

**RAJALAKSHMI ENGINEERING COLLEGE**

**RAJALAKSHMI NAGAR, THANDALAM – 602 105**



**RAJALAKSHMI  
ENGINEERING COLLEGE**

**CS23331**

**Design and Analysis Algorithm**

**Laboratory Observation Note Book**

[Dashboard](#) / [My courses](#) / [CS23331-DAA-2023-CSE](#) / [Divide and Conquer](#) / [1-Number of Zeros in a Given Array](#)

Started on	Friday, 30 August 2024, 1:37 PM
State	Finished
Completed on	Friday, 13 September 2024, 2:00 PM
Time taken	14 days
Marks	1.00/1.00
Grade	10.00 out of 10.00 (100%)

Question 1

Correct

Mark 1.00 out of 1.00

Problem Statement

Given an array of 1s and 0s this has all 1s first followed by all 0s. Aim is to find the number of 0s. Write a program using Divide and Conquer to Count the number of zeroes in the given array.

Input Format

- First Line Contains Integer m – Size of array
- Next m lines Contains m numbers – Elements of an array

Output Format

- First Line Contains Integer – Number of zeroes present in the given array.

Answer: (penalty regime: 0 %)

```
#include <stdio.h>
int main() {
    int m;
    scanf("%d", &m);
    int arr[m];
    for (int i = 0; i < m; i++) {
        scanf("%d", &arr[i]);
    }
    int left = 0;
    int right = m - 1;
    int firstZeroIndex = -1;
    while (left <= right) {
        int mid = left + (right - left) / 2;
        if (arr[mid] == 0) {
            firstZeroIndex = mid;
            right = mid - 1;
        } else {
            left = mid + 1;
        }
    }
    printf("%d", firstZeroIndex + 1);
}
```

	Input	Expected	Got	
✓	5 1 1 1 0 0	2	2	✓
✓	10 1 1 1 1 1 1 1 1 1 1	0	0	✓

	Input	Expected	Got	
✓	8 0 0 0 0 0 0 0 0 0	8	8	✓
✓	17 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0	2	2	✓

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.

◀ 5-G-Product of Array elements-Minimum

Jump to...

2-Majority Element ▶

[Dashboard](#) / [My courses](#) / [CS23331-DAA-2023-CSE](#) / [Divide and Conquer](#) / [2-Majority Element](#)

<b>Started on</b>	Friday, 20 September 2024, 1:31 PM
<b>State</b>	Finished
<b>Completed on</b>	Friday, 20 September 2024, 1:52 PM
<b>Time taken</b>	20 mins 25 secs
<b>Marks</b>	1.00/1.00
<b>Grade</b>	<b>10.00</b> out of 10.00 ( <b>100%</b> )

Question 1

Correct

Mark 1.00 out of 1.00

Given an array `nums` of size `n`, return *the majority element*.

The majority element is the element that appears more than  $\lfloor n / 2 \rfloor$  times. You may assume that the majority element always exists in the array.

Example 1:

Input: `nums = [3,2,3]`  
Output: `3`

Example 2:

Input: `nums = [2,2,1,1,1,2,2]`  
Output: `2`

Constraints:

- `n == nums.length`
- `1 <= n <= 5 * 104`
- `-231 <= nums[i] <= 231 - 1`

For example:

Input	Result
3 3 2 3	3
7 2 2 1 1 1 2 2	2

Answer: (penalty regime: 0 %)

```
1 #include<stdio.h>
2 int main(){
3     int n;
4     scanf("%d",&n);
5     int a[n];
6     for(int i=0;i<n;i++){
7         scanf("%d",&a[i]);
8     }
9     for(int i=0;i<n;i++){
10        int count=0;
11        for(int j=0;j<n;j++){
12            if(a[i]==a[j]){
13                count++;
14            }
15        }
16        if(count>n/2){
17            printf("%d",a[i]);
18            break;
19        }
20    }
21 }
```

	Input	Expected	Got	
✓	3 3 2 3	3	3	✓

Passed all tests! ✓

**Correct**

Marks for this submission: 1.00/1.00.

[◀ 1-Number of Zeros in a Given Array](#)

Jump to...

