

ADVANCED DATA STRUCTURE

A PROJECT REPORT

Submitted by

SHIBANI.K.R.

(Reg. No: 19S033)

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

Jan 2021

THIAGARAJAR COLLEGE OF ENGINEERING, MADURAI



DEPARTMENT OF APPLIED MATHEMATICS AND COMPUTATIONAL SCIENCE

BONAFIDE CERTIFICATE

Certified that this project report "ADVANCED DATA STRUCTURE" is the bonafide work of SHIBANI.K.R (19S033), 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 09.01.2021 during the academic year 2021-2022.

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

TABLE OF INDEX

LIST OF FIGURES

FIGURE INDEX	TITLE OF THE FIGURE	PAGE NO
	PROBLEM 1	
4.1	SAMPLE CASE	11
4.2	TEST CASES	12
4.3	LEADER BOARD	12
	PROBLEM 2	
4.4.1	SAMPLE CASE	20
4.5.1	TEST CASES FROM 0-11	21
4.5.2	TEST CASES FROM 12-25	21
4.5.3	TEST CASES FROM 26-32	22
4.6.1	LEADER BOARD FOR PROBLEM 2	22

	PAGE NO
PROBLEM 1:	
A. Introduction	6
B. Objective of the Project	7
C. Description of the Project	8
D. Implementation	
➤ Algorithm	9
➤ Program code	9
➤ Program explanation	10
➤ Sample cases	11
➤ Final Test cases	12
➤ Leader board	12
E. Significance of the project	13
F. Conclusion	13
G. Project Worksheet / Diary	

PROBLEM 2:	PAGE NO
A. Introduction	14
B. Objective of the Project	15
C. Description of the Project	16
D. Implementation	
➤ Algorithm	17
➤ Program code	17
➤ Program explanation	19
➤ Sample cases	20
➤ Final Test cases	20
➤ Leader board	22
E. Significance of the project	22
F. Conclusion	23
G. Project Worksheet / Diary	24

PROBLEM 1: Insertion Sort Advanced Analysis

A.INTRODUCTION

The ability to test the code is probably one of the most important skills to know as a developer. Testing can mean adding print statements to output data at certain points in a program to visually check it. Testing can also involve developing separate programs to test the functionality of the code itself.

The Aim of the project is to bring a better level of understanding about sorting program in C++ and to pursue its objective. Insertion sort is explained in this project. Insertion sort is a simple sorting algorithm that builds the final sorted array one item at a time. It is much less efficient on large lists than more advanced algorithms such as quicksort, heapsort, or merge sort.

C++ is a general-purpose programming language that gives a clear structure to the programs and allows code to be reused, lowering development cost. C++ is a strongly typed and fast programming language makes it an ideal candidate for writing operating systems. In addition to this, it has a wide collection of system-level functions that also help in writing low-level programs.

Other applications include image processing, mobile sensor, banking applications and etc.. C++ becomes a default choice for implementing such systems as it is close to the hardware.

HACKERRANK:

The online platform used is Hackerrank. On Hackerrank, programs are done in hands-on manner in the browser. HackerRank is a place where programmers come together to solve problems in a wide range of Computer Science domains such as C++, algorithms, machine learning, or artificial intelligence, as well as to practice different programming paradigms like functional programming.

B. OBJECTIVE OF THE PROBLEM:

Insertion Sort is a simple sorting technique. Sometimes, arrays may be too large for us to wait around for insertion sort to finish. Insertion sort is the sorting mechanism where the sorted array is built having one item at a time. The array elements are compared with each other sequentially and then arranged simultaneously in some particular order.

Even though insertion sort is efficient, still, if it is provided with a sorted array to the insertion sort algorithm, it will still execute the outer for loop, thereby requiring n steps to sort an already sorted array of n elements, which makes its best case time complexity a linear function of n .

This is an in-place comparison-based sorting algorithm. Here, a sub-list is maintained which is always sorted. For example, the lower part of an array is maintained to be sorted. An element which is to be inserted in this sorted sub-list, has to find its appropriate place and then it has to be inserted there. Hence the name, insertion sort.

PROBLEM STATEMENT:

The objective of the problem is to calculate the number of shifts an insertion sort performs when sorting an array, 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 $k[1] + k[2] + \dots + k[n]$.

