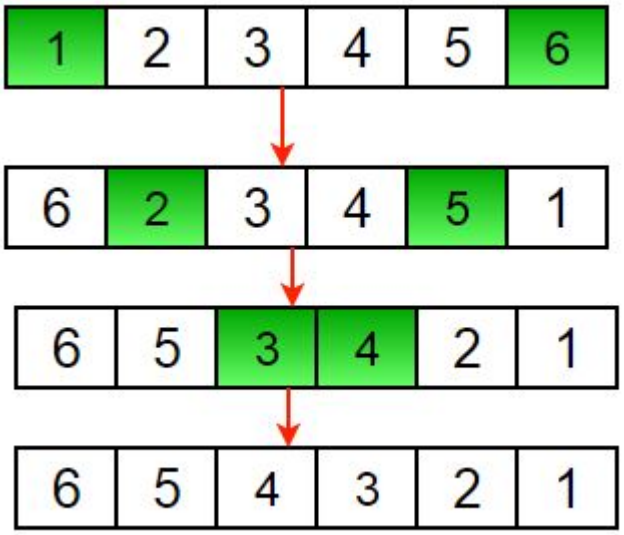
```
Input: arr[] = {1, 2, 3}
Output: arr[] = {3, 2, 1}

Input:  arr[] = {4, 5, 1, 2}
Output: arr[] = {2, 1, 5, 4}
```

Iterative Solution

- **Method 1 (Using Temporary Array)**: The idea is to first copy all of the elements of the given array in a temporary array. Then traverse the temporary array from end and replace elements in original array by elements of temp array. This method will take extra space in order of O(N).
- **Method 2 (Efficient)**: This method is efficient then the above method and avoids using extra spaces. The idea is to traverse the array from both ends and keep swapping elements from both ends until middle of the array is reached. For Example: **Time Complexity**: O(N), where N is the number of elements in the array.

  ```
  1) Initialize start and end indexes as
     start = 0, end = N-1
  2) In a loop, swap arr[start] with arr[end]
     and change start and end as follows :
         start = start + 1,
         end = end – 1
  ```



Recursive Solution

The recursive approach is almost similar to that of the method 2 of the iterative solution. Below is the recursive algorithm to reverse an array:

```
1) Initialize start and end indexes as
   start = 0, end = n-1
2) Swap arr[start] with arr[end]
3) Recursively call reverse for rest of the array.
```

Below is the recursive function to reverse an array:

```
void rvereseArray(arr[], start, end)
{
    if (start >= end)
        return;

    // Swap elements at start and end
    temp = arr[start];
    arr[start] = arr[end];
    arr[end] = temp;

    // Recursive Function calling
    rvereseArray(arr, start + 1, end - 1);
}
```

Time Complexity : O(N), where N is the number of elements in the array.