RAJALAKSHMI ENGINEERING COLLEGE RAJALAKSHMI NAGAR, THANDALAM – 602 105

RAJALAKSHMI
ENGINEERING COLLEGE
An AUTONOMOUS Institution
Affiliated to ANNA UNIVERSITY, Chennai

Laboratory Record Notebook

Name : GOWTHAM BR

Register No : 230701524

Branch : B.E COMPUTER SCIENCE AND ENGINEERING

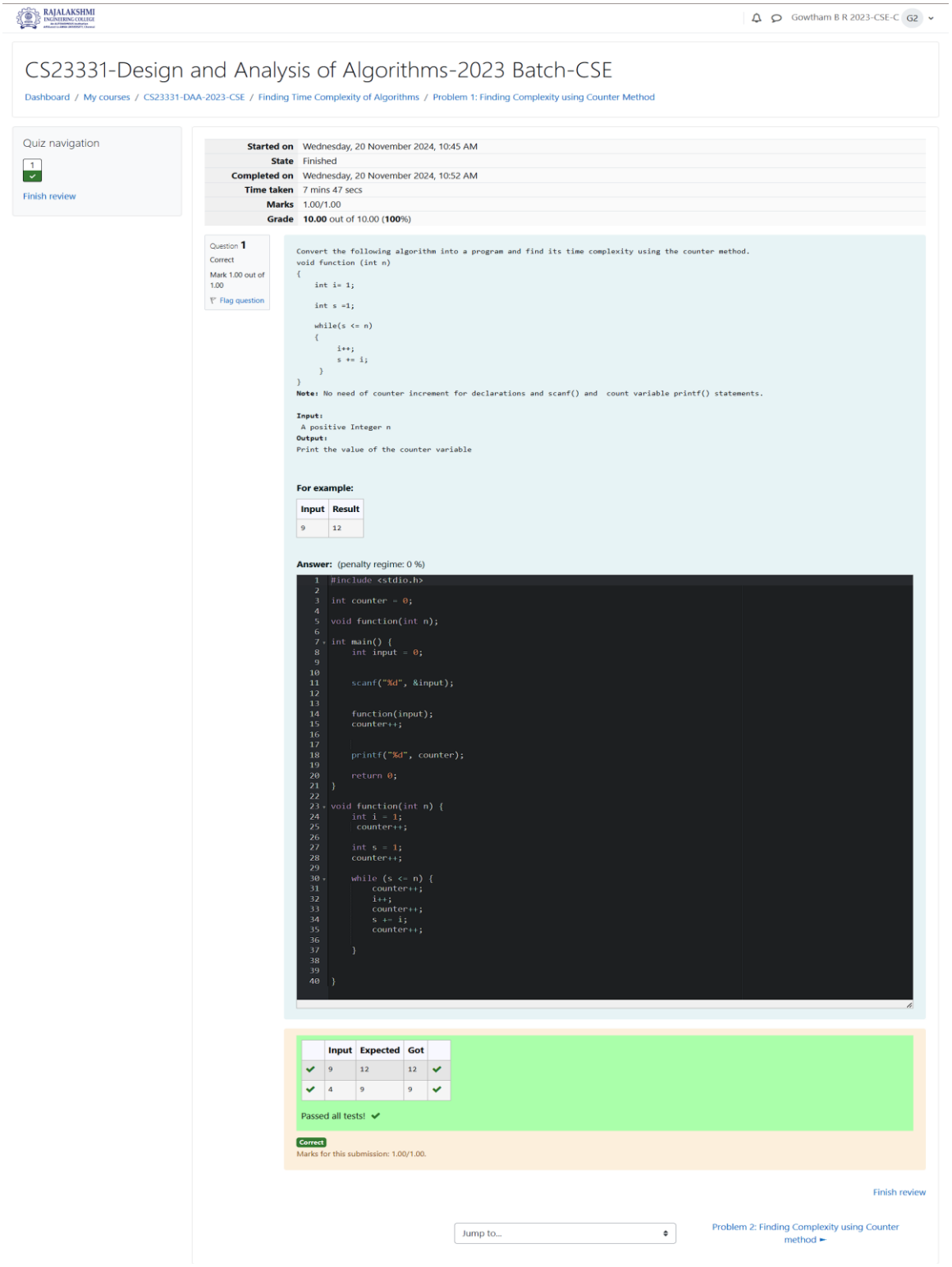
Year : II

Section : C

Semester : III

WEEK 1

FINDING TIME COMPLEXITY OF ALGORITHMS USING COUNTER METHOD



RAJALAKSHMI ENGINEERING COLLEGE

Gowtham B R 2023-CSE-C G2

CS23331-Design and Analysis of Algorithms-2023 Batch-CSE

Dashboard / My courses / CS23331-DAA-2023-CSE / Finding Time Complexity of Algorithms / Problem 1: Finding Complexity using Counter Method