C.DESCRPTION OF THE PROBLEM:

In an insertion sort, the first element in the array is considered as sorted, even if it is an unsorted array. Each element in the array is checked with the previous elements, resulting in a growing sorted output list. With each iteration, the sorting algorithm removes one element at a time and finds the appropriate location within the sorted array and inserts it there. The iteration continues until the whole list is sorted.

Here the function description about the problem is given where sort function has following parameters. First line of the input will contain a positive integer. Number to be sorted are given in the array and each number are compared. While comparing if the is swapping then shifting value is counted and incremented to the next value to compare and it continues till the end of the array. At the end of the array the value(times how much time shifting has taken place) is printed.

Function description:

Sort has the following parameter(s):

- *int a[n]*: an array of integers

Returns- *int*: the number of shifts required to sort the array

Sample Input:

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

Sample output:

```
0
4
```


D.IMPLEMENTATION:

1. ALGORITHM

STEP 1: Start

STEP 2: Initialize array[10], tc, n for max number and i, j, temp, no_swap are passed as argument in the function. sort function is declared and defined.

STEP 3: In sort function for loop is implemented for sorting of the array elements and then j is initialised to I and incrementation of comp takes place.

STEP 4: While loop is implemented in which if condition is used to checked by comparing the array elements, and comp is incremented to next element and swap process takes place. No_swap variable is incremented whenever swap happens.

STEP 5: In main function, get the test case from the user and get the value of n.

STEP 6: No of test cases(tc) is read and while loop takes the test case as argument (it runs tc times) and array elements are got using for loop.

STEP 7: Sort function is called by passing the array, n and the result (total number of shift) is printed.

STEP 8: At last tc is decremented.

STEP 9: Stop

2. PROGRAM CODE

```
#include <iostream.h>
#include<stdio.h>

#include<conio.h>
int sort(int a[],int n);

void main()
{
    clrscr();

    int array[10];
```

```

int tc,n,i;
cout<<"\n enter test case..";
cin>>tc;

while(tc){
cout<<"\n enter n..";
cin>>n;
for (i = 0; i < n; i++)
scanf("%d",&array[i]);
cout<<"\n no.of swaps.."<<sort(array,n);

tc--;
}
getch();
}

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
}}

```

3. PROGRAM EXPLANATION

In this program, total number of shifting is calculated. First size of the array is taken and elements are inserted if the not arranged then it becomes unsorted array. when the unsorted array is taken and first element is compared with other elements if a minimum element is found then the element is shifted with the first element so it reaches its correct position and that shifting is taken into

account and followed till all the elements in the array are placed at their correct position.

FOR EXAMPLE:

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

4. SAMPLE TEST CASES

EXECUTION OF THE SAMPLE CODE:

The screenshot displays a code execution environment with the following components:

- Test Cases:** Two sample test cases are listed on the left, both marked as successful with green checkmarks:
 - Sample Test case 0
 - Sample Test case 1
- Input (stdin):** A table showing the input for the test cases:

1	2
2	5
3	1 1 1 2 2
4	5
5	2 1 3 1 2
- Your Output (stdout):** A table showing the output generated by the code:

1	0
2	4
- Expected Output:** A table showing the expected output for comparison:

1	0
2	4

Each input and output section includes a 'Download' link in the top right corner.

