Experiment5.1

StudentName: Arushi Rana
UID:22BCS12899
Branch:BE-CSE
Section/Group: IoT_641(A)
Semester:6th
DateofPerformance:28/2/25
SubjectName: AdvancedProgramming-2
SubjectCode:22CSP-351

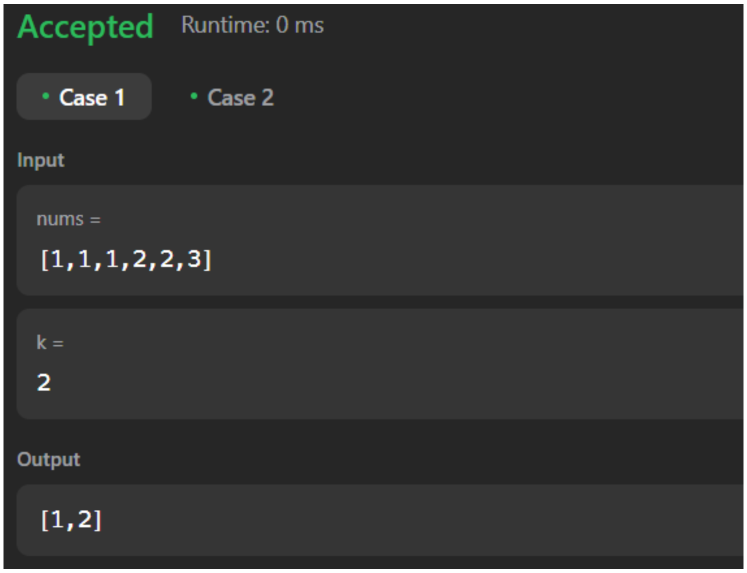
1. **Aim:** Find the k most frequent elements in an array. Use a hash map to count frequencies,thenuseamin-heaporbucketsorttoextractthetopkelements.The heap approach runs in O(n log k) time. Edge cases include k = 1 or all elements being unique.
2. **Objective:** Implement an algorithm to identify and return the k most frequently occurring elements within a given array.
3. **Implementation/Code:**

```
classSolution{ public:
    vector<int>topKFrequent(vector<int>&nums,intk){
        unordered_map<int,int> mp;
        int n=nums.size();
        for(inti=0;i<n;i++){
            mp[nums[i]]++;
        }
        vector<pair<int,int>>freq(mp.begin(),mp.end());
        sort(freq.begin(),freq.end(),[](auto&a,auto&b){ return
        a.second >b.second;
    });
        vector<int>ans;

        for(int i=0;i<k;i++){
            ans.push_back(freq[i].first);
        }

        returnans;
    }
};
```

4. Output



5. LearningOutcomes

- Understandtheconceptof hashmapsandmin-heaps.
- Learnthowtoimplementthealgorithminaprogramminglanguage.
- Learnt how to evaluate and compare the min-heap and bucket sort approaches for top-k extraction.
- Understandtheimportanceofcodeefficiency,andhowtocreatecodethatruns within the desired time constraints.

<u>Experiment5.2</u>

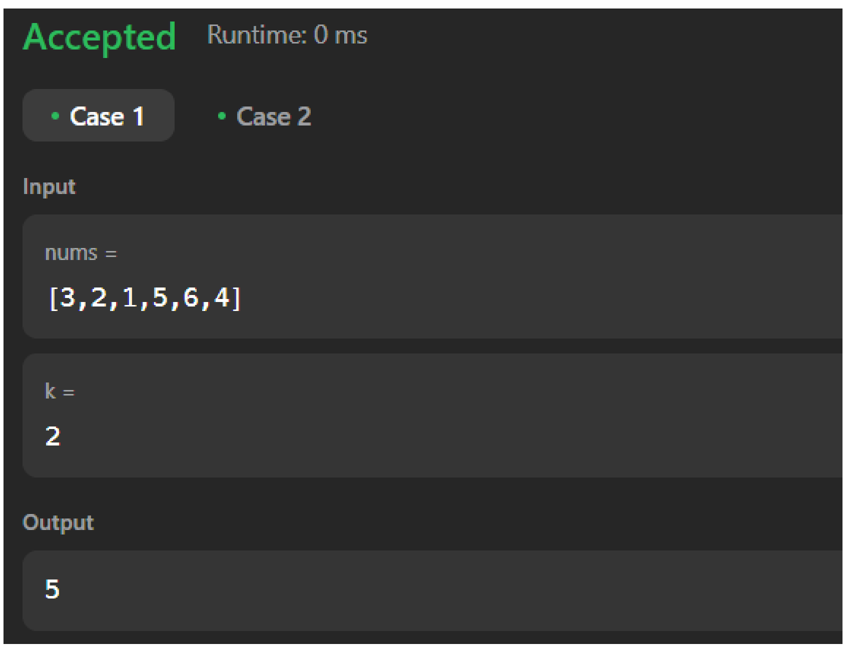
1. **Aim**:Findthekthlargestelementinanarrayusingamin-heaporQuickselect. The heap approach runs in O(n log k) time, while Quickselect runs in O(n) on average. Edge cases include k = 1 (max element) and k = n (min element).
2. **Objective**:Implementalgorithmstoefficientlyfindthekthlargestelement within a given array.