[3-Finding Floor Value ▶](#)

[Dashboard](#) / [My courses](#) / [CS23331-DAA-2023-CSE](#) / [Divide and Conquer](#) / [3-Finding Floor Value](#)

Started on	Friday, 20 September 2024, 1:55 PM
State	Finished
Completed on	Friday, 20 September 2024, 1:57 PM
Time taken	1 min 54 secs
Marks	1.00/1.00
Grade	10.00 out of 10.00 (100%)



Question 1

Correct

Mark 1.00 out of 1.00

Problem Statement:

Given a sorted array and a value x, the floor of x is the largest element in array smaller than or equal to x. Write divide and conquer algorithm to find floor of x.

Input Format

- First Line Contains Integer n – Size of array
- Next n lines Contains n numbers – Elements of an array
- Last Line Contains Integer x – Value for x

Output Format

- First Line Contains Integer – Floor value for x

Answer: (penalty regime: 0 %)

```
1 #include <stdio.h>
2 int Floor(int arr[], int n, int x) {
3     int low = 0, high = n - 1;
4     int floorValue = -1;
5     while (low <= high) {
6         int mid = low + (high - low) / 2;
7         if (arr[mid] == x) {
8             return arr[mid];
9         } else if (arr[mid] < x) {
10            floorValue = arr[mid];
11            low = mid + 1;
12        } else {
13            high = mid - 1;
14        }
15    }
16    return floorValue;
17 }
18 int main() {
19     int n;
20     scanf("%d", &n);
21     int arr[n];
22     for (int i = 0; i < n; i++) {
23         scanf("%d", &arr[i]);
24     }
25     int x;
26     scanf("%d", &x);
27     int result = Floor(arr, n, x);
28     if (result == -1) {
29         printf("No floor value found\n");
30     } else {
31         printf("%d\n", result);
32     }
33     return 0;
34 }
```

	Input	Expected	Got	
✓	6 1 2 8 10 12 19 5	2	2	✓

	Input	Expected	Got	
✓	5 10 22 85 108 129 100	85	85	✓
✓	7 3 5 7 9 11 13 15 10	9	9	✓

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.

◀ 2-Majority Element

Jump to...

4-Two Elements sum to x ▶

[Dashboard](#) / [My courses](#) / [CS23331-DAA-2023-CSE](#) / [Divide and Conquer](#) / [4-Two Elements sum to x](#)

Question 1

Correct

Mark 1.00 out of 1.00

Problem Statement:

Given a sorted array of integers say arr[] and a number x. Write a recursive program using divide and conquer strategy to check if there exist two elements in the array whose sum = x. If there exist such two elements then return the numbers, otherwise print as "No".  
Note: Write a Divide and Conquer Solution

Input Format

- First Line Contains Integer n – Size of array
- Next n lines Contains n numbers – Elements of an array
- Last Line Contains Integer x – Sum Value

Output Format

- First Line Contains Integer – Element1
- Second Line Contains Integer – Element2 (Element 1 and Elements 2 together sums to value "x")

Answer: (penalty regime: 0 %)

```
1 #include <stdio.h>
2 int Pair(int arr[], int left, int right, int x) {
3     if (left >= right) {
4         return 0;
5     }
6     if (arr[left] + arr[right] == x) {
7         printf("%d\n%d\n", arr[left], arr[right]);
8         return 1;
9     }
10    else if (arr[left] + arr[right] < x) {
11        return Pair(arr, left + 1, right, x);
12    }
13    else {
14        return Pair(arr, left, right - 1, x);
15    }
16 }
17 int main() {
18     int n;
19     scanf("%d", &n);
20     int arr[n];
21     for (int i = 0; i < n; i++) {
22         scanf("%d", &arr[i]);
23     }
24     int x;
25     scanf("%d", &x);
26     if (Pair(arr, 0, n - 1, x)) {
27     }
28     else {
29         printf("No\n");
30     }
31     return 0;
32 }
```

Check

	Input	Expected	Got	
✓	4 2 4 8 10 14	4 10	4 10	✓
✓	5 2 4 6 8 10 100	No	No	✓

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.

◀ 3-Finding Floor Value

Jump to...