Quiz navigation

1

Finish review

Started on	Wednesday, 20 November 2024, 10:45 AM
State	Finished
Completed on	Wednesday, 20 November 2024, 10:52 AM
Time taken	7 mins 47 secs
Marks	1.00/1.00
Grade	**10.00** out of 10.00 (**100**%)

Question **1**

Correct

Mark 1.00 out of 1.00

Flag question

Convert the following algorithm into a program and find its time complexity using the counter method.

```
void function (int n)
{
    int i= 1;

    int s =1;

    while(s <= n)
    {
        i++;
        s += i;
     }
}
```

Note: No need of counter increment for declarations and scanf() and count variable printf() statements.

Input:
A positive Integer n
Output:
Print the value of the counter variable

For example:

Input	Result
9	12

Answer: (penalty regime: 0 %)

```
#include <stdio.h>

int counter = 0;

void function(int n);

int main() {
    int input = 0;


    scanf("%d", &input);


    function(input);
    counter++;


    printf("%d", counter);

    return 0;
}
void function(int n) {
    int i = 1;
     counter++;

    int s = 1;
    counter++;

    while (s <= n) {
        counter++;
        i++;
        counter++;
        s += i;
        counter++;

    }


}
```

	Input	Expected	Got	
✔	9	12	12	✔
✔	4	9	9	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

Finish review

Jump to...

Problem 2: Finding Complexity using Counter method ►

CS23331-Design and Analysis of Algorithms-2023 Batch-CSE

Quiz navigation

1

Finish review

Started on	Wednesday, 20 November 2024, 10:54 AM
State	Finished
Completed on	Wednesday, 20 November 2024, 10:58 AM
Time taken	4 mins 14 secs
Marks	1.00/1.00
Grade	**10.00** out of 10.00 (**100**%)

Question **1**

Correct

Mark 1.00 out of 1.00

Flag question

Convert the following algorithm into a program and find its time complexity using counter method.

```
Factor(num) {
{
   for (i = 1; i <= num;++i)
   {
    if (num % i== 0)
      {
        printf("%d ", i);
      }
    }
 }
```

Note: No need of counter increment for declarations and scanf() and counter variable printf() statement.

Input:
A positive Integer n
Output:
Print the value of the counter variable

Answer:

```c
# include <stdio.h>

int counter = 0;

void Factor(int num){
    for(int i = 1;i<= num;i++){
        counter++;
        if(num % i == 0){
            counter++;
        }
        counter++;
    }
}

int main(){
    int n = 0;

    scanf("%d",&n);

    Factor(n);
    counter++;


    printf("%d",counter);
    return 0;
}
```

	Input	Expected	Got	
✔	12	31	31	✔
✔	25	54	54	✔
✔	4	12	12	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

Finish review

◄ Problem 2: Finding Complexity using Counter method

Jump to...

Problem 4: Finding Complexity using Counter Method ►

CS23331-Design and Analysis of Algorithms-2023 Batch-CSE

Quiz navigation

1

Finish review

Started on	Wednesday, 20 November 2024, 10:59 AM
State	Finished
Completed on	Wednesday, 20 November 2024, 11:06 AM
Time taken	7 mins 19 secs
Marks	1.00/1.00
Grade	**10.00** out of 10.00 (**100**%)

Question **1**

Correct

Mark 1.00 out of 1.00

Flag question

Convert the following algorithm into a program and find its time

complexity using counter method.

```
void function(int n)
{
    int c= 0;
    for(int i=n/2; i<n; i++)
        for(int j=1; j<n; j = 2 * j)
            for(int k=1; k<n; k = k * 2)
                c++;
}
```

Note: No need of counter increment for declarations and scanf() and count variable printf() statements.

