WINTER WINNING CAMP DAY 5 QUESTIONS

Name: Nipun Chugh
UID: 22BCS10636
Class: KPIT - 901 / A

Q1. Given a sorted array of distinct integers and a target value, return the index if the target is found. If not, return the index where it would be if it were inserted in order. You must write an algorithm with O(log n) runtime complexity.
Q2. Left most and Right most index: Given a sorted array with possibly duplicate elements. The task is to find indexes of first and last occurrences of an element X in the given array.
Q3. Pair Sum Closet to 0: Given an integer array of N elements. You need to find the maximum sum of two elements such that sum is closest to zero.
Q4. Max Chunks To Make Sorted II: You are given an integer array arr.
We split arr into some number of chunks (i.e., partitions), and individually sort each chunk. After concatenating them, the result should equal the sorted array.
Q5. Sorted GCD Pair Queries: You are given an integer array nums of length n and an integer array queries.
Let gcdPairs denote an array obtained by calculating the GCD of all possible pairs (nums[i], nums[j]), where 0 <= i < j < n, and then sorting these values in ascending order. For each query queries[i], you need to find the element at index queries[i] in gcdPairs. Return an integer array answer, where answer[i] is the value at gcdPairs[queries[i]] for each query.

Solutions :

A1.

```
#include <iostream>
#include <vector>
using namespace std;

class Solution {
public:
    int searchInsert(vector<int>& nums, int target) {
        int left = 0, right = nums.size() - 1;

        while (left <= right) {
            int mid = left + (right - left) / 2;

            if (nums[mid] == target) {
                return mid; // Target found
            } else if (nums[mid] < target) {
                left = mid + 1; // Target is on the right side
            } else {
                right = mid - 1; // Target is on the left side
            }
        }

        // If target is not found, `left` will be the insertion position
        return left;
    }
};

int main() {
    Solution solution;

    // Test cases
```

```
    vector<int> nums1 = {1, 3, 5, 6};
    int target1 = 5;
    cout << "Input: nums = [1,3,5,6], target = 5 -> Output: " << solution.searchInsert(nums1, target1) << endl;

    vector<int> nums2 = {1, 3, 5, 6};
    int target2 = 2;
    cout << "Input: nums = [1,3,5,6], target = 2 -> Output: " << solution.searchInsert(nums2, target2) << endl;

    vector<int> nums3 = {1, 3, 5, 6};
    int target3 = 7;
    cout << "Input: nums = [1,3,5,6], target = 7 -> Output: " << solution.searchInsert(nums3, target3) << endl;

    return 0;
}
```

Output :

```
PS C:\Users\Lenovo\OneDrive - Chandigarh University\c++\WWC Codes> g++ .\VED5.cpp
PS C:\Users\Lenovo\OneDrive - Chandigarh University\c++\WWC Codes> .\a.exe
Input: nums = [1,3,5,6], target = 5 -> Output: 2
Input: nums = [1,3,5,6], target = 2 -> Output: 1
Input: nums = [1,3,5,6], target = 7 -> Output: 4
```

A2.

```
#include <iostream>
#include <vector>
using namespace std;

class Solution {
public:
    pair<int, int> findFirstAndLast(vector<int>& nums, int x) {
        int first = binarySearch(nums, x, true);
        int last = binarySearch(nums, x, false);
        return {first, last};
    }

private:
    // Helper function to perform binary search
    int binarySearch(vector<int>& nums, int x, bool findFirst) {
        int left = 0, right = nums.size() - 1, result = -1;

        while (left <= right) {
            int mid = left + (right - left) / 2;

            if (nums[mid] == x) {
                result = mid;
                if (findFirst) {
                    right = mid - 1; // Look for earlier occurrence
                } else {
                    left = mid + 1; // Look for later occurrence
                }
            } else if (nums[mid] < x) {
                left = mid + 1;
            } else {
                right = mid - 1;
            }
        }
```

```
        return result;
    }
};

int main() {
    Solution solution;

    // Example 1
    vector<int> nums1 = {1, 3, 5, 5, 5, 5, 67, 123, 125};
    int x1 = 5;
    auto result1 = solution.findFirstAndLast(nums1, x1);
    cout << "Input: v[] = {1, 3, 5, 5, 5, 5, 67, 123, 125}, X = 5\n";
    cout << "Output: " << result1.first << " " << result1.second << endl;

    // Example 2
    vector<int> nums2 = {1, 3, 5, 5, 5, 5, 7, 123, 125};
    int x2 = 7;
    auto result2 = solution.findFirstAndLast(nums2, x2);
    cout << "Input: v[] = {1, 3, 5, 5, 5, 5, 7, 123, 125}, X = 7\n";
    cout << "Output: " << result2.first << " " << result2.second << endl;

    return 0;
}
```

Output :

```
PS C:\Users\Lenovo\OneDrive - Chandigarh University\c++\WWC Codes> g++ .\ED5.cpp
PS C:\Users\Lenovo\OneDrive - Chandigarh University\c++\WWC Codes> .\a.exe
Input: v[] = {1, 3, 5, 5, 5, 5, 67, 123, 125}, X = 5
Output: 2 5
Input: v[] = {1, 3, 5, 5, 5, 5, 7, 123, 125}, X = 7
Output: 6 6
```