5-Implementation of Quick Sort ▶

[Dashboard](#) / [My courses](#) / [CS23331-DAA-2023-CSE](#) / [Divide and Conquer](#) / [5-Implementation of Quick Sort](#)

<b>Started on</b>	Friday, 20 September 2024, 2:00 PM
<b>State</b>	Finished
<b>Completed on</b>	Friday, 20 September 2024, 2:04 PM
<b>Time taken</b>	4 mins 12 secs
<b>Marks</b>	1.00/1.00
<b>Grade</b>	<b>10.00</b> out of 10.00 ( <b>100%</b> )

## Question 1

Correct

Mark 1.00 out of 1.00

Write a Program to Implement the Quick Sort Algorithm

Input Format:

The first line contains the no of elements in the list-n

The next n lines contain the elements.

Output:

Sorted list of elements

For example:

Input	Result
5 67 34 12 98 78	12 34 67 78 98

Answer:

```

1  #include <stdio.h>
2  void swap(int* a, int* b) {
3      int temp = *a;
4      *a = *b;
5      *b = temp;
6  }
7  int partition(int arr[], int low, int high) {
8      int pivot = arr[high];
9      int i = low - 1;
10     for (int j = low; j < high; j++) {
11         if (arr[j] <= pivot) {
12             i++;
13             swap(&arr[i], &arr[j]);
14         }
15     }
16     swap(&arr[i + 1], &arr[high]);
17     return i + 1;
18 }
19 void quickSort(int arr[], int low, int high) {
20     if (low < high) {
21         int pi = partition(arr, low, high);
22         quickSort(arr, low, pi - 1);
23         quickSort(arr, pi + 1, high);
24     }
25 }
26 int main() {
27     int n;
28     scanf("%d", &n);
29     int arr[n];
30     for (int i = 0; i < n; i++) {
31         scanf("%d", &arr[i]);
32     }
33     quickSort(arr, 0, n - 1);
34     for (int i = 0; i < n; i++) {
35         printf("%d ", arr[i]);
36     }
37     return 0;
38 }
39

```

	Input	Expected	Got	
✓	5 67 34 12 98 78	12 34 67 78 98	12 34 67 78 98	✓

	Input	Expected	Got	
✓	10 1 56 78 90 32 56 11 10 90 114	1 10 11 32 56 56 78 90 90 114	1 10 11 32 56 56 78 90 90 114	✓
✓	12 9 8 7 6 5 4 3 2 1 10 11 90	1 2 3 4 5 6 7 8 9 10 11 90	1 2 3 4 5 6 7 8 9 10 11 90	✓

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.

◀ 4-Two Elements sum to x

Jump to...

1-DP-Playing with Numbers ▶



[Dashboard](#) / [My courses](#) / [CS23331-DAA-2023-CSE](#) / [Dynamic Programming](#) / [1-DP-Playing with Numbers](#)

Started on	Friday, 18 October 2024, 1:49 PM
State	Finished
Completed on	Friday, 18 October 2024, 2:25 PM
Time taken	36 mins 44 secs
Grade	10.00 out of 10.00 (100%)

## Question 1

Correct

Mark 10.00 out of 10.00

**Playing with Numbers:**

Ram and Sita are playing with numbers by giving puzzles to each other. Now it was Ram term, so he gave Sita a positive integer 'n' and two numbers 1 and 3. He asked her to find the possible ways by which the number n can be represented using 1 and 3. Write any efficient algorithm to find the possible ways.

**Example 1:****Input:** 6**Output:** 6**Explanation:** There are 6 ways to 6 represent number with 1 and 3

1+1+1+1+1+1

3+3

1+1+1+3

1+1+3+1

1+3+1+1

3+1+1+1

**Input Format**

First Line contains the number n

**Output Format****Print: The number of possible ways 'n' can be represented using 1 and 3**

Sample Input

6

Sample Output

6

**Answer:** (penalty regime: 0 %)

```

1  #include <stdio.h>
2  #include <stdlib.h>
3  long long count_ways(int n);
4  int main() {
5      int n;
6      scanf("%d", &n);
7      if (n < 0) {
8          printf("0\n");
9          return 1;
10     }
11     long long result = count_ways(n);
12     printf("%lld\n", result);
13     return 0;
14 }
15 long long count_ways(int n) {
16     if (n < 0) return 0;
17     long long*dp = (long long*)malloc((n + 1)* sizeof(long long));
18     if (dp == NULL) {
19         return -1;
20     }
21     dp[0] = 1;
22     dp[1] = 1;
23     dp[2] = 1;
24     dp[3] = 2;
25     for (int i = 4; i <= n; i++) {
26         dp[i] = dp[i - 1] + dp[i - 3];
27     }
28     long long result = dp[n];
29     free(dp);
30     return result ;
31 }
32