Input:
A positive Integer n
Output:
Print the value of the counter variable

Answer:

```
#include<stdio.h>

int counter = 0;
void function(int n)
{
    int c= 0;

    for(int i=n/2; i<n; i++){
         counter++;

        for(int j=1; j<n; j = 2 * j){
             counter++;


            for(int k=1; k<n; k = k * 2){
                counter++;
                c++;
                 counter++;

        }
        counter++;
    }
    counter++;
}
counter++;
}

int main(){
    int n = 0;

    scanf("%d",&n);

    function(n);
    counter++;


    printf("%d",counter);

    return 0;
}
```

	Input	Expected	Got	
✔	4	30	30	✔
✔	10	212	212	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

Finish review

◄ Problem 3: Finding Complexity using Counter Method

Jump to...

Problem 5: Finding Complexity using counter method ►

CS23331-Design and Analysis of Algorithms-2023 Batch-CSE

Quiz navigation

1

Finish review

Started on	Wednesday, 20 November 2024, 11:07 AM
State	Finished
Completed on	Wednesday, 20 November 2024, 11:14 AM
Time taken	7 mins 39 secs
Marks	1.00/1.00
Grade	**10.00** out of 10.00 (**100**%)

Question **1**

Correct

Mark 1.00 out of 1.00

Flag question

Convert the following algorithm into a program and find its time complexity using counter method.

```
void reverse(int n)
{
   int rev = 0, remainder;
   while (n != 0)
    {
        remainder = n % 10;
        rev = rev * 10 + remainder;
        n/= 10;

    }
print(rev);
}
```

Note: No need of counter increment for declarations and scanf() and count variable printf() statements.

Input:
A positive Integer n
Output:
Print the value of the counter variable

Answer:

```c
#include <stdio.h>
int count = 0;
void reverse(int n)
{
   int rev = 0, remainder;
   count++;


   while (n != 0)
    {
         count++;
         remainder = n % 10;
         count++;


        rev = rev * 10 + remainder;
        count++;

        n/= 10;
        count++;



    }
    count++;


}
int main(){
    int n ;


    scanf("%d",&n);
    count++;
    reverse(n);
    printf("%d",count);

}
```

	Input	Expected	Got	
✔	12	11	11	✔
✔	1234	19	19	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

Finish review

WEEK-2

GREEDY ALGORITHMS

RAJALAKSHMI ENGINEERING COLLEGE

Gowtham B R 2023-CSE-C G2

CS23331-Design and Analysis of Algorithms-2023 Batch-CSE

Dashboard / My courses / CS23331-DAA-2023-CSE / Greedy Algorithms / 1-G-Coin Problem

Quiz navigation

1

Finish review

Started on	Wednesday, 20 November 2024, 11:15 AM
State	Finished
Completed on	Wednesday, 20 November 2024, 11:24 AM
Time taken	8 mins 44 secs
Marks	1.00/1.00
Grade	**10.00** out of 10.00 (**100**%)

Question **1**

Correct

Mark 1.00 out of 1.00

Flag question

Write a program to take value V and we want to make change for V Rs, and we have infinite supply of each of the denominations in Indian currency, i.e., we have infinite supply of { 1, 2, 5, 10, 20, 50, 100, 500, 1000} valued coins/notes, what is the minimum number of coins and/or notes needed to make the change.

Input Format:

Take an integer from stdin.

Output Format:

print the integer which is change of the number.

Example Input :

64

Output:

4

Explanaton:

We need a 50 Rs note and a 10 Rs note and two 2 rupee coins.

Answer: (penalty regime: 0 %)

```c

#include <stdio.h>
#include <limits.h>

int min(int a, int b) {
    return (a < b) ? a : b;
}

int minChange(int V) {
    int denominations[] = {1, 2, 5, 10, 20, 50, 100, 500, 1000};
    int n = sizeof(denominations) / sizeof(denominations[0]);
    int dp[V + 1];
    dp[0] = 0;
    for (int i = 1; i <= V; i++) {
        dp[i] = INT_MAX;
        for (int j = 0; j < n; j++) {
            if (denominations[j] <= i) {
                dp[i] = min(dp[i], dp[i - denominations[j]] + 1);
            }
        }
    }

    return dp[V];
}
int main() {
    int V;
    scanf("%d", &V);
    printf("%d\n",minChange(V));
    return 0;
}
```

	Input	Expected	Got	
✔	49	5	5	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

Finish review

◄ Problem 5: Finding Complexity using counter method

Jump to...