A3.

```
#include <iostream>
#include <vector>
#include <algorithm>
#include <climits>
using namespace std;

class Solution {
public:
    int closestToZero(vector<int>& arr, int n) {
        // Sort the array
        sort(arr.begin(), arr.end());

        int left = 0, right = n - 1;
        int closestSum = INT_MAX;
        int maxSumClosestToZero = INT_MIN;

        // Two-pointer approach
        while (left < right) {
            int sum = arr[left] + arr[right];

            // Update closest sum to zero
            if (abs(sum) < abs(closestSum)) {
```

```
            closestSum = sum;
            maxSumClosestToZero = sum;
        } else if (abs(sum) == abs(closestSum)) {
            // If the sums are equally close to zero, choose the maximum sum
            maxSumClosestToZero = max(maxSumClosestToZero, sum);
        }

        // Move pointers
        if (sum < 0) {
            left++; // Increase the sum by moving left pointer
        } else {
            right--; // Decrease the sum by moving right pointer
        }
    }

    return maxSumClosestToZero;
  }
};

int main() {
  Solution solution;

  // Example 1
  vector<int> arr1 = {-8, -66, -60};
  int n1 = arr1.size();
  cout << "Input: arr[] = {-8, -66, -60}\n";
  cout << "Output: " << solution.closestToZero(arr1, n1) << endl;

  // Example 2
  vector<int> arr2 = {-21, -67, -37, -18, 4, -65};
  int n2 = arr2.size();
  cout << "Input: arr[] = {-21, -67, -37, -18, 4, -65}\n";
  cout << "Output: " << solution.closestToZero(arr2, n2) << endl;

  return 0;
}
```

Output :

```
PS C:\Users\Lenovo\OneDrive - Chandigarh University\c++\WWC Codes> g++ .\MD5.cpp
PS C:\Users\Lenovo\OneDrive - Chandigarh University\c++\WWC Codes> .\a.exe
Input: arr[] = {-8, -66, -60}
Output: -68
Input: arr[] = {-21, -67, -37, -18, 4, -65}
Output: -14
```

A4.

```
#include <iostream>
#include <vector>
#include <algorithm>
using namespace std;

class Solution {
public:
  int maxChunksToSorted(vector<int>& arr) {
```

```
        int n = arr.size();

        // Initialize two arrays for tracking max from the left and min from the right
        vector<int> leftMax(n), rightMin(n);

        // Fill leftMax array
        leftMax[0] = arr[0];
        for (int i = 1; i < n; i++) {
            leftMax[i] = max(leftMax[i - 1], arr[i]);
        }

        // Fill rightMin array
        rightMin[n - 1] = arr[n - 1];
        for (int i = n - 2; i >= 0; i--) {
            rightMin[i] = min(rightMin[i + 1], arr[i]);
        }

        // Calculate the maximum number of chunks
        int chunks = 0;
        for (int i = 0; i < n - 1; i++) {
            if (leftMax[i] <= rightMin[i + 1]) {
                chunks++;
            }
        }

        // Include the final chunk
        return chunks + 1;
    }
};

int main() {
    Solution solution;

    // Example 1
    vector<int> arr1 = {5, 4, 3, 2, 1};
    cout << "Input: arr = [5, 4, 3, 2, 1]\n";
    cout << "Output: " << solution.maxChunksToSorted(arr1) << endl;

    // Example 2
    vector<int> arr2 = {2, 1, 3, 4, 4};
    cout << "Input: arr = [2, 1, 3, 4, 4]\n";
    cout << "Output: " << solution.maxChunksToSorted(arr2) << endl;

    return 0;
}
```

Output :

```
PS C:\Users\Lenovo\OneDrive - Chandigarh University\c++\WWC Codes> g++ .\HD5.cpp
PS C:\Users\Lenovo\OneDrive - Chandigarh University\c++\WWC Codes> .\a.exe
Input: arr = [5, 4, 3, 2, 1]
Output: 1
Input: arr = [2, 1, 3, 4, 4]
Output: 4
```

A5.

```
#include <iostream>
#include <vector>
#include <algorithm>
using namespace std;

// Custom GCD function
int customGCD(int a, int b) {
    while (b != 0) {
        int temp = b;
        b = a % b;
        a = temp;
    }
    return a;
}

class Solution {
public:
    vector<int> gcdQueries(vector<int>& nums, vector<int>& queries) {
        vector<int> gcdPairs;
        int n = nums.size();

        // Compute all GCD pairs
        for (int i = 0; i < n; i++) {
            for (int j = i + 1; j < n; j++) {
                gcdPairs.push_back(customGCD(nums[i], nums[j]));
            }
        }

        // Sort the GCD pairs in ascending order
        sort(gcdPairs.begin(), gcdPairs.end());

        // Prepare the answer for each query
        vector<int> result;
        for (int q : queries) {
            if (q >= 0 && q < gcdPairs.size()) {
                result.push_back(gcdPairs[q]);
            } else {
                result.push_back(-1); // Placeholder for out-of-bounds queries
            }
        }

        return result;
    }
};

int main() {
    Solution solution;

    // Example 1
    vector<int> nums1 = {2, 3, 4};
    vector<int> queries1 = {0, 2, 2};
    vector<int> result1 = solution.gcdQueries(nums1, queries1);
    cout << "Output: ";
    for (int r : result1) cout << r << " ";
    cout << endl;
```

```
    // Example 2
    vector<int> nums2 = {4, 4, 2, 1};
    vector<int> queries2 = {5, 3, 1, 0};
    vector<int> result2 = solution.gcdQueries(nums2, queries2);
    cout << "Output: ";
    for (int r : result2) cout << r << " ";
    cout << endl;

    // Example 3
    vector<int> nums3 = {2, 2};
    vector<int> queries3 = {0, 0};
    vector<int> result3 = solution.gcdQueries(nums3, queries3);
    cout << "Output: ";
    for (int r : result3) cout << r << " ";
    cout << endl;

    return 0;
}
```

Output :