```

	Input	Expected	Got	
✓	6	6	6	✓
✓	25	8641	8641	✓
✓	100	24382819596721629	24382819596721629	✓

Passed all tests! ✓

Correct

Marks for this submission: 10.00/10.00.

◀ 5-Implementation of Quick Sort

Jump to...

2-DP-Playing with chessboard ▶

[Dashboard](#) / [My courses](#) / [CS23331-DAA-2023-CSE](#) / [Dynamic Programming](#) / [2-DP-Playing with chessboard](#)

Started on	Friday, 18 October 2024, 2:26 PM
State	Finished
Completed on	Friday, 18 October 2024, 2:37 PM
Time taken	10 mins 57 secs
Grade	10.00 out of 10.00 (100%)

## Question 1

Correct

Mark 10.00 out of 10.00

**Playing with Chessboard:**

Ram is given with an  $n \times n$  chessboard with each cell with a monetary value. Ram stands at the (0,0), that the position of the top left white rook. He is been given a task to reach the bottom right black rook position ( $n-1, n-1$ ) constrained that he needs to reach the position by traveling the maximum monetary path under the condition that he can only travel one step right or one step down the board. Help ram to achieve it by providing an efficient DP algorithm.

**Example:****Input**

```
3
1 2 4
2 3 4
8 7 1
```

**Output:**

```
19
```

**Explanation:**

Totally there will be 6 paths among that the optimal is

Optimal path value:  $1+2+8+7+1=19$

**Input Format**

First Line contains the integer  $n$

The next  $n$  lines contain the  $n \times n$  chessboard values

**Output Format**

Print Maximum monetary value of the path

**Answer:** (penalty regime: 0 %)

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 int main() {
4     int n;
5     scanf("%d",&n);
6     int **cb = (int **)malloc(n * sizeof(int *));
7     int **dp = (int **)malloc(n * sizeof(int *));
8     for (int i = 0; i < n; i++) {
9         cb[i] = (int *)malloc(n * sizeof(int));
10        dp[i] = (int *)malloc(n * sizeof(int));
11        for (int j = 0; j < n; j++) {
12            scanf("%d", &cb[i][j]);
13        }
14    }
15    dp[0][0] = cb[0][0];
16    for (int j = 1; j < n; j++)
17        dp[0][j] = dp[0][j - 1] + cb[0][j];
18    for (int i = 1; i < n; i++)
19        dp[i][0] = dp[i - 1][0] + cb[i][0];
20    for (int i = 1; i < n; i++)
21        for (int j = 1; j < n; j++)
22            dp[i][j] = cb[i][j] + (dp[i - 1][j] > dp[i][j - 1] ? dp[i - 1][j] : dp[i][j - 1]);
23    printf("%d\n", dp[n - 1][n - 1]);
24    for (int i = 0; i < n; i++) {
25        free(cb[i]);
26        free(dp[i]);
27    }
28    free(cb);
29    free(dp);
30    return 0;
31 }
32
33
```

	Input	Expected	Got	
✓	3 1 2 4 2 3 4 8 7 1	19	19	✓
✓	3 1 3 1 1 5 1 4 2 1	12	12	✓
✓	4 1 1 3 4 1 5 7 8 2 3 4 6 1 6 9 0	28	28	✓

Passed all tests! ✓

Correct

Marks for this submission: 10.00/10.00.

◀ 1-DP-Playing with Numbers

Jump to...

3-DP-Longest Common Subsequence ▶

[Dashboard](#) / [My courses](#) / [CS23331-DAA-2023-CSE](#) / [Dynamic Programming](#) / [3-DP-Longest Common Subsequence](#)

Started on	Friday, 18 October 2024, 2:40 PM
State	Finished
Completed on	Friday, 18 October 2024, 2:48 PM
Time taken	8 mins 29 secs
Marks	1.00/1.00
Grade	10.00 out of 10.00 (100%)

Question 1

Correct

Mark 1.00 out of 1.00

Given two strings find the length of the common longest subsequence(need not be contiguous) between the two.

Example:

s1: ggtabe

s2: tgatasb

s1

a

g

g

t

a

b

s2

g

x

t

x

a

y

b

The length is 4

Solveing it using Dynamic Programming

For example:

Input	Result
aab	2
azb	

Answer: (penalty regime: 0 %)

