



Name- Kuldeep

UID- 22BCS10071

Section- KPIT_901/A

Searching and Sorting QUESTIONS :-

1.Searching a Number

Given an integer k and array arr . Your task is to return the position of the first occurrence of k in the given array and if element k is not present in the array then return -1.

Note: 1-based indexing is followed here.

Example1:

Input: $k = 16$, $arr = [9, 7, 16, 16, 4]$

Output: 3

Explanation: The value 16 is found in the given array at positions 3 and 4, with position 3 being the first occurrence.

Example2:

Input: $k=98$, $arr = [1, 22, 57, 47, 34, 18, 66]$

Output: -1

Example2:

Input: k=9 , arr = [1, 22, 57, 47, 34, 9, 66]

Output: 6

Explanation: k = 98 isn't found in the given array.

Expected Time Complexity: O(n)

Expected Auxiliary Space: O(1)

Constraints:

- $1 \leq \text{arr.size} \leq 10^6$
- $1 \leq \text{arr}[i] \leq 10^9$
- $1 \leq k \leq 10^6$

Reference.: <https://www.geeksforgeeks.org/problems/searching-a-number0324/1>

Code:-

```
#include <iostream>

#include <vector>

using namespace std;

int searchNumber(int k, const vector<int>& arr) {

    for (int i = 0; i < arr.size(); i++) {

        if (arr[i] == k) {

            return i + 1;

        }

    }

}
```

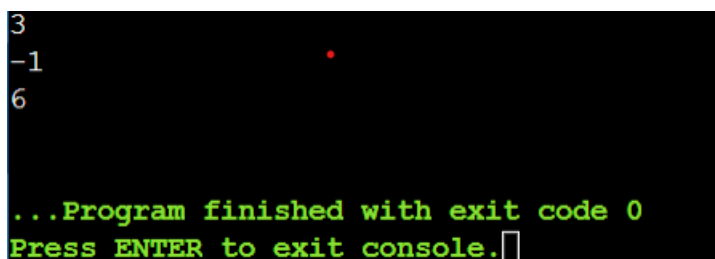
```

    return -1;
}

int main() {
    int k1 = 16;
    vector<int> arr1 = {9, 7, 16, 16, 4};
    cout << searchNumber(k1, arr1) << endl;
    int k2 = 98;
    vector<int> arr2 = {1, 22, 57, 47, 34, 18, 66};
    cout << searchNumber(k2, arr2) << endl;
    int k3 = 9;
    vector<int> arr3 = {1, 22, 57, 47, 34, 9, 66};
    cout << searchNumber(k3, arr3) << endl;
    return 0;
}

```

Output:-



```

3
-1
6
...Program finished with exit code 0
Press ENTER to exit console.

```

6. Minimum Number of Moves to Seat Everyone

There are n available seats and n students standing in a room. You are given an array `seats` of length n , where `seats[i]` is the position of the i th seat. You are also given the array `students` of length n , where `students[j]` is the position of the j th student.

You may perform the following move any number of times:

Increase or decrease the position of the i th student by 1 (i.e., moving the i th student from position x to $x + 1$ or $x - 1$)

Return the minimum number of moves required to move each student to a seat such that no two students are in the same seat.

Note that there may be multiple seats or students in the same position at the beginning.

Example 1:

Input: seats = [3,1,5], students = [2,7,4]

Output: 4

Explanation: The students are moved as follows:

- The first student is moved from position 2 to position 1 using 1 move.
 - The second student is moved from position 7 to position 5 using 2 moves.
 - The third student is moved from position 4 to position 3 using 1 move.
- In total, $1 + 2 + 1 = 4$ moves were used.

Example 2:

Input: seats = [4,1,5,9], students = [1,3,2,6]

Output: 7

Explanation: The students are moved as follows:

- The first student is not moved.
- The second student is moved from position 3 to position 4 using 1 move.
- The third student is moved from position 2 to position 5 using 3 moves.
- The fourth student is moved from position 6 to position 9 using 3 moves.

In total, $0 + 1 + 3 + 3 = 7$ moves were used.