Fig 4.1-Sample case

5. FINAL TEST CASES

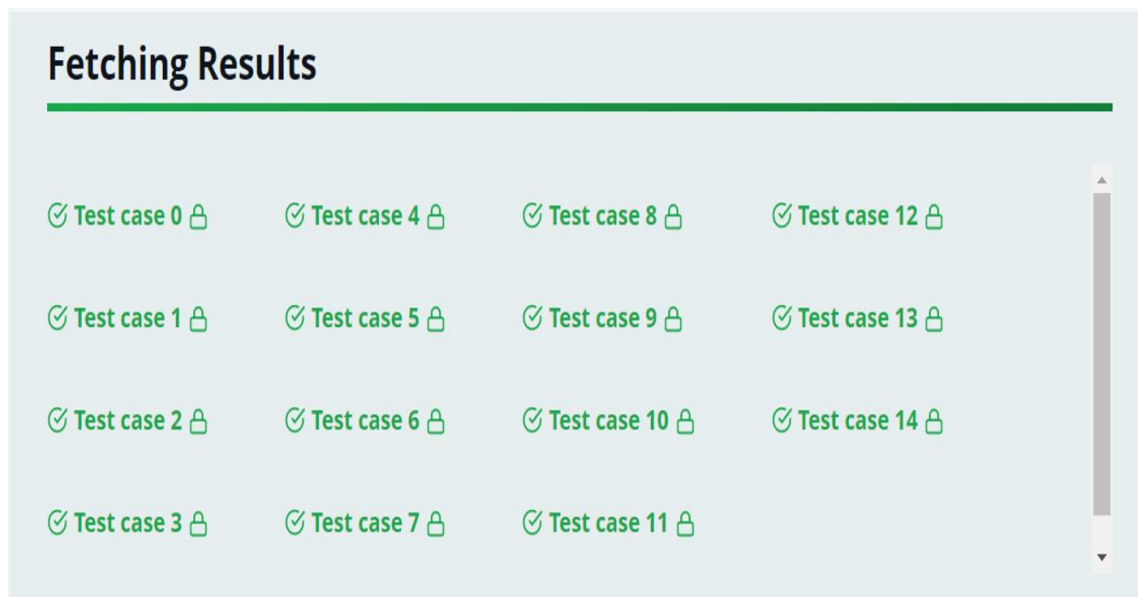



Fig 4.2-Test cases

6. LEADER BOARD

HACKER	RANK	COUNTRY ▾	SCORE
shibani2	01		50.00









HACKER	RANK	COUNTRY ▾	LANGUAGE	SCORE	
shibani2	01		C++	50.00	View solution
akshay3004	01		C++	50.00	View solution
akshaydixi	01		C++	50.00	View solution
JoeyLittleFat	01		C++	50.00	View solution
hacker_zelae7j7	01		C	50.00	View solution
wangrn	01		C++	50.00	View solution
serial_CODER	01		C++	50.00	View solution
XaCaHaa	01		C++	50.00	View solution

Fig 4.3- leaderboard

E.SIGNIFICANCE:

- Insertion sort is easy to implement and is quite efficient for small sets of data. It has low overhead and can sort the list as it receives data.
- Insertion sort needs only a constant amount of memory space for the whole operation- $O(1)$..
- It is more efficient than other similar algorithms such as bubble sort or selection sort.
- Efficient for small data sets, especially in practice than other quadratic algorithms-i.e) $O(n^2)$.
- It is a stable sorting technique, as it does not change the relative order of elements which are equal

F. CONCLUSION:

The given program code is tested using the framework. Insertion sort is used here because it is stable and it is adaptive at any cost. It is often used as recursive base case.

Insertion sort are used during,

- When the array is nearly sorted - since insertion sort is adaptive.
- When it has memory usage constraints
- When a simple sorting implementation is desired
- When the array to be sorted is relatively small
- When user need to sort elements online - that is sorting them as they come in.

PROBLEM 2: AND XOR OR

A.INTRODUCTION:

A Stack is an Abstract Data Type (ADT), commonly used in most programming languages. It is named stack as it behaves like a real-world stack. Stack allows all data operations at one end only. At any given time, the top element of the stack is accessed.

Stacks makes it LIFO data structure. LIFO stands for Last-in-first-out. Here, the element which is placed (inserted) last, is accessed first. The insertion of the element into stack is called as push operation. And removal of the element from the stack is called pop operation. Stack is also called as single linked list A stack can be implemented by means of Array, Structure, Pointer, and Linked List. Stack can either be a fixed size one or it may have a sense of dynamic resizing.

Bitwise operators are used for manipulating data at the bit level, also called bit level programming. Bitwise operates on one or more bits patterns or binary numerals at the level of their individual bits. They are used in numerical computations to make the calculation process faster. Bitwise operation operates on a bit string, a bit array or a bit string at the level of its individual bits.

A bitwise AND is a binary operation that takes two equal-length binary representations and performs the logical AND operation on each pair of the corresponding bits, which is equivalent to multiplying them. Thus, if both bits in the compared position are 1, the bit in the resulting binary representation is 1, otherwise, the result is 0.

B. OBJECTIVE OF THE PROBLEM:

The objective of the given problem is to find the maximum XOR Value from the given expression. A bitwise XOR is a binary operation that takes two bits patterns of equal length and performs the logical exclusive OR operation on each pair of corresponding bits. The result in each position is 1 if only one of the bits is 1, but will be 0 if both are 0 or both are 1. In this we perform the comparison of two bits, being 1 if the two bits are different, and 0 if they are the same.

