INSERTION SORT ADVANCED ANALYSIS AND XOR OR

A PROJECT REPORT

Submitted by

SHOBIKA.R
(Reg. No: 19DS034)

of

5 Year Integrated M.Sc., (Data Science)

In

DEPARTMENT OF APPLIED MATHEMATICS AND COMPUTATIONAL SCIENCE

THIAGARAJAR COLLEGE OF ENGINEERING

(A Govt. Aided Autonomous Institution

affiliated to Anna University)

MADURAI – 625015

June 2020

THIAGARAJAR COLLEGE OF ENGINEERING,
MADURAI

DEPARTMENT OF APPLIED MATHEMATICS AND

COMPUTATIONAL SCIENCE

BONAFIDE CERTIFICATE

Certified that this project report on "Insertion Sort Advanced Analysis" and "AND xor OR" is the bonafide work of SHOBIKA.R (19S034), Third Semester student of 5 Year Integrated MSc (Data Science) Degree Programme, who carried out the project under my supervision from 04.01.2021 to 10.01.2021 during the academic year 2020-2021.

The project report was submitted to the department on 10/01/2021 for evaluation/assessment.

Dr. S. Parthasarathy
Professor & Head
Department of Applied Mathematics
and Computational Science

Dr. D. Anitha
Project Guide
Assistant Professor in Data Science
Department of Applied
Mathematics and Computational
Science

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Problem 1 :

1. Introduction :

One of the most strongest and challenging draws for the advent years is program coding. Knowing how to code is also critical. Coding is a basic literacy in the digital age, and it is important for people to be able to work with and understand the evolving technology around them. Each coding skill learned and that is developed can positively influence the ability to solve problems strategically, as well as the ability to form logical modeling skills and cognitive styles. In its most basic terms, programming is really just assigning a computer a task to do based on the logical guidelines outlined. Highly complex tasks are essentially a collection of smaller operations once broken. This methodical and logic-heavy approach to problem solving can be a boon for figuring out problems beyond a coding challenge.

A frequently used method to organize, process, design, store and retrieve data is Data Structures. A data structure is a named location that can be used to store and organize data. And, an algorithm is a collection of steps to solve a particular problem. This project deals with analyzing Sorting algorithm using insertion sort in Hackerrank, an online programming platform that enables users to solve programming paradigms from topics across different languages.

2. Objective of the project :

This project covers three main objective:

- ✓ Grasping knowledge on Sorting Algorithms.
- ✓ Using insertion Sort to understand and solve a problem in Hackerrank.
- ✓ Clearing all test cases and submission.

Sorting Algorithm:

A Sorting Algorithm is used to rearrange a given array or list elements according to a comparison operator on the elements. The comparison operator is used to decide the new order of element in the respective data structure. Sorting is also often useful for canonicalizing data and for producing human-readable output. More formally, the output of any sorting algorithm must satisfy two conditions:

1. The output is in non-decreasing order (each element is no smaller than the previous element according to the desired total order)
2. The output is a permutation (a reordering, yet retaining all of the original elements) of the input.

Further, the input data is often stored in an array, which allows random access.

Insertion Sort:

Insertion sort is a simple sorting algorithm that works similar to sorting playing cards. The array is virtually split into a sorted and an unsorted part. Values from the unsorted part are picked and placed at the correct position in the sorted part. We can use binary search to reduce the number of comparisons in normal insertion sort. Binary Insertion Sort uses binary search to find the proper location to insert the selected item at each iteration. In normal insertion, sorting takes O(i) (at ith iteration) in worst case. We can reduce it to O(logi) by using binary search. The algorithm, as a whole, still has a running worst case running time of O(n2) because of the series of swaps required for each insertion.

Test cases:

In Coding Questions, test cases are the different types of inputs to the code to test the defined logic and produce the output. A test case is termed "passed" when the output from code exactly matches the expected output. Else, the status "Wrong Answer" is indicated against the test case. The problem contains two sample test cases for which he desired output is expected and consists a total of 14 hidden test cases for which the output is verified for invariably different inputs.

Problem statement:

The proposed problem is : Sometimes, arrays may be too large for us to wait around for insertion sort to finish. Another way to calculate the number of shifts an insertion sort performs when sorting an array is -

If $k[i]$ is the number of elements over which the i^{th} element of the array has to shift, then the total number of shifts will be $k[1] + k[2] + \ldots + k[n]$.

Example:

arr = [4,3,2,1]

Array	Shifts
[4,3,2,1]	
[3,4,2,1]	1
[2,3,4,1]	2
[1,2,3,4]	3

Total shifts = 1 + 2 + 3 = 6

Complete the insertionSort function in the editor of Hackerrank.

insertionSort has the following parameter(s):

int arr[n]: an array of integers

Returns:

int: the number of shifts required to sort the array

3. Description of the project :

The problem needs to be dealt by using the above technique of finding the number of Total shifts. The program should yield the desired output for the given input. The Input format, constraints along with sample input is as below :

Input Format:

The first line contains a single integer t, the number of queries to perform.

The following t pairs of lines are as follows :

- The first line contains an integer n, the length of arr.
- The second line contains n space-separated integers arr[i].

Constraints:

Constraints mention the range of the variables used to describe the problem statement. Constraints help the contestants to assess the input size and to figure out if the algorithm will terminate in allotted time.

The given constraints are :

$1 \le t \le 15$

$1 \le n \le 100000$

$1 \le a[i] \le 10000000$

Sample Input:

```
2
5
1 1 1 2 2
5
2 1 3 1 2
```

Sample Output:

```
0
4
```

Solution:

The problem can be solved by using insertion sort in C++.