2-G-Cookies Problem ►

CS23331-Design and Analysis of Algorithms-2023 Batch-CSE

Quiz navigation

1

Finish review

Started on	Wednesday, 20 November 2024, 11:24 AM
State	Finished
Completed on	Wednesday, 20 November 2024, 11:30 AM
Time taken	5 mins 44 secs
Marks	1.00/1.00
Grade	**10.00** out of 10.00 (**100**%)

Question **1**

Correct

Mark 1.00 out of 1.00

Flag question

Assume you are an awesome parent and want to give your children some cookies. But, you should give each child at most one cookie.

Each child i has a greed factor g[i], which is the minimum size of a cookie that the child will be content with; and each cookie j has a size s[j]. If s[j] >= g[i], we can assign the cookie j to the child i, and the child i will be content. Your goal is to maximize the number of your content children and output the maximum number.

Example 1:

Input:

3

1 2 3

2

1 1

Output:

1

Explanation: You have 3 children and 2 cookies. The greed factors of 3 children are 1, 2, 3.

And even though you have 2 cookies, since their size is both 1, you could only make the child whose greed factor is 1 content.

You need to output 1.

Constraints:

1 <= g.length <= 3 * 10^4

0 <= s.length <= 3 * 10^4

1 <= g[i], s[j] <= 2^31 - 1

Answer: (penalty regime: 0 %)

```c
#include <stdio.h>
#include <stdlib.h>

int compare(const void* a, const void* b) {
    return (*(int*)a - *(int*)b);
}

int findContentChildren(int* g, int gSize, int* s, int sSize) {

    qsort(g, gSize, sizeof(int), compare);
    qsort(s, sSize, sizeof(int), compare);

    int i = 0, j = 0;
    int satisfied = 1;


    while (i < gSize && j < sSize) {
        if (s[j] >= g[i])
        j++;
    }{

            satisfied++;
            i++;
        }

    return satisfied;
}

int main() {
    int g[] = {1, 2, 3};
    int s[] = {1, 2};

    int gSize = sizeof(g) / sizeof(g[0]);
    int sSize = sizeof(s) / sizeof(s[0]);

    int result = findContentChildren(g, gSize, s, sSize);
    printf("%d\n", result);
}
```

	Input	Expected	Got	
✔	2 1 2 3 1 2 3	2	2	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

Finish review

◄ 1-G-Coin Problem

Jump to...

3-G-Burger Problem ►

CS23331-Design and Analysis of Algorithms-2023 Batch-CSE

Quiz navigation

1

Finish review

Started on	Wednesday, 20 November 2024, 11:30 AM
State	Finished
Completed on	Wednesday, 20 November 2024, 11:38 AM
Time taken	7 mins 16 secs
Marks	1.00/1.00
Grade	**10.00** out of 10.00 (**100**%)

Question **1**

Correct

Mark 1.00 out of 1.00

Flag question

A person needs to eat burgers. Each burger contains a count of calorie. After eating the burger, the person needs to run a distance to b
If he has eaten i burgers with c calories each, then he has to run at least $3^i * c$ kilometers to burn out the calories. For example, if
burgers with the count of calorie in the order: [1, 3, 2], the kilometers he needs to run are $(3^0 * 1) + (3^1 * 3) + (3^2 * 2) = 1 + 9 + 18$
But this is not the minimum, so need to try out other orders of consumption and choose the minimum value. Determine the minimum distance
he needs to run. Note: He can eat burger in any order and use an efficient sorting algorithm.Apply greedy approach to solve the problem.

Input Format

First Line contains the number of burgers

Second line contains calories of each burger which is n space-separate integers

Output Format

Print: Minimum number of kilometers needed to run to burn out the calories

Sample Input

```
3
5 10 7
```

Sample Output

```
76
```

For example:

Test	Input	Result
Test Case 1	3 1 3 2	18

Answer: (penalty regime: 0 %)

```c
#include<stdio.h>
#include<math.h>
int minDistance(int calories[],int n){
    for(int i=0;i<n-1;i++){
        for(int j=0;j<n-i-1;j++){
            if(calories[j]<calories[j+1]){
                int temp=calories[j];
                calories[j]=calories[j+1];
                calories[j+1]=temp;
            }
        }
    }
    int distance=0;
    for(int i=0;i<n;i++){
        distance+=pow(n,i)*calories[i];
    }
    return distance;
}
int main(){
    int n;
    scanf("%d",&n);