This problem is implemented using stack.

PROBLEM STATEMENT:

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.

Your task is to find the maximum possible value of S_i .

C.DESCRPTION OF THE PROBLEM:

Input Format

First line contains integer N. Second line contains integers, representing elements of the array.

Constraints

$$1 < N \leq 10^6$$
$$1 \leq A_i \leq 10^9$$

Output Format

Print the value of maximum possible value of Si .

Sample Input

5
9 6 3 5 2

Sample Output

15

Explanation

Consider the interval [1,2] the result will be maximum.

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

FOR EXAMPLE

$$(((9 \wedge 6) \oplus (9 \vee 6)) \wedge (9 \oplus 6)) = 15$$

D.IMPLEMENTATION:

1. ALGORITHM

STEP 1: Start

STEP 2: array is declared and header for stack is taken and defining ll for long long.

STEP 3: In main function, variables as n, ans=0, val=0 are declared and initialised.

STEP 4: Reference variable of stack is created and get the array input and no of input from the user.

STEP 5: Push function is called and insertion of elements take place into the stack, before which the it checks whether the stack is empty or full using for loop.

STEP 6: While loop and if condition is used to calculate the maximum value and result is popped out from the stack.

STEP 7: The popped out result is printed

STEP 8: Stop

2. 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;

}

```

3. PROGRAM EXPLANATION

In this program, initialization of `ans=0, val, n`. Get the value of `n` and print the array elements. Push the first array element into stack and compare the array element with the stack top elements, if greater perform xor operation for `a[i] & s.top()` and store the value in `val`. If condition is initiated to check `val` is greater compared to `ans` (`val>ans`) and assign `val=ans`. if the stack is empty then push the elements and print the xor value.

4. 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) [Download](#)

1	5
2	9 6 3 5 2

Your Output (stdout)

1	15
---	----

Expected Output [Download](#)

1	15
---	----

Fig 4.4.1-Sample case

5. FINAL TEST CASES

Fetching Results

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

Fig 4.5.1- Test cases from 0-11

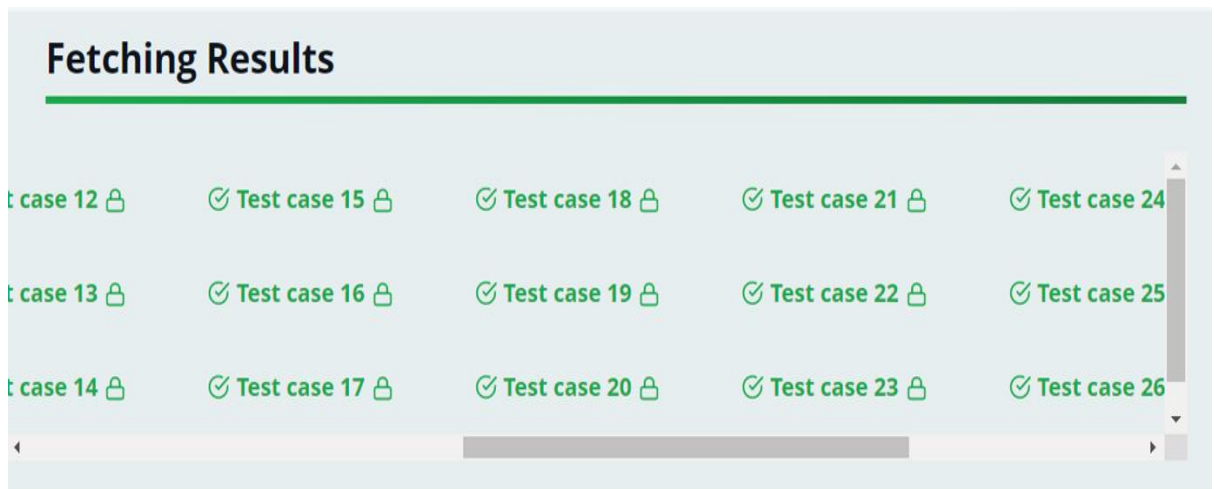


Fig 4.5.2-Test cases from 12-25

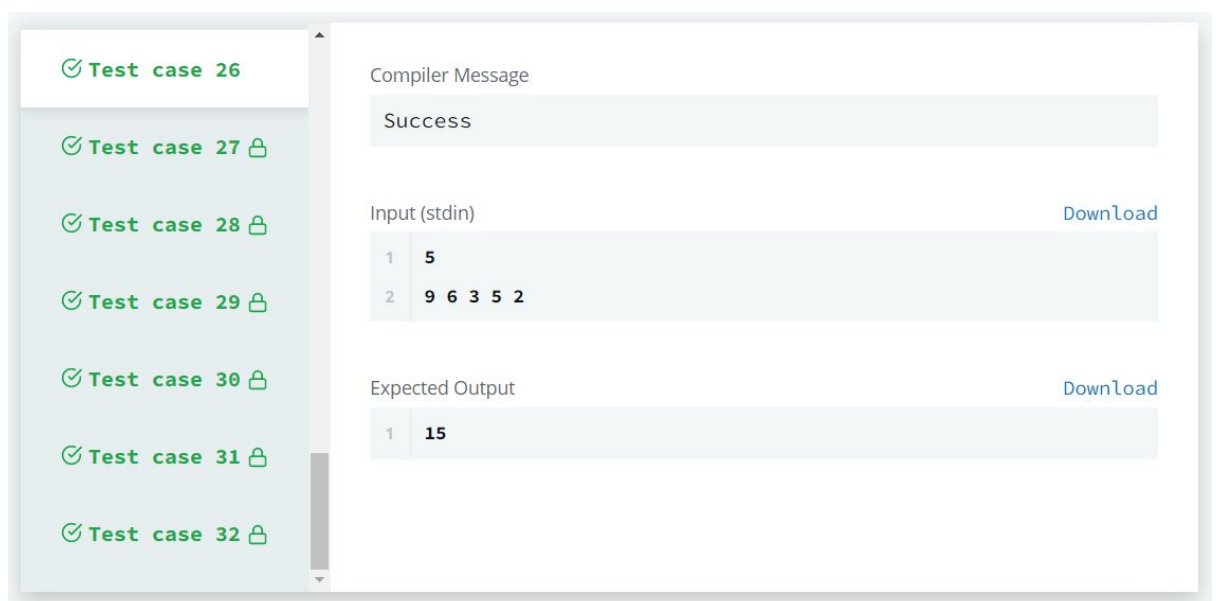


Fig 4.5.3-Test cases from 26-32

