Arrays: Leetcode

(Part 1)

Q4. Median of Two Sorted Arrays

```
class Solution {
public:
    double findMedianSortedArrays(vector<int>& A,
vector<int>& B) {
        if (A.size() > B.size()) return
findMedianSortedArrays(B, A);
        int m = A.size(), n = B.size();
        int low = 0, high = m;
        while (low <= high) {
            int i = (low + high) / 2;
            int j = (m + n + 1) / 2 - i;
            int Aleft = (i == 0 ? INT_MIN : A[i - 1]);
            int Aright = (i == m ? INT_MAX : A[i]);
            int Bleft = (j == 0 ? INT_MIN : B[j - 1]);
            int Bright = (j == n ? INT_MAX : B[j]);
            if (Aleft <= Bright && Bleft <= Aright) {
                if ((m + n) % 2)
                    return max(Aleft, Bleft);
                return (max(Aleft, Bleft) + min(Aright,
Bright)) / 2.0;
            }
            (Aleft > Bright) ? high = i - 1 : low = i + 1;
        }
        return 0.0;
    }
};
```

Q11. Container With Most Water

```
class Solution {
public:
    int maxArea(vector<int>& h) {
        int l = 0, r = h.size() - 1, area = 0;
        while (l < r) {
            area = max(area, min(h[l], h[r]) * (r - l));
            if (h[l] < h[r]) ++l; else --r;
        }
        return area;
    }
};
```

Q15. 3Sum

```
class Solution {
public:
    vector<vector<int>> threeSum(vector<int>& a) {
        sort(a.begin(), a.end());
        vector<vector<int>> res;
        for (int i = 0; i < a.size(); ++i) {
            if (i && a[i] == a[i - 1]) continue;
            int l = i + 1, r = a.size() - 1;
            while (l < r) {
                int s = a[i] + a[l] + a[r];
                if (s < 0) ++l;
                else if (s > 0) --r;
                else {
                    res.push_back({a[i], a[l], a[r]});
                    while (l < r && a[l] == a[l + 1]) ++l;
                    while (l < r && a[r] == a[r - 1]) --r;
                    ++l; --r;
                }
            }
        }
        return res;
    }
};
```

Q16. 3Sum Closest

```
class Solution {
public:
    int threeSumClosest(vector<int>& a, int target) {
        sort(a.begin(), a.end());
        int best = a[0] + a[1] + a[2];
        for (int i = 0; i < a.size(); ++i) {
            int l = i + 1, r = a.size() - 1;
            while (l < r) {
                int s = a[i] + a[l] + a[r];
                if (abs(s - target) < abs(best - target))
                    best = s;
                if (s < target) ++l;
                else --r;
            }
        }
        return best;
    }
};
```

Q18. 4Sum

```
class Solution {
public:
    vector<vector<int>> fourSum(vector<int>& a, int target)
{
        sort(a.begin(), a.end());
        vector<vector<int>> res;
        int n = a.size();
        for (int i = 0; i < n - 3; ++i) {
            if (i && a[i] == a[i - 1]) continue;
            for (int j = i + 1; j < n - 2; ++j) {
                if (j > i + 1 && a[j] == a[j - 1]) continue;
                int l = j + 1, r = n - 1;
                while (l < r) {
                    int s = a[i] + a[j] + a[l] + a[r];
                    if (s < target) ++l;
                    else if (s > target) --r;
                    else {
                        res.push_back({a[i], a[j], a[l],
a[r]});
                        while (l < r && a[l] == a[l + 1])
++l;
                        while (l < r && a[r] == a[r - 1])
--r;
                        ++l; --r;
                    }
                }
            }
        }
        return res;
```

```
        }
    };
```

Q26. Remove Duplicates from Sorted Array

```
class Solution {
public:
    int removeDuplicates(vector<int>& a) {
        int i = 0;
        for (int num : a) {
            if (i == 0 || num != a[i - 1])
                a[i++] = num;
        }
        return i;
    }
};
```

Q27. Remove Element

```
class Solution {
public:
    int removeElement(vector<int>& a, int val) {
        int i = 0;
        for (int num : a)
            if (num != val)
                a[i++] = num;
        return i;
    }
};
```

Q31. Next Permutation

```
class Solution {
public:
    void nextPermutation(vector<int>& a) {
        int i = a.size() - 2;
        while (i >= 0 && a[i] >= a[i + 1]) --i;
        if (i >= 0) {
            int j = a.size() - 1;
            while (a[j] <= a[i]) --j;
            swap(a[i], a[j]);
        }
        reverse(a.begin() + i + 1, a.end());
    }
};
```

Q33. Search in Rotated Sorted Array

```
class Solution {
public:
    int search(vector<int>& a, int target) {
        int l = 0, r = a.size() - 1;
        while (l <= r) {
            int mid = (l + r) / 2;
            if (a[mid] == target) return mid;
            if (a[l] <= a[mid]) {
                if (a[l] <= target && target < a[mid]) r =
mid - 1;
                else l = mid + 1;
            } else {
                if (a[mid] < target && target <= a[r]) l =
mid + 1;
                else r = mid - 1;
            }
        }
        return -1;
    }
};
```