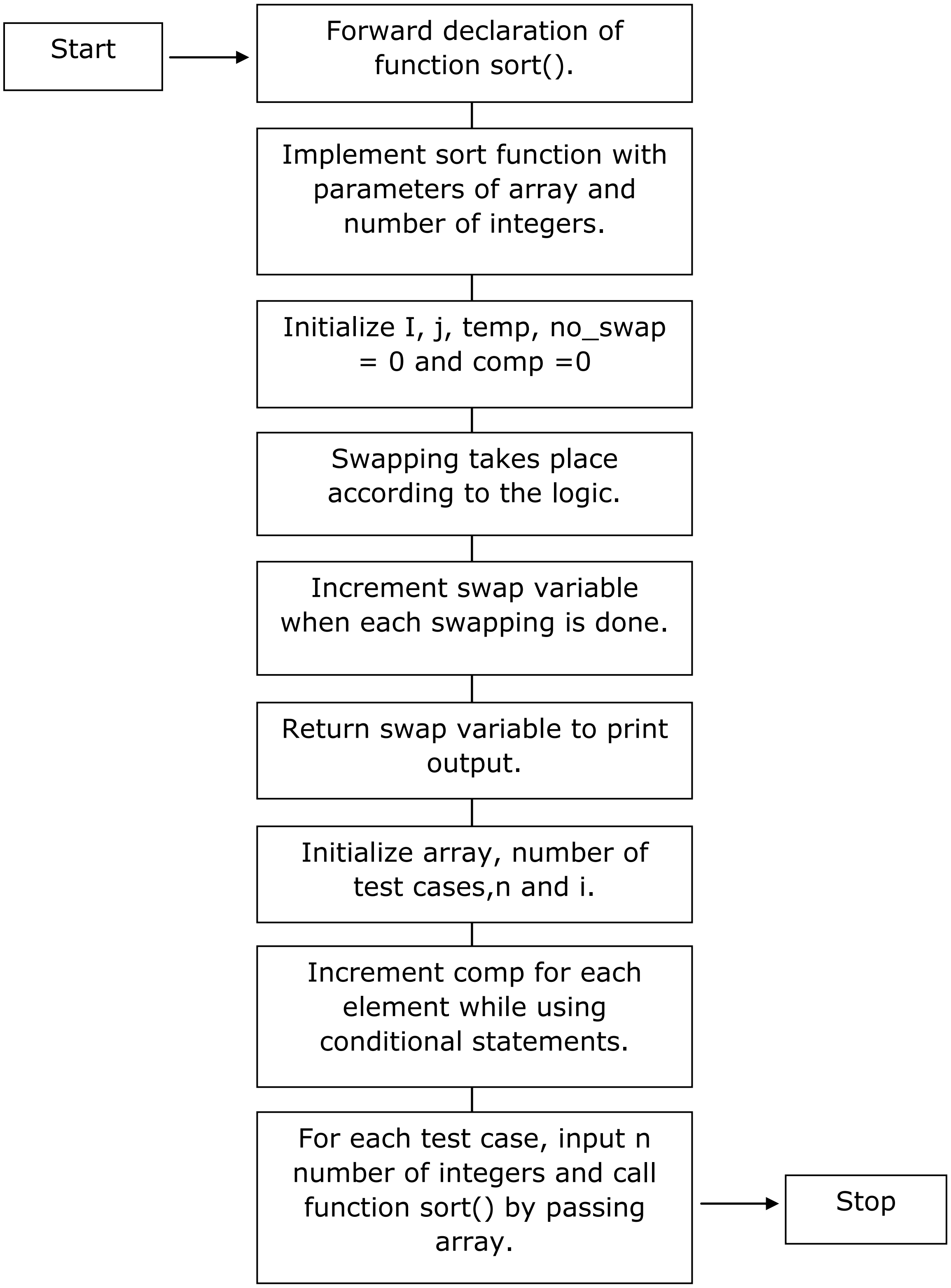
The first query is already sorted, so there is no need to shift any elements. In the second case, it will proceed in the following way.

There will be 4 moves to represent
Array: 2 1 3 1 2 -> 1 2 3 1 2 -> 1 1 2 3 2 -> 1 1 2 2 3
to give the desired output as 0 and 4.

Array	Moves:
1 2 3 1 2	1
1 1 2 3 2	2
1 1 2 2 3	1

4. Implementation :

a) Algorithm:


Start
Forward declaration of function sort().
Implement sort function with parameters of array and number of integers.
Initialize I, j, temp, no_swap = 0 and comp =0
Swapping takes place according to the logic.
Increment swap variable when each swapping is done.
Return swap variable to print output.
Initialize array, number of test cases,n and i.
Increment comp for each element while using conditional statements.
For each test case, input n number of integers and call function sort() by passing array.
Stop

b) Program Code:

```
#include <iostream>
#include<stdio.h>
using namespace std;
int sort(int a[],int n);
int main()
{
    int array[10];
    int tc,n,i;
    cin>>tc;
    while(tc)
    {
    cin>>n;
    for (i = 0; i < n; i++)
    cin>>array[i];
    cout<<sort(array,n)<<endl;
    tc--;
    }
    return 0;
}
int sort(int array[] ,int n)
{
int i,j,temp,no_swap=0,comp=0;
    for (i = 1; i < n; i++)
    {
    j = i;
    comp++;
    while ((j > 0) && (array[j - 1] > array[j]))
        {
```

```
        if(array[j-1]>array[j])
        comp++;
        temp = array[j - 1];
        array[j - 1] = array[j];
        array[j] = temp;
        j--;
        no_swap++;//increment swap variable when actually swap is done
        }
    }
    return no_swap;
}
```

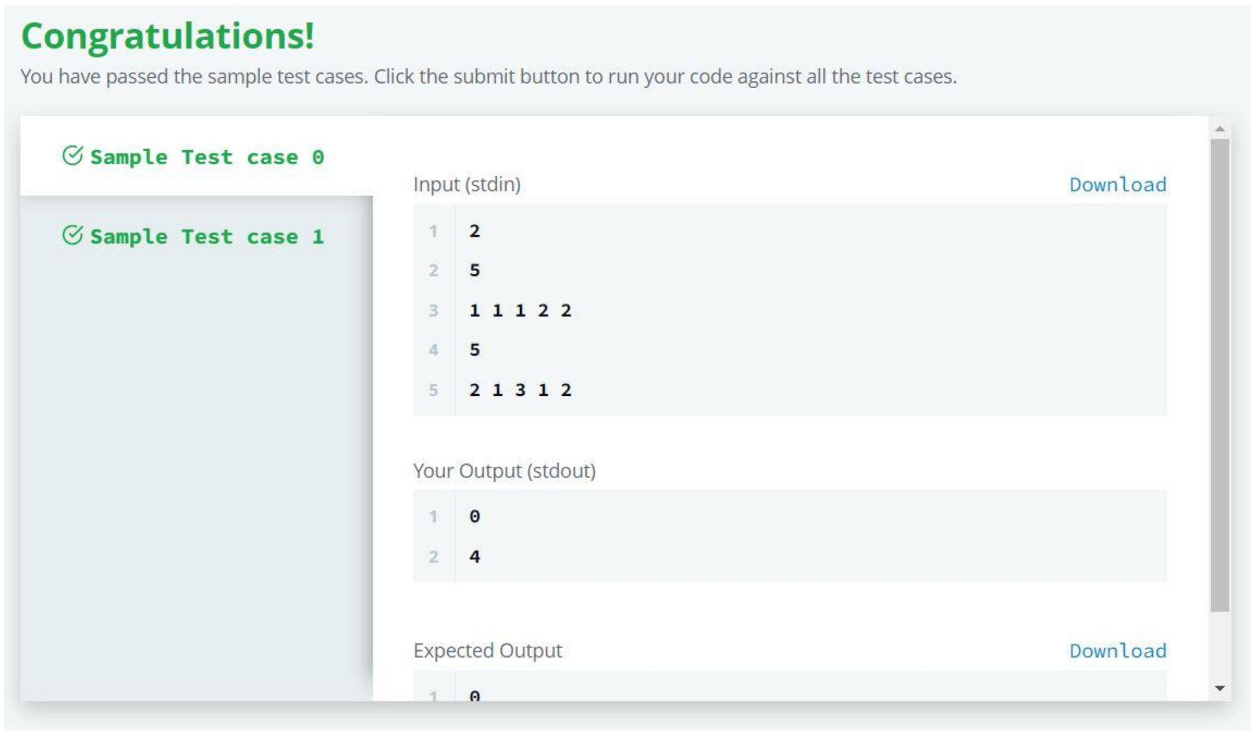
c) Program Explanation:

The sort function is declared beforehand and then implemented in the program. Integer data type array[] and number of elements are passed as parameters. Variables like i, j, temp, no_swap and comp are declared. no_swap is to count the number of times swapping is performed and temp is created to store the temporary variable. For every integer in array, I is copied to j while comp is incremented. Condition for greater number is checked for each integer and comp is incremented if the integer is lesser than the previous integer. To sort through and swap two integers, store the previous integer in temporary variable and store the current variable in previous variable place and assign the current variable to temporary variable. Decrement j and increment no of swap. Return number of swap as the variable.

In the driver program, initialize variables like number of test cases(tc), n and i. Input number of test cases to run on loop and n number of integers as array using for loop. Pass array and n as parameters and call for function sort(). Print the value that is returned to get total number of shifts.

d) Sample Test Cases:

The desired output is verified for the given 2 sample input of test cases.



Congratulations!

You have passed the sample test cases. Click the submit button to run your code against all the test cases.

✓ Sample Test case 0

✓ Sample Test case 1

Input (stdin) Download

```
2
5
1 1 1 2 2
5
2 1 3 1 2
```

Your Output (stdout)

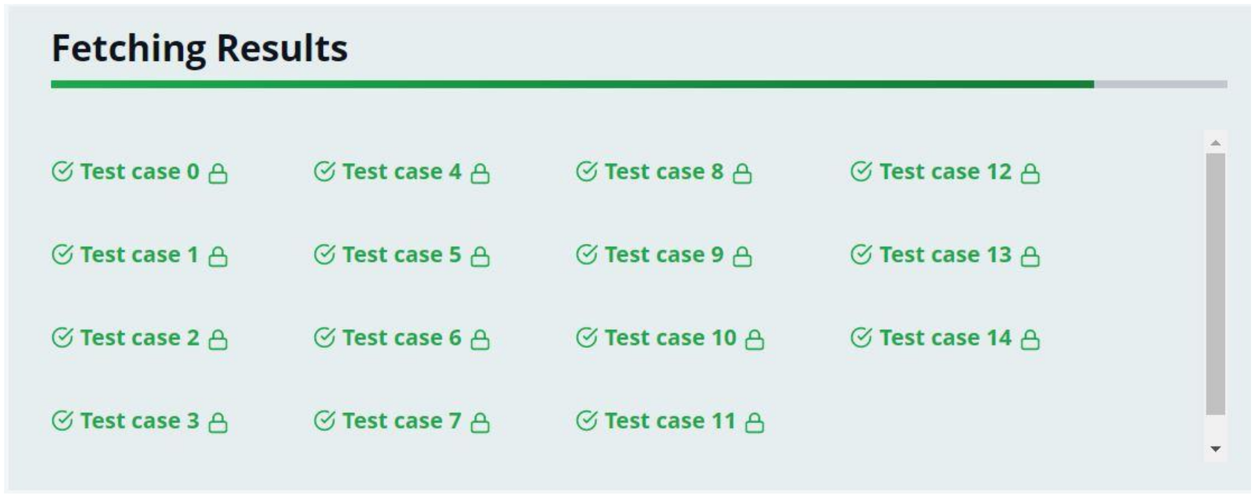
```
0
4
```

Expected Output Download

```
0
```

Fig 1.4.1 : Sample test case

e) Final Test Cases:

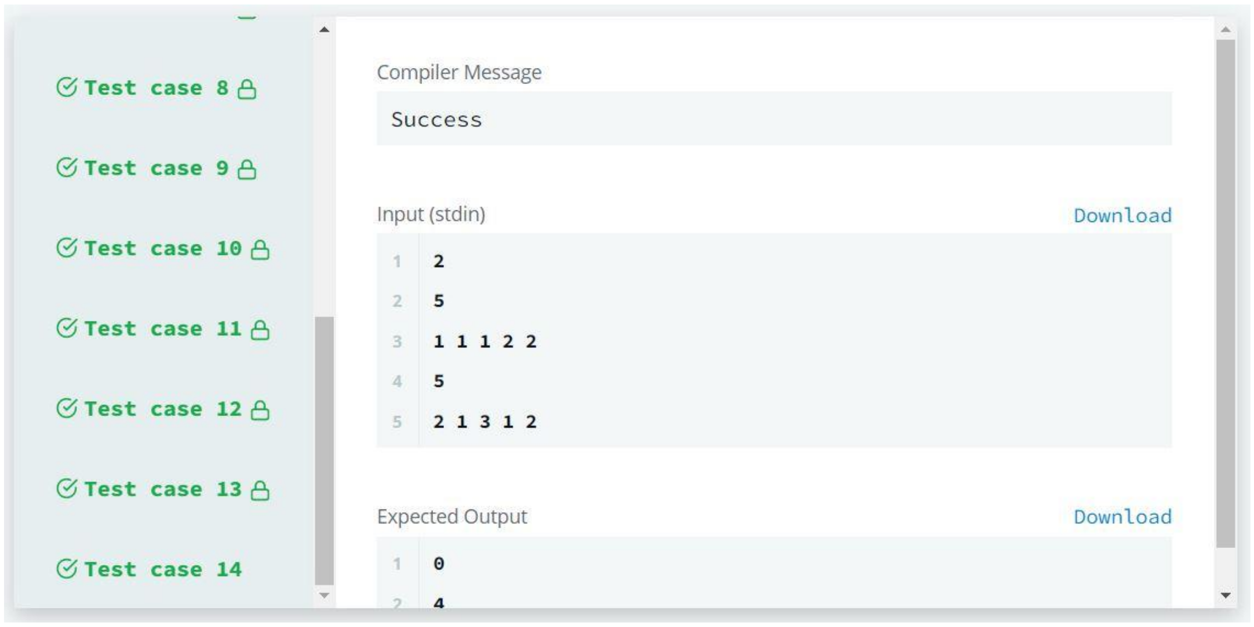


Fetching Results

✓ Test case 0	✓ Test case 4	✓ Test case 8	✓ Test case 12
✓ Test case 1	✓ Test case 5	✓ Test case 9	✓ Test case 13
✓ Test case 2	✓ Test case 6	✓ Test case 10	✓ Test case 14
✓ Test case 3	✓ Test case 7	✓ Test case 11	

Fig 1.4.2 : Compilation

The final 14 test cases are passed accordingly.



Compiler Message

```
Success
```

Input (stdin)

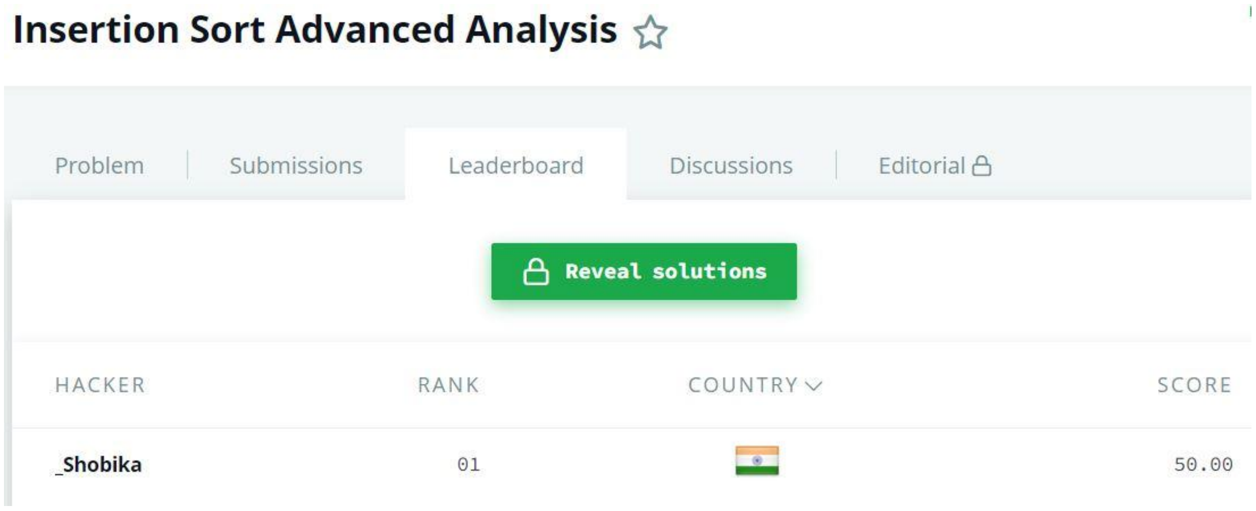
```
2
5
1 1 1 2 2
5
2 1 3 1 2
```

Expected Output

```
0
4
```

Fig 1.4.3 : Final test cases

f) Leader board:



Insertion Sort Advanced Analysis

HACKER	RANK	COUNTRY	SCORE
_Shobika	01	🇮🇳	50.00

Fig 1.4.4 : Leader board

5. Significance of the project:

Insertion sort is one of the sorting algorithms in Advanced data structures with time complexity O(n*2). Insertion sort is mostly used when input array is almost sorted, only few elements are misplaced in complete big array.

Boundary Cases: Insertion sort takes maximum time to sort if elements are sorted in reverse order. And it takes minimum time (Order of n) when elements are already sorted.

Auxiliary Space: O(1)
Algorithmic Paradigm: Incremental Approach
Sorting In Place: Yes
Stable: Yes
Online: Yes

Advantages:

- ✓ The pure simplicity of the algorithm.
- ✓ The relative order of items with equal keys does not change.
- ✓ The ability to sort a list as it is being received.
- ✓ It only requires a constant amount of additional memory space—O(1).
- ✓ Adaptive - efficient for data sets that are already substantially sorted: the time complexity is O(n + d), where d is the number of inversions.
- ✓ More efficient in practice than most other simple quadratic, i.e. O(n2) algorithms such as selection sort or bubble sort; the best case (nearly sorted input) is O(n)

Challenges:

The disadvantage of the insertion sort is that it does not perform as well as other, better sorting algorithms. With n-squared steps required for every n element to be sorted, the insertion sort does not deal well with a huge list. Therefore, the insertion sort is particularly useful only when sorting a list of few items.

6. Conclusion:

Thus the problem is resolved by implementing Insertion Sort and the output is extracted. As stated in the objective, a clear understanding between sorting algorithms is proposed. All the test cases have been cleared and screenshots attached. The performance of insertion sort in the above program is examined as -

Worst-case performance	O(n^2) comparisons and swaps
Best-case performance	O(n) comparisons, O(1) swaps
Average performance	O(n^2) comparisons and swaps
Worst-case space complexity	O(n) total, O(1) auxiliary

Table 1.6.1 : Complexity and performance

The program can be tested against custom input and different types of sort like Bubble sort, Merge sort, Quick sort, Selection sort, Heap sort, Bucket sort can be implemented to analyze time complexity and efficiency between them.

Problem 2 :

1. Introduction :

Learning data structures and algorithms allow us to write efficient and optimized computer programs. Data structures represent the knowledge in which data is organized, that is implemented to decrease time complexity and increase efficiency. Basic data structure includes Stack, Queue while complex and Advanced Data Structure deals with Disjoin Sets, Trees, Tries etc…

The following project revolves around handling program related to Simple Data Structure and solving an advanced problem in this field through Hackerrank. Solving problems in Data Structures using C++ has been chosen because of the recent course completion and immense interest in Advanced Data Structures.

AND xor OR are logical operators in Computer science used to solve problems. This problem deals with implementing Stack data structure to use logical operations on two variables.