6. LEADER BOARD





HACKER	RANK	COUNTRY ▾	LANGUAGE	SCORE	
shibani2	01		C++	70.00	View solution
roy0203	01		C++	70.00	View solution
zhangsen1121	01		C++	70.00	View solution
Rikka	01		C++	70.00	View solution

Fig 4.6.1-Leader board for problem 2

5.SIGNIFICANCE:

The bitwise AND may be used to clear selected bits (or flags) of a register in which each bit represents an individual Boolean state. This technique is an efficient way to store a number of Boolean values using as little memory as possible.

A stack is an example of a linear data structure. Linear data structures are collections of components arranged in a straight line. When there is insertion or removal of components of linear data structures, they grow and shrink. It restrict the growth of a linear data structure so that new components can only be added or removed only at one end, we have a stack.

6.CONCLUSION:

Stack is a linear data structure which follows LIFO (Last in first out). The application of stack includes like backtracking, compile time memory management, efficient algorithms and expression evaluation and syntax parsing. Here if the stack is full it is called as overflow where elements can't be added. If the stack is empty it is called underflow. Stack is used for expression evaluation. Stack is an ordered list of similar data type.

Stack can be easily implemented using an Array or a Linked List. Arrays are quick, but are limited in size and Linked List requires overhead to allocate, link, unlink, and deallocate, but is not limited in size.

As given in the objective, Stack was implemented which is one of the most effective, and easier way to test all the possible outcomes.

PROJECT WORKSHEET / DIARY

WEEK 1	Date	Topics learned / Activity carried out / Task completed / Online /E-resources accessed
	04.01.2021	Discussion for selection of scheme. Programs are selected based on the scheme.
	05.01.2021	The program is splitted among the team members. Each member is assigned a task.
	06.01.2021	Finding solution for program 1 and coding part for program 1 is completed.
	07.01.2021	Solution for program 2 is discussed and test cases are passed.
	08.01.2021	Preparing report for two programs.
	09.01.2021	Completing the rest half of the report and finalizing the entire document for submission.

**Signature of the student
guide**

(with date)

Signature of the faculty

(with date)