Faculty of Engineering, University of Jaffna,

Department of Computer Engineering

EC4070: Data Structures and algorithms

Lab 02

Chapter 2: Divide and conquer

Duration: 3 Hours Lecturer: Ms. Sujanthika M.

Instructions

i. Submit the code files and screenshot of the outputs in a zipped folder by naming as 2022EAAA_Lab01(AAA – Your Registration Number)
ii. Submit your files before the given deadline.
iii. Any plagiarized work will be given 0 marks.

---

1. Implement Merge sort algorithm to sort an integer array in ascending order.

   Input format:

   - First line: the size of the array
   - Second line: the array to be sorted (numbers separated by spaces)

   Output format: The sorted array

   Sample input:
   6
   1 9 3 5 4 7

   Sample Output:
   1 3 4 5 7 9

2. Milly is at the examination hall where she is reading a question paper. She checked the question paper and discovered that there are **N** questions in that paper. Each question has some score values. Ideally, it's like questions requiring more time have more score value and strangely no two questions on the paper require same time to be solved.
   She is very excited by looking at these questions. She decided to solve **K** questions while maximizing their score value. Could you please help Milly to determine the exact time she needs to solve the questions.

Use Merge sort to solve this problem

Input

- First line of input contains two space separated integers **N** and **Q**, where **N** is the number of questions available, and **Q** is number of queries
- Next line contains **N** space separated integers denoting the time $\mathbf{T_i}$ of **N** questions
- Next line contains **N** space separated integers denoting the scores $\mathbf{S_i}$ of **N** questions
- Next **Q** lines contain a number **K** each, the number of questions she wants to solve

Output

- Print the time required for each query in a separate line.

Sample input:
5 2
2 3 9 4 5
3 5 11 6 7
5
3

Sample Output:
23
18

3. You are given an array **A** with **N** elements with ascending order, and **Q** queries to deal with. For each query, you are given an integer **X**, and you're supposed to find out if **X** is present in the array **A** or **not**.

Input Format

- The first line contains two integers, **N** and **Q**, denoting the size of array **A** and number of queries **Q**.
- The second line contains **N** space separated integers, denoting the array of elements $\mathbf{A_i}$. The next **Q** lines contain a single integer **X** per line.

Output Format

For each query, print YES if the **X** is in the array, otherwise print NO.

Sample Input 0

```
10 5
1 2 3 4 5 66 77 88 99 890
1
66
55
345
890
```

Sample Output 0

```
YES
YES
NO
NO
YES
```

The **pseudocode** for the binary search is given below:

```
BinarySearch(array, target):
{
  left = lowestBound(array)
  right = highestBound(array)

  while (left <= right)
  {
    middle = (left + right) / 2
    if(target = array[middle])
      return middle
    else if(target < array[middle])
      right = middle - 1
    else
      left = middle + 1
  }
  return -1
}
```