Day 85/180

Quick Sort 

---

1. Use a quick sort algorithm to sort elements in descending order

```
class Solution {
public:
    int partitionArray(vector<int> &nums, int low, int high) {
        if (low >= high) return -1;
        int pivot = low, l = pivot + 1, r = high;
        while (l <= r)
            if (nums[l] < nums[pivot]) l++;
            else if (nums[r] >= nums[pivot]) r--;
            else swap(nums[l], nums[r]);
        swap(nums[pivot], nums[r]);
        return r;
    }
    void quickSort(vector<int> &nums, int low, int high) {
        if (low >= high) return;
        swap(nums[low + rand() % (high - low + 1)], nums[low]);
        int pivot = partitionArray(nums, low, high);
        quickSort(nums, low, pivot);
        quickSort(nums, pivot + 1, high);
    }
    vector<int> sortArray(vector<int>& nums) {
        quickSort(nums,0,nums.size()-1);
        return nums;
    }
};
```

Code Explanation and Complexity of Aggressive Cows

1. The code defines two functions and has one predefined function that is the main method:

    - partitionArray: This method performs the partitioning step of the quicksort algorithm. It takes an array nums and two indices low and high as parameters and returns the index of the pivot element after partitioning.

    - quickSort: This method implements the quicksort algorithm recursively. It takes an array nums and two indices low and high as parameters and sorts the array in ascending order using the quicksort algorithm.

    - sortArray: This method is the entry point for sorting the array. It calls the quickSort method with the initial indices.

2. The partitioning in partitionArray is done using a scheme, where a pivot element is chosen and the elements less than the pivot are moved to its left, and elements greater than the pivot are moved to its right.

3. The quickSort method uses a random pivot selection strategy. It chooses a random element between low and high as the pivot and then partitions the array accordingly.

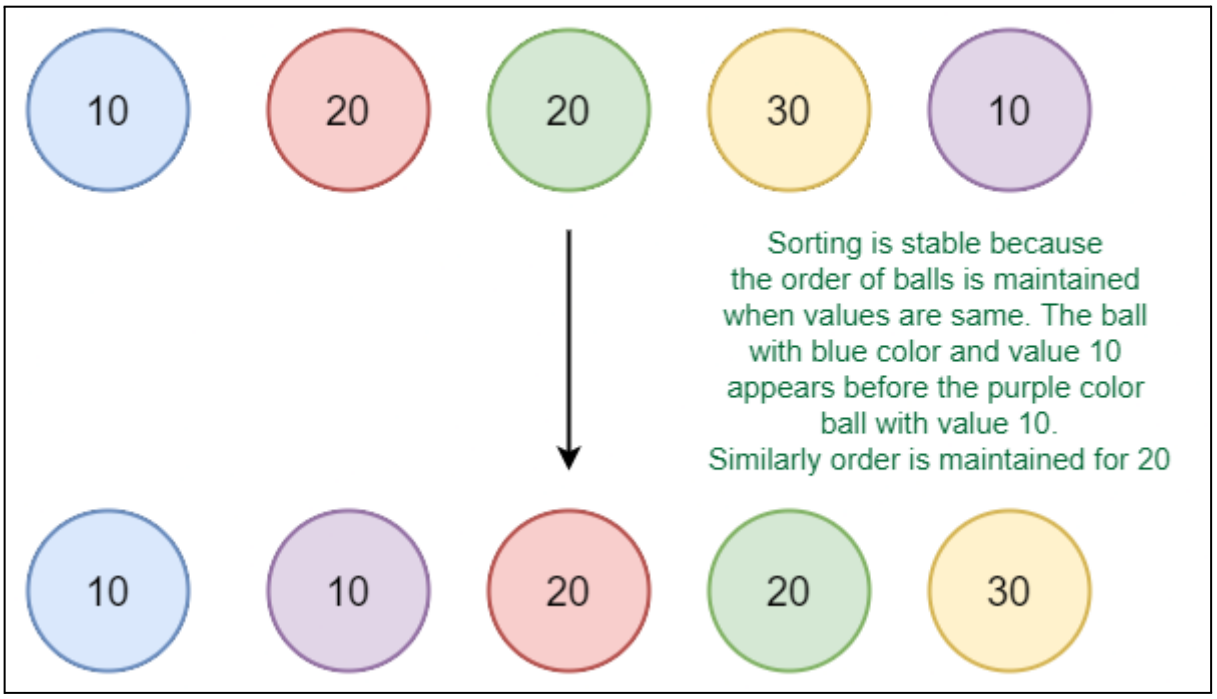
4. The sortArray method is the main method that initializes the quicksort process by calling quickSort with the entire array.

5. Time Complexity:
    - The average time complexity of quicksort is O(n log n), where n is the number of elements in the array.
    - The worst-case time complexity is O(n^2), but the random pivot selection strategy helps in achieving average-case performance.

6. Space Complexity:
    - The space complexity is O(log n) due to the recursive nature of the quicksort algorithm. This is the maximum depth of the call stack during the recursion.
    - Note: The code swaps elements in-place, minimizing the space usage compared to other sorting algorithms.

2. What is a stable algorithm? Find whether Bubble sort, Selection Sort, Insertion Sort, Merge Sort and Quicksort are stable algorithms.

A **stable sorting algorithm** is one where the **relative order of equal elements in the sorted output is the same as their relative order in the original input**. In other words, if there are two equal elements in the input array, and the one that appears first maintains its order with respect to the other in the sorted output, the algorithm is considered stable.



Example of stable sort

Bubble Sort:
Bubble Sort is stable. When two elements are equal, the algorithm does not swap them if they are in the correct order with respect to each other.

Selection Sort:
Selection Sort is not stable. It may change the relative order of equal elements during the selection process.

Insertion Sort:
Insertion Sort is generally stable. When equal elements are encountered, the algorithm does not change their relative order.

Merge Sort:
Merge Sort is stable. In the merge step, when elements are equal, the algorithm maintains their order as they appear in the input.

Quick Sort:

Quick Sort is not inherently stable. The partitioning step may change the relative order of equal elements. However, modifications can be made to the algorithm to make it stable, but this often comes at the cost of increased complexity.

In summary:
Stable: Bubble Sort, Insertion Sort, Merge Sort
Not Stable: Selection Sort, Quick Sort (without additional modifications)

3. What is an in-place algorithm? Find whether Bubble sort, Selection Sort, Insertion Sort, Merge Sort and Quicksort are in-place algorithms.

In-place has more than one definition. One strict definition is.

An **in-place algorithm** is an algorithm that **does not need an extra space** and produces an output in the same memory that contains the data by transforming the input 'in-place'. However, a small constant extra space used for variables is allowed.

A more broad definition is,

In-place means that the algorithm **does not use extra space** for manipulating the input but may require a small though non-constant extra space for its operation. Usually, **this space is O(log n)**, though sometimes anything in O(n) (Smaller than linear) is allowed.

Which Sorting Algorithms are In-Place and which are not?

In Place: Bubble sort, Selection Sort, Insertion Sort, Heapsort.

Not In-Place: Merge Sort. Note that merge sort requires O(n) extra space.

What about QuickSort? Why is it called In-Place?

QuickSort uses extra space for recursive function calls. It is called in-place according to broad definition as extra space required is not used to manipulate input, but only for recursive calls.

_____ END _____