3. **Code**:

```
classSolution{public:
intfindKthLargest(vector<int>&nums,intk){
```

```
        priority_queue<int>max;
        inta=1;
       for(autop:nums){
           max.push(p);
       }
       for(inti=0;i<nums.size();i++){
           if(a<k){
               max.pop();
               a++;
           }
       }
       returnmax.top();
}
};
```

4. Output



5. LearningOutcomes:

- Understandthestepsinvolvedinboththemin-heapandQuickselect algorithms for finding the kth largest element.
- Learnhowtoanalyzeandcomparethetimecomplexityofthemin-heap (O(n log k)) and Quickselect (O(n) average) approaches.
- Implementboththemin-heapandQuickselectalgorithmsina programming language.

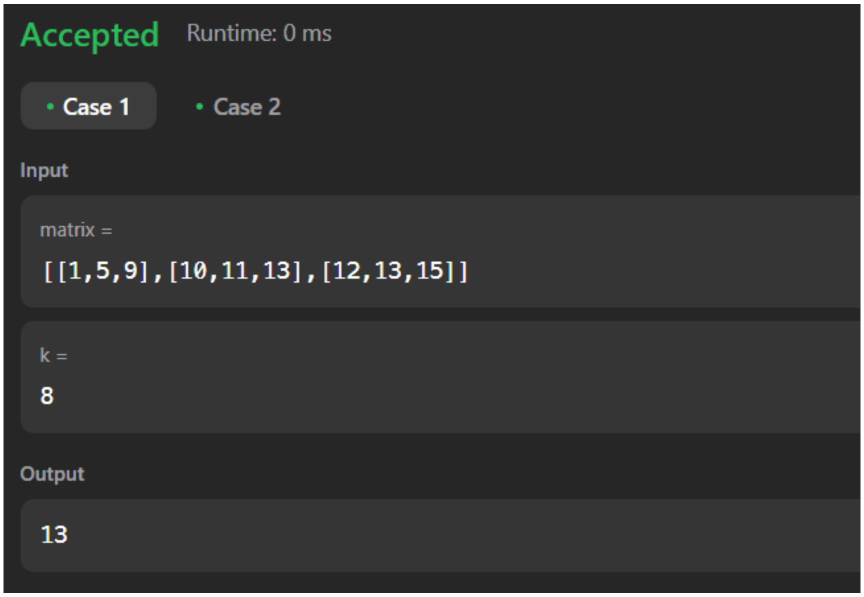
Experiment5.3

1. **Aim**:Findthekthsmallestelementinarow-andcolumn-sortedmatrix.Usea min-heap (O(k log n)) or binary search on values (O(n log max-min)). Edge cases include k = 1 (smallest element) and k = n^2 (largest element).
2. **Objective**:Implementanalgorithmtofindthekthsmallestelementusing binary search on values.

3. **Code**:

```
classSolution
{
public:
    intkthSmallest(vector<vector<int>>&matrix,intk)
    {
        intn=matrix.size();
        intle=matrix[0][0],ri=matrix[n-1][n-1]; int mid =
        0;
        while(le< ri)
        {
            mid=le+(ri-le)/2; int
            num = 0;
            for(inti=0;i<n;i++)
            {
                intpos=upper_bound(matrix[i].begin(),matrix[i].end(),mid)-
matrix[i].begin();
                num+=pos;
            }
            if(num< k)
            {
                le=mid+1;
            }
            else
            {
                ri=mid;
            }
        }
        returnle;
    }
};
```

4. **Output**



5. **LearningOutcomes:**
   - Understandthestepsinvolvedinboththemin-heapandbinarysearch algorithms.
   - Learnhowtoanalyzeandcomparethetimecomplexity.
   - Implementappropriatealgorithmbasedontheproblem'sconstraints.