



Experiment No. 2

Title: Implementation of problem on competitive programming platform and optimizing it using array and hash table data structures



Batch:A2**Roll No: 1601421063****Experiment No.:2**

Aim: To implement a problem on hacker earth and optimize it's solution using array and hash table data structures

Resources needed: Web Browser to access Hackerearth platform

Theory:

Competitive Programming involves solving coding problems using data structures and algorithms. Competitive programming helps to improve logical and analytical skills. There are various platforms available for Competitive Programming such as Hackerearth, Codechef, Codeforces etc. On Competitive Programming platform, one needs to write solution under various restrictions such as memory limits, execution time, constraints on input size and so on. The Competitive Programming Platform then evaluates the solution against pre-defined test cases for a problem statement.

Hackerearth is a competitive programming platform which hosts programming challenges and coding competitions. On Hackerearth platform, the problem statement is defined in terms of real world scenario followed by input format, output format, constraints, sample input, sample output, time limit in seconds and memory limit. Hackerearth supports several programming languages such as C, C++, Java, Python, JavaScript and so on. Any of these supported programming languages can be selected for the implementation of the solution. There is a code editor provided where the code can be written and compiled. When the solution is submitted on the hackerearth platform, it is tested against pre-defined test cases for the problem statement. And it displays the list of all test cases and the status of solution against each test case whether solution has passed that test case or not.

Solving problem on Competitive Programming Platform like hackerearth helps to:

1. Understand problem in terms of real-world scenario
2. Interpret input, output and processing information and constraints from the given real world scenario problem
3. Develop logical and analytical ability for optimization of solution
4. Understand and perform test-case based evaluation of solution

Activity:

Consider the following problem statement. Solve it using array and hash table data structures on hackerearth platform

There are N numbers $A_1, A_2, A_3, \dots, A_N$, and you are given Q queries. In each query, you are given two integers L and R.

You are required to print the sum of all the numbers whose frequency of occurrence is between L and R (including L and R). Print a single integer for each query in a new line.

Input format

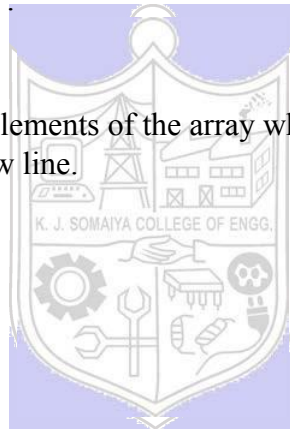
- The first line contains N denoting the size of the array.
- The second line contains N integers denoting the elements of the array.
- The third line contains Q denoting the number of queries.
- Next Q lines contain L and R.

Output format

For each query, print the sum of all elements of the array whose frequency of occurrence is between L and R (inclusive) in a new line.

Constraints

$$1 \leq N \leq 10^6$$
$$1 \leq A_i \leq 10^6$$
$$1 \leq Q \leq 10^6$$
$$1 \leq l \leq r \leq N$$

**Sample Input**

```
8
4 4 6 5 3 3 3 9
4
1 4
2 7
3 7
5 6
```

Sample Output

```
37
17
9
0
```

Solve the given problem statement on hackerearth on the following link:

<https://www.hackerearth.com/practice/data-structures/arrays/1-d/practice-problems/algorithm/su>

m-as-per-frequency-88b00c1f/

Program:

```
#include <bits/stdc++.h>
#define int long long
using namespace std;





































int32_t main() {
    ios_base::sync_with_stdio(false);
    cin.tie(NULL);
    int n;
    cin>>n;
    map<int,int> m;
    for(int i=0;i<n;i++){
        int x;
        cin>>x;
        m[x]++;
    }
    vector<int> v2(n+1,0);
    for(auto itr=m.begin();itr!=m.end();itr++){
        v2[itr->second]+=itr->first*itr->second;
    }

    for(int i=1;i<n+1;i++){
        v2[i]+=v2[i-1];
    }
    int q;
    cin>>q;
    while(q--){
        int x,y;
        cin>>x>>y;
        cout<<v2[y]-v2[x-1]<<"\n";
    }

    return 0;
}
```