Ø **Reference:** <https://leetcode.com/problems/minimum-number-of-moves-to-seat-everyone/description/>

Code:-

```
#include <iostream>
```

```
#include <vector>
```

```
#include <algorithm>

using namespace std;

int minMovesToSeat(vector<int>& seats, vector<int>& students) {

    sort(seats.begin(), seats.end());

    sort(students.begin(), students.end());

    int moves = 0;

    for (int i = 0; i < seats.size(); i++) {

        moves += abs(seats[i] - students[i]);

    }

    return moves;

}

int main() {

    vector<int> seats1 = {3, 1, 5};

    vector<int> students1 = {2, 7, 4};

    cout << minMovesToSeat(seats1, students1) << endl;

    vector<int> seats2 = {4, 1, 5, 9};

    vector<int> students2 = {1, 3, 2, 6};

    cout << minMovesToSeat(seats2, students2) << endl;

    return 0;

}
```

Output:-

```
4
7

...Program finished with exit code 0
Press ENTER to exit console.□
```

Medium (Questions 11–15)

12. Find First and Last Position of Element in Sorted Array.

Given an array of integers `nums` sorted in non-decreasing order, find the starting and ending position of a given target value.

If target is not found in the array, return `[-1, -1]`.

You must write an algorithm with $O(\log n)$ runtime complexity.

Example 1:

Input: `nums = [5,7,7,8,8,10]`, `target = 8`

Output: `[3,4]`

Example 2:

Input: `nums = [5,7,7,8,8,10]`, `target = 6`

Output: `[-1,-1]`

Example 3:

Input: `nums = []`, `target = 0`

Output: `[-1,-1]`

Constraints:

$0 \leq \text{nums.length} \leq 10^5$

$-10^9 \leq \text{nums}[i] \leq 10^9$

nums is a non-decreasing array.

$-10^9 \leq \text{target} \leq 10^9$

Ø **Reference:** <https://leetcode.com/problems/find-first-and-last-position-of-element-in-sorted-array/description/>

Code:-

```
#include <iostream>
```

```
#include <vector>
```

```
using namespace std;
```

```
vector<int> searchRange(vector<int>& nums, int target) {
```

```
    int left = -1, right = -1;
```

```
    int low = 0, high = nums.size() - 1;
```

```
    while (low <= high) {
```

```
        int mid = low + (high - low) / 2;
```

```
        if (nums[mid] >= target) high = mid - 1;
```

```
        else low = mid + 1;
```

```
    }
```

```
    if (low < nums.size() && nums[low] == target) left = low;
```

```
    low = 0, high = nums.size() - 1;
```

```
    while (low <= high) {
```

```

    int mid = low + (high - low) / 2;

    if (nums[mid] <= target) low = mid + 1;

    else high = mid - 1;

}

if (high >= 0 && nums[high] == target) right = high;

return {left, right};

}

int main() {

    vector<int> nums1 = {5, 7, 7, 8, 8, 10};

    int target1 = 8;

    vector<int> res1 = searchRange(nums1, target1);

    cout << "[" << res1[0] << ", " << res1[1] << "]" << endl;

    vector<int> nums2 = {5, 7, 7, 8, 8, 10};

    int target2 = 6;

    vector<int> res2 = searchRange(nums2, target2);

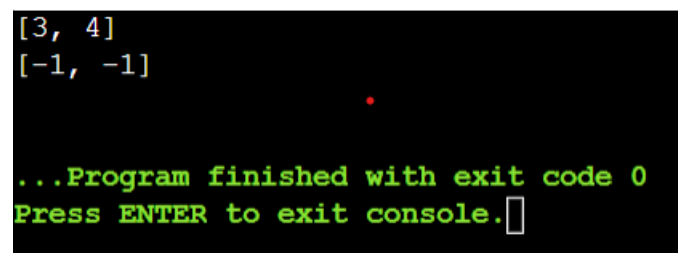
    cout << "[" << res2[0] << ", " << res2[1] << "]" << endl;

    return 0;

}

```

Output:-



```

[3, 4]
[-1, -1]

...Program finished with exit code 0
Press ENTER to exit console.

```


Hard (Questions 16–20)

15. Find the Kth Smallest Sum of a Matrix With Sorted Rows.

You are given an $m \times n$ matrix `mat` that has its rows sorted in non-decreasing order and an integer `k`.

You are allowed to choose exactly one element from each row to form an array.

Return the k th smallest array sum among all possible arrays.

Example 1:

Input: `mat = [[1,3,11],[2,4,6]]`, `k = 5`

Output: 7

Explanation: Choosing one element from each row, the first k smallest sum are:

[1,2], [1,4], [3,2], [3,4], [1,6]. Where the 5th sum is 7.

Example 2:

Input: `mat = [[1,3,11],[2,4,6]]`, `k = 9`

Output: 17

Example 3:

Input: `mat = [[1,10,10],[1,4,5],[2,3,6]]`, `k = 7`

Output: 9

Explanation: Choosing one element from each row, the first k smallest sum are:

[1,1,2], [1,1,3], [1,4,2], [1,4,3], [1,1,6], [1,5,2], [1,5,3]. Where the 7th sum is 9.