```
1 #include <stdio.h>
2 #include <string.h>
3 #define MAX_LEN 100
4 int lcsLength(const char *s1, const char *s2)
5 {
6     int len1 = strlen(s1);
7     int len2 = strlen(s2);
8     int dp[MAX_LEN + 1][MAX_LEN + 1];
9     for (int i = 0; i <= len1; i++) {
10         for (int j = 0; j <= len2; j++) {
11             if (i == 0 || j == 0) {
12                 dp[i][j] = 0;
13             } else if (s1[i - 1] == s2[j - 1]) {
14                 dp[i][j] = dp[i - 1][j - 1] + 1;
15             } else {
16                 dp[i][j] = dp[i - 1][j] > dp[i][j - 1] ? dp[i - 1][j] : dp[i][j - 1];
17             }
18         }
19     }
20     return dp[len1][len2];
21 }
22 int main() {
23     char s1[MAX_LEN + 1], s2[MAX_LEN + 1];
24     scanf("%s", s1);
25     scanf("%s", s2);
26     int result = lcsLength(s1, s2);
27     printf("%d\n", result);
28     return 0;
29 }
30
```

	Input	Expected	Got	
✓	aab azb	2	2	✓



	Input	Expected	Got	
✓	ABCD ABCD	4	4	✓

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.

◀ 2-DP-Playing with chessboard

Jump to...

4-DP-Longest non-decreasing Subsequence ▶

[Dashboard](#) / [My courses](#) / [CS23331-DAA-2023-CSE](#) / [Dynamic Programming](#) / [4-DP-Longest non-decreasing Subsequence](#)

<b>Started on</b>	Sunday, 17 November 2024, 1:13 PM
<b>State</b>	Finished
<b>Completed on</b>	Sunday, 17 November 2024, 1:15 PM
<b>Time taken</b>	1 min 46 secs
<b>Marks</b>	1.00/1.00
<b>Grade</b>	<b>10.00</b> out of 10.00 ( <b>100%</b> )

Question 1

Correct

Mark 1.00 out of 1.00

Problem statement:

Find the length of the Longest Non-decreasing Subsequence in a given Sequence.

Eg:

Input:9

Sequence:[-1,3,4,5,2,2,2,2,3]

the subsequence is [-1,2,2,2,2,3]

Output:6

Answer: (penalty regime: 0 %)

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 int longestNonDecreasingSubsequence(int seq[], int n) {
5     int dp[n];
6
7     for (int i = 0; i < n; i++) {
8         dp[i] = 1;
9     }
10
11     for (int i = 1; i < n; i++) {
12         for (int j = 0; j < i; j++) {
13             if (seq[i] >= seq[j]) {
14                 dp[i] = dp[i] > dp[j] + 1 ? dp[i] : dp[j] + 1;
15             }
16         }
17     }
18
19     int maxLength = 1;
20     for (int i = 0; i < n; i++) {
21         if (dp[i] > maxLength) {
22             maxLength = dp[i];
23         }
24     }
25
26     return maxLength;
27 }
28
29 int main() {
30     int seq[] = {-1, 3, 4, 5, 2, 2, 2, 2, 3};
31     int n = sizeof(seq) / sizeof(seq[0]);
32
33     int length = longestNonDecreasingSubsequence(seq, n);
34     printf("%d\n", length);
35
36     return 0;
37 }
38
```

	Input	Expected	Got	
✓	9 -1 3 4 5 2 2 2 2 3	6	6	✓
✓	7 1 2 2 4 5 7 6	6	6	✓

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.

[◀ 3-DP-Longest Common Subsequence](#)[1-Finding Duplicates- \$O\(n^2\)\$  Time Complexity, \$O\(1\)\$  Space Complexity ▶](#)