2. Objective of the project :

This project covers two main objective:

- ✓ Understanding and implementing Stack to solve the given problem in Hackerrank.
- ✓ Clearing sample test cases and all other test cases.

Fathoming the problem and establishing a clear understanding between different data structures and executing the solution.

Stack Data structure:

Stack is a linear data structure which follows a particular order in which the operations are performed. The order may be

1. LIFO(Last In First Out) or
2. FILO(First In Last Out).

Mainly the following three basic operations are performed in the stack:

- **Push:** Adds an item in the stack. If the stack is full, then it is said to be an Overflow condition.
- **Pop:** Removes an item from the stack. The items are popped in the reversed order in which they are pushed. If the stack is empty, then it is said to be an Underflow condition.
- **Peek or Top:** Returns top element of stack.
- **isEmpty:** Returns true if stack is empty, else false.

There are two ways to implement a stack:

- Using array
- Using linked list

Test cases:

The AND xor OR problem is solved using Stack data structure. It consists a total of 32 test cases for which the output is verified for invariably different inputs.

Problem statement:

The proposed problem is : Given an array A[] of N distinct elements, let M_1 and M_2 be the smallest and the next smallest element in the interval [L,R] where $1 \leq L < R \leq N$.

$$S_i = (((M_1 \wedge M_2) \oplus (M_1 \vee M_2)) \wedge (M_1 \oplus M_2))$$

Where $\wedge$, $\vee$, $\oplus$ are the bitwise operators AND, OR and XOR respectively.

Find the maximum possible value of S_i.

Example:

The given array;

arr = [10,5,7,2]

The smallest interval would be [1,2]

The result will be maximum of S_i.

Complete the program in the editor of Hackerrank.

3. Description of the project :

The problem is to find the maximum value of S_i by computing its value using M_1 and M_2 with the help of logical operators. The program should yield the desired output for the given input. The Input format, constraints along with sample input is as below :

Input Format:

The first line contains a single integer N.

Second line contains N integers representing elements of the array A[].

Constraints:

The given constraints are :

$1 \le N \le 10^6$

$1 \le a[i] \le 10^9$

Sample Input:

5

9 6 3 5 2

Sample Output:

15

Solution:

The problem can be solved by using stack operations in C++.

Considering the interval [1,2] according to the input, the result will be maximum.

In that case M_1 would be 9 and M_2 will be 6, deriving the answer according to the formula of S_i

$$S_i = (((9 \Lambda 6) \oplus (9\ V\ 6)) \Lambda (9 \oplus 6))$$

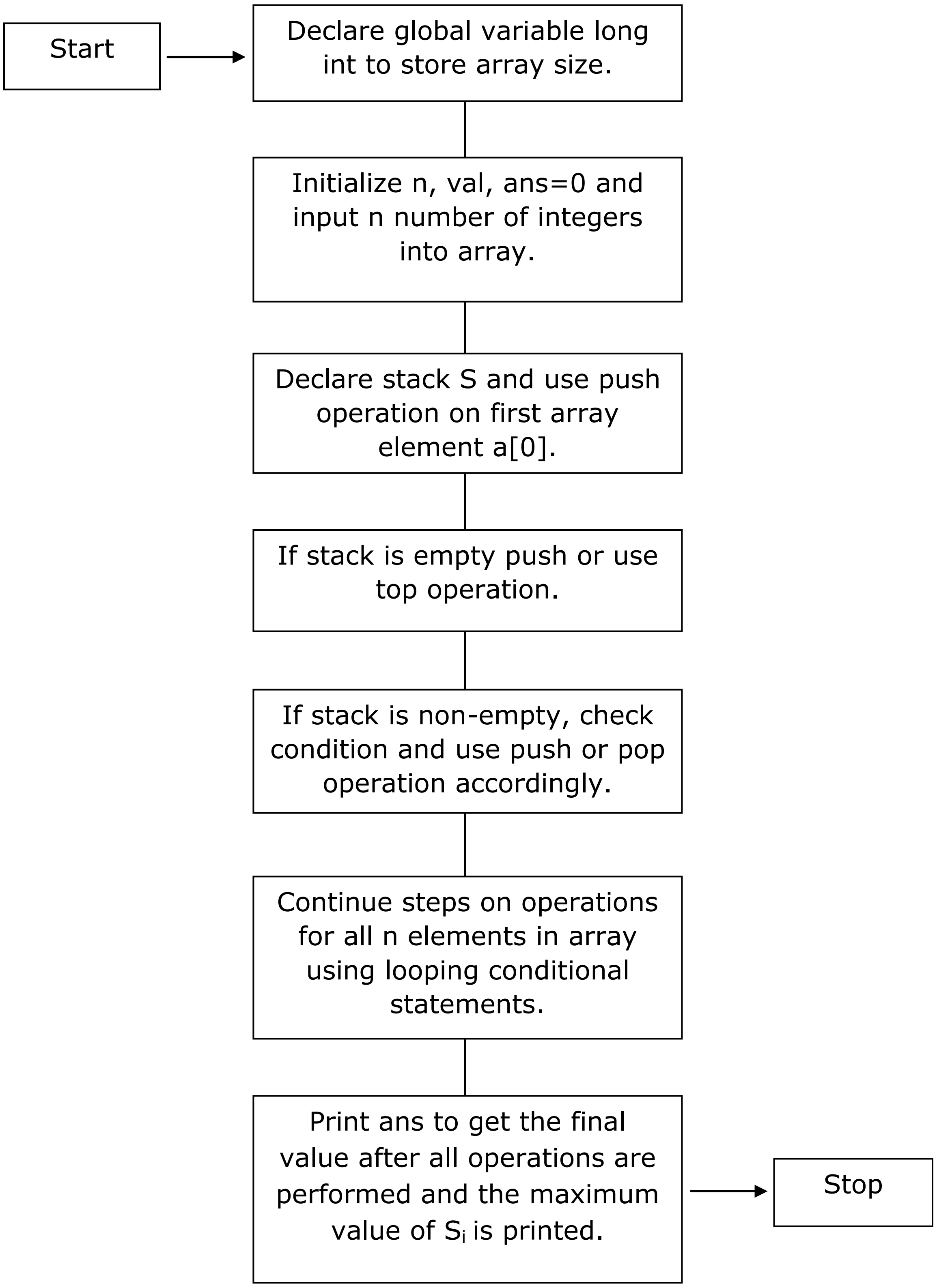
=15

4. Implementation :

Stack operations implemented in this problem are:

1. empty()
2. top()
3. push()
4. pop()

a) Algorithm:


Start
Declare global variable long int to store array size.
Initialize n, val, ans=0 and input n number of integers into array.
Declare stack S and use push operation on first array element a[0].
If stack is empty push or use top operation.
If stack is non-empty, check condition and use push or pop operation accordingly.
Continue steps on operations for all n elements in array using looping conditional statements.
Print ans to get the final value after all operations are performed and the maximum value of S_i is printed.
Stop

b) Program Code:

```
#include<iostream>
#include<stack>
#define ll long long
int a[1000000];
using namespace std;
int main()
{
    int n;
    ll int ans=0,val;
    cin>>n;
    for(int i=0;i<n;i++)
        cin>>a[i];
    stack<ll int> S;
    S.push(a[0]);
    for(int i=1;i<n;i++)
    {
        while(!S.empty() && a[i]<S.top())
        {
            val=S.top() ^ a[i];
            if(val > ans)
                ans=val;
            S.pop();
        }
        if(S.empty())
            S.push(a[i]);
        else
        {
            val=S.top() ^ a[i];
            if(val > ans)
                ans=val;
            S.push(a[i]);
        }
    }
    cout<<ans;
    return 0;
}
```

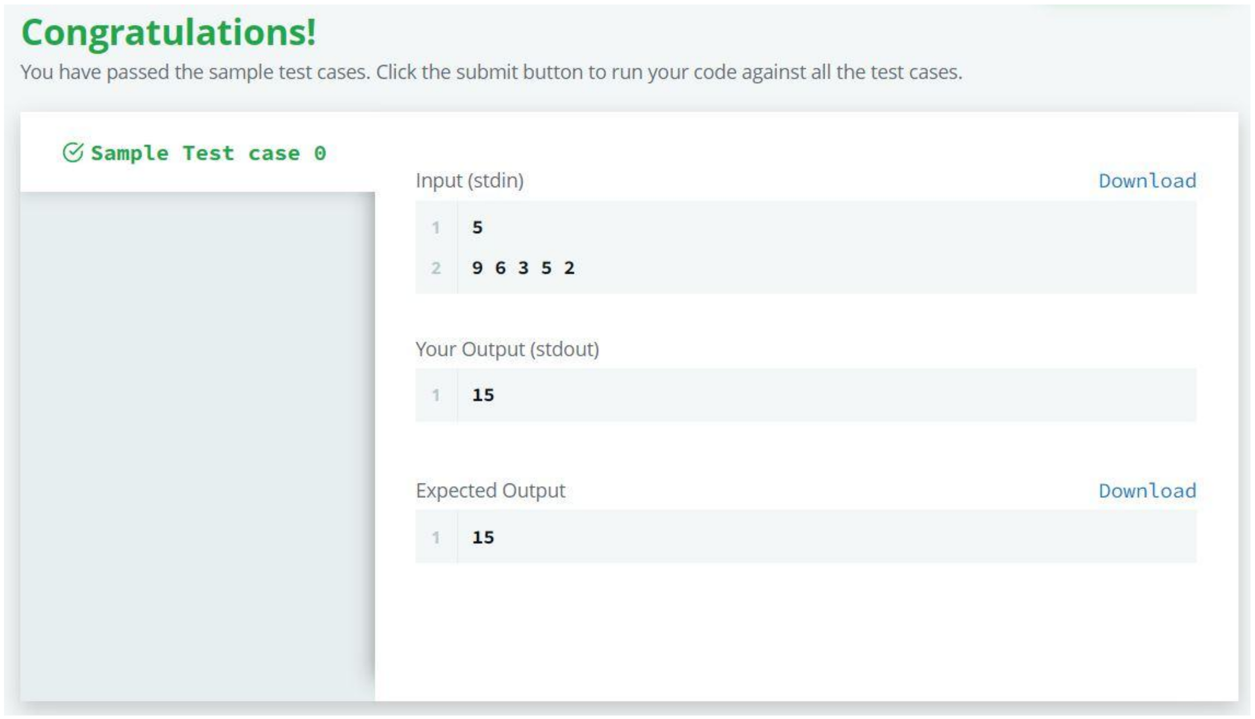
c) Program Explanation:

The given constraint has array size of 10^9, so we declare array as a global variable with long int. In the driver program, initialize nuber of integers(n), ans=0 and value. Input the total number of integers and use array to store the given n elements. Create an empty stack S to perform operations. Push the first element a[0] into the empty stack. Initialize a for loop and for each integer from 1 to n do the following operations; there are 3 conditions for the integers in the array to follow.

1) In case of non empty stack and if the current element is lesser than the top element, assign value. If value is greater than ans which is initially 0, then assign ans as value and perform pop operation on stack.
2) If stack is empty the current element is pushed inside to occupy the first position without any extensive operations.
3) If any of these condition do not satisfy then again assign value. Check the conditionif ans is greater than val. If satisfied val is assigned to ans. If condition fails, element is pushed into stack.

Thus operations top(), push(), pop(), empty() are implemented using stack data structure. The integer in ans is printed to obtain the maximum value of S_i according to the proposed formula in the problem statement

d) Sample Test Cases:



Congratulations!

You have passed the sample test cases. Click the submit button to run your code against all the test cases.

✓ Sample Test case 0

Input (stdin)

```
5
9 6 3 5 2
```

Your Output (stdout)