Output:**RESULT:**  Accepted[? Refer judge environment](#)

Score	Time (sec)	Memory (KiB)	Language
30	3.59564	47796	C++17

Input	Result	Time (sec)	Memory (KiB)	Score	Your Output	Correct Output	Diff
Input #1	 Accepted	0.009557	2	10			
Input #2	 Accepted	0.099118	512	10			
Input #3	 Accepted	0.642044	47796	10			
Input #4	 Accepted	0.041831	4240	10			
Input #5	 Accepted	0.050177	5740	10			
Input #6	 Accepted	0.768296	38712	10			
Input #7	 Accepted	0.316913	26916	10			
Input #8	 Accepted	0.133844	8740	10			
Input #9	 Accepted	0.851837	44152	10			
Input #10	 Accepted	0.682028	35928	10			

Custom Input

```
8
4 4 6 5 3 3 3 9
4
1 4
2 7
3 7
5 6
```

Status Successfully executed Date 2023-02-01 05:37:59 Time 0.009081 sec Mem 5468 kB

Input

```
8
4 4 6 5 3 3 3 9
4
1 4
2 7
3 7
5 6
```

Output

```
37
17
9
0
```



Test Result:

Custom Input

```
8
4 6 6 5 3 2 6 9
4
1 4
2 7
3 7
5 6
```

Status Successfully executed Date 2023-02-01 05:57:42 Time 0.007332 sec Mem 5448 kB

Input

```
8
4 6 6 5 3 2 6 9
4
1 4
2 7
3 7
5 6
```

Output

```
41
18
18
0
```

Custom Input

```
10
4 6 6 5 3 2 6 9 12 6
4
1 4
2 7
3 7
5 6
```

Status Successfully executed **Date** 2023-02-01 06:00:31 **Time** 0.011276 sec **Mem** 5452 kB**Input**

```
10
4 6 6 5 3 2 6 9 12 6
4
1 4
2 7
3 7
5 6
```

Output

```
59
24
24
0
```

11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100

Custom Input

```
10
4 6 6 5 3 2 6 9 12 6
4
1 8
2 4
3 9
5 6
```

Status Successfully executed **Date** 2023-02-01 06:01:05 **Time** 0.007711 sec **Mem** 5448 kB**Input**

```
10
4 6 6 5 3 2 6 9 12 6
4
1 8
2 4
3 9
5 6
```

Output

```
59
24
24
0
```

Custom Input

```

10
1 1 1 1 1 1 1 1 1 1
5
1 10
2 4
3 9
5 6

```

Status Successfully executed Date 2023-02-01 06:05:38 Time 0.007772 sec Mem 5380 kB

Input

```

10
1 1 1 1 1 1 1 1 1 1
5
1 10
2 4
3 9
5 6

```

Output

```

10
0
0
0
0
0

```

Outcomes:

CO2. Understand the fundamental concepts for managing the data using different data structures such as lists, queues, trees etc.

Conclusion: (Conclusion to be based on the objectives and outcomes achieved)

Successfully understood how to approach a problem which requires the usage of map and also utilized the concept of prefix sum to solve each query in $O(1)$ time thus reducing time required

References:

1. <https://www.hackerearth.com/practice/data-structures/arrays/1-d/practice-problems/algorithm/sum-as-per-frequency-88b00c1f/>
2. T.H. Coreman ,C.E. Leiserson,R.L. Rivest, and C. Stein, " Introduction to algorithms", 3rd Edition 2009, Prentice Hall India Publication

3. Antti Laaksonen, “Guide to Competitive Programming”, Springer, 2018
4. Gayle Laakmann McDowell, “Cracking the Coding Interview”, CareerCup LLC, 2015
5. Steven S. Skiena Miguel A. Revilla, “Programming challenges, The Programming Contest Training Manual”, Springer, 2006
6. Antti Laaksonen, “Competitive Programmer’s Handbook”, Hand book, 2018
7. Steven Halim and Felix Halim, “Competitive Programming 3: The Lower Bounds of Programming Contests”, Handbook for ACM ICPC