Constraints:

- `m == mat.length`
- `n == mat.length[i]`
- `1 <= m, n <= 40`

- $1 \leq \text{mat}[i][j] \leq 5000$
- $1 \leq k \leq \min(200, n^m)$
- $\text{mat}[i]$ is a non-decreasing array.

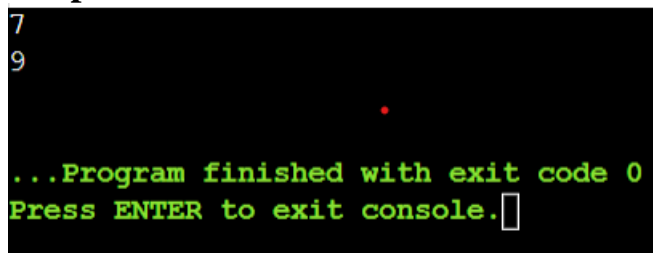
Reference:-

<https://leetcode.com/problems/find-the-kth-smallest-sum-of-a-matrix-with-sorted-rows/description/>

Code:-

```
#include <iostream>
#include <vector>
#include <queue>
#include <algorithm>
using namespace std;
int kthSmallestSum(vector<vector<int>>& mat, int k) {
    priority_queue<int> pq;
    pq.push(0);
    for (auto& row : mat) {
        priority_queue<int> nextPq;
        while (!pq.empty()) {
            int sum = pq.top(); pq.pop();
            for (int num : row) {
                nextPq.push(sum + num);
                if (nextPq.size() > k) nextPq.pop();
            }
        }
        swap(pq, nextPq);
    }
    return pq.top();
}
int main() {
    vector<vector<int>> mat1 = {{1, 3, 11}, {2, 4, 6}};
    int k1 = 5;
    cout << kthSmallestSum(mat1, k1) << endl;
    vector<vector<int>> mat2 = {{1, 10, 10}, {1, 4, 5}, {2, 3, 6}};
    int k2 = 7;
    cout << kthSmallestSum(mat2, k2) << endl;
    return 0;
}
```

Output:-



```
7
9
...Program finished with exit code 0
Press ENTER to exit console.
```

Very Hard (Questions 21–25)

21.Find Minimum in Rotated Sorted Array II.

Suppose an array of length n sorted in ascending order is rotated between 1 and n times. For example, the array `nums = [0,1,4,4,5,6,7]` might become:

`[4,5,6,7,0,1,4]` if it was rotated 4 times.

`[0,1,4,4,5,6,7]` if it was rotated 7 times.

Notice that rotating an array `[a[0], a[1], a[2], ..., a[n-1]]` 1 time results in the array `[a[n-1], a[0], a[1], a[2], ..., a[n-2]]`.

Given the sorted rotated array `nums` that may contain duplicates, return the minimum element of this array.

You must decrease the overall operation steps as much as possible.

Example 1:

Input: `nums = [1,3,5]`

Output: 1

Example 2:

Input: `nums = [2,2,2,0,1]`

Output: 0

Constraints:

- $n == \text{nums.length}$
- $1 \leq n \leq 5000$
- $-5000 \leq \text{nums}[i] \leq 5000$
- `nums` is sorted and rotated between 1 and n times.

Reference:-

<https://leetcode.com/problems/find-minimum-in-rotated-sorted-array-ii/description/>

Code:-

```
#include <iostream>

#include <vector>

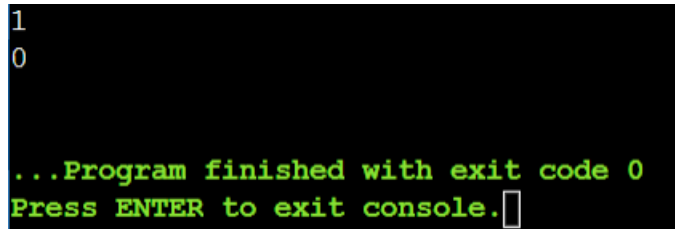
using namespace std;

int findMin(vector<int>& nums) {
    int low = 0, high = nums.size() - 1;
    while (low < high) {
        int mid = low + (high - low) / 2;
        if (nums[mid] > nums[high]) {
            low = mid + 1;
        } else if (nums[mid] < nums[high]) {
            high = mid;
        } else {
            high--;
        }
    }
    return nums[low];
}

int main() {
    vector<int> nums1 = {1, 3, 5};
    cout << findMin(nums1) << endl;
    vector<int> nums2 = {2, 2, 2, 0, 1};
```

```
cout << findMin(nums2) << endl;  
return 0;  
}
```

Output:-



```
1  
0  
  
...Program finished with exit code 0  
Press ENTER to exit console.█
```