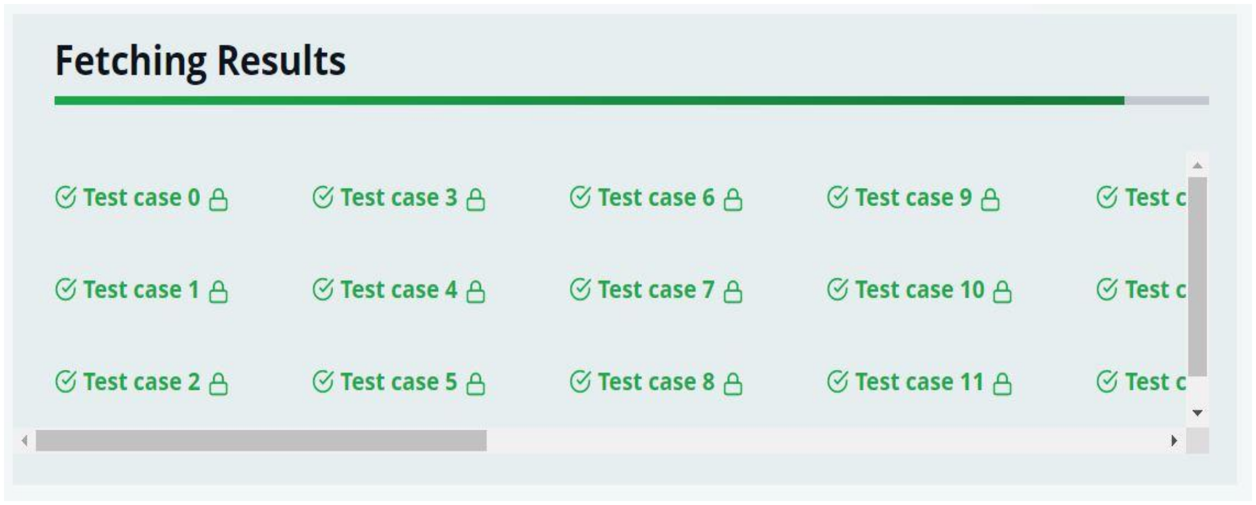
```
15
```

Expected Output

```
15
```

Fig 2.4.1 : Sample test case

e) Final Test Cases:



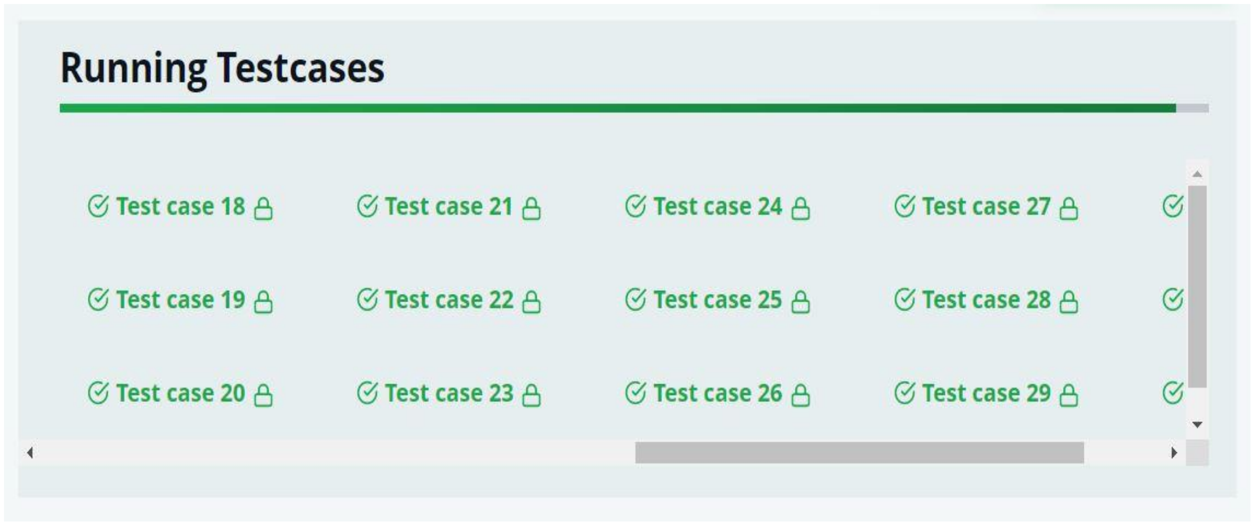
Fetching Results

✓ Test case 0 ✓ Test case 3 ✓ Test case 6 ✓ Test case 9 ✓ Test c

✓ Test case 1 ✓ Test case 4 ✓ Test case 7 ✓ Test case 10 ✓ Test c

✓ Test case 2 ✓ Test case 5 ✓ Test case 8 ✓ Test case 11 ✓ Test c

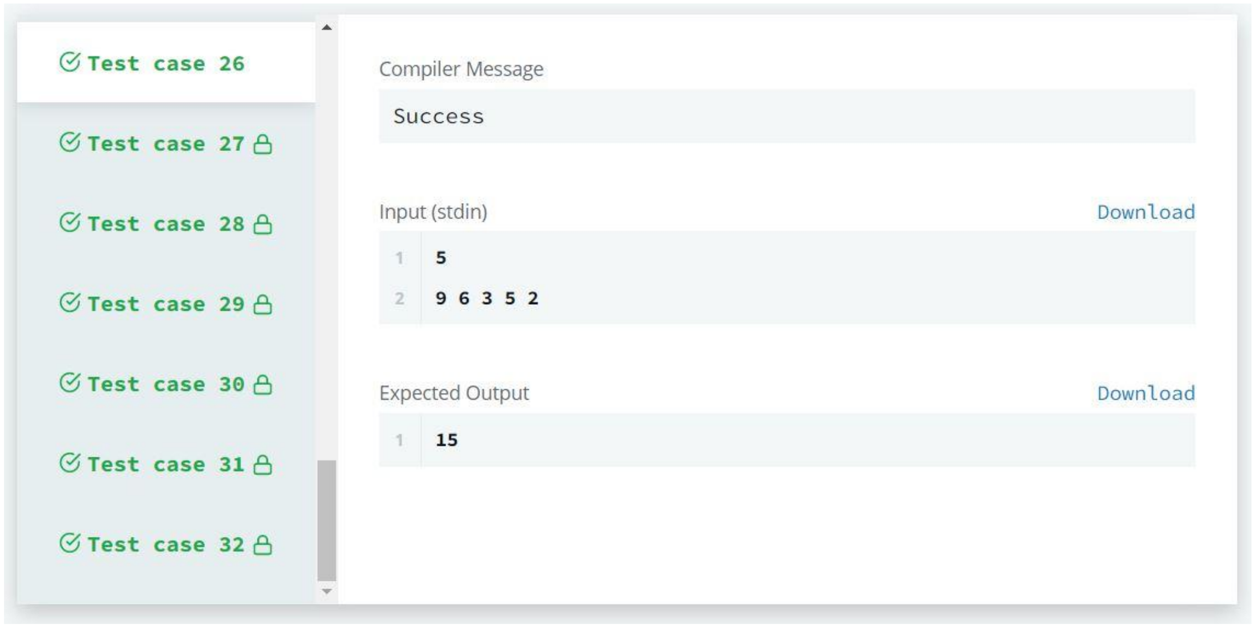
Fig 2.4.2a



Running Testcases

Test case 18 Test case 21 Test case 24 Test case 27
Test case 19 Test case 22 Test case 25 Test case 28
Test case 20 Test case 23 Test case 26 Test case 29

Fig 2.4.2b : Compilation



Test case 26
Test case 27
Test case 28
Test case 29
Test case 30
Test case 31
Test case 32

Compiler Message

```
Success
```

Input (stdin) Download

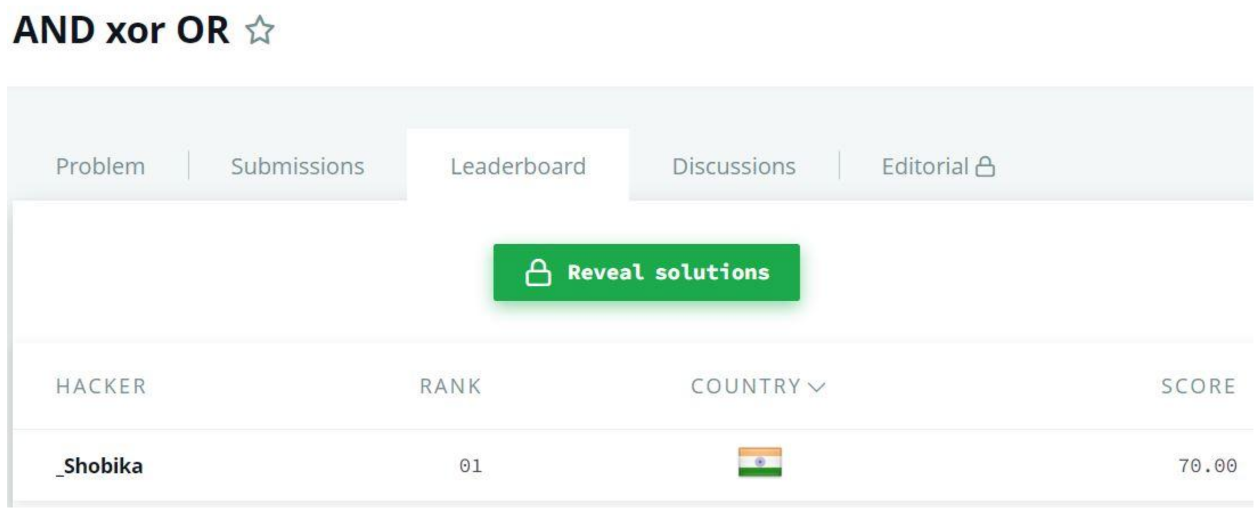
```
5
9 6 3 5 2
```

Expected Output Download

```
15
```

Fig 2.4.3 : Final test cases

f) Leader board:



AND xor OR ☆

Problem | Submissions | Leaderboard | Discussions | Editorial

Reveal solutions

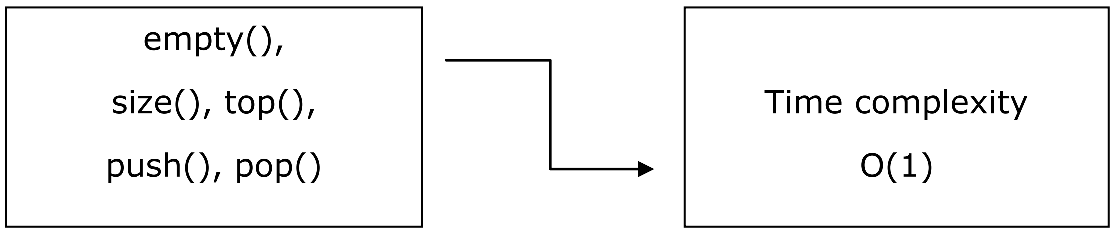
HACKER	RANK	COUNTRY	SCORE
_Shobika	01	🇮🇳	70.00

Fig 2.4.4 : Leader board

5. Significance of the project:

Stacks are useful data structures and are used in a variety of ways in computer science. In general, stacks are useful for processing nested structures or for functions which call other functions (or themselves). tacks are used to implement functions, parsers, expression evaluation, and backtracking algorithms.

A pile of books, a stack of dinner plates, a box of pringles potato chips can all be thought of examples of stacks. The basic operating principle is that last item you put in is first item you can take out. Some functions and their time complexity are



Advantages:

- ✓ Helps you to manage the data in a Last In First Out(LIFO) method which is not possible with Linked list and array.
- ✓ When a function is called the local variables are stored in a stack, and it is automatically destroyed once returned.
- ✓ A stack is used when a variable is not used outside that function.
- ✓ It allows you to control how memory is allocated and deallocated.
- ✓ Stack automatically cleans up the object.
- ✓ Not easily corrupted
- ✓ Variables cannot be resized.

Challenges:

- Stack memory is very limited.
- Creating too many objects on the stack can increase the risk of stack overflow.
- Random access is not possible.
- Variable storage will be overwritten, which sometimes leads to undefined behavior of the function or program.
- The stack will fall outside of the memory area, which might lead to an abnormal termination.

6. Conclusion:

Stack data structure has a variety of uses and a wide range of applications including;

- Balancing of symbols
- Infix to Postfix /Prefix conversion
- Redo-undo features at many places like editors, photoshop.
- Forward and backward feature in web browsers
- Used in many algorithms like Tower of Hanoi, tree traversals, stock span problem, histogram problem.
- Backtracking is one of the algorithm designing technique .Some example of back tracking are Knight-Tour problem,N-Queen problem,find your way through maze and game like chess or checkers in all this problems we dive into someway if that way is not efficient we come back to the previous state and go into some another path. To get back from current state we need to store the previous state for that purpose we need stack.
- In Graph Algorithms like Topological Sorting and Strongly Connected Components
- In Memory management any modern computer uses stack as the primary-management for a running purpose.Each program that is running in a computer system has its own memory allocations
- String reversal is also a another application of stack.Here one by one each character get inserted into the stack.So the first character of string is on the bottom of the stack and the last element of string is on the top of stack. After Performing the pop operations on stack we get string in reverse order .

The given program was successfully completed with all the given test cases passed an output verified.

PROJECT WORKSHEET / DIARY

	Date	Topics learned / Activity carried out / Task completed / Online /E-resources accessed
WEEK 1	04.01.2021	Discussion for selection of scheme. Problems are selected based on the scheme.
	05.01.2021	The progam is split among members and each member is assigned a task.
	06.01.2021	Solving program 1 and completing the coding part
	07.01.2021	2nd problem is discussed, solved and test cases are passed.
	08.01.2021	Conclusion and Solution is outlined and project report is worked upon.
	09.01.2021	Completion of project reprt and finalizing the document.
	10.01.2021	Submission of project report

Signature of the student **Signature of the faculty guide**

(with date) **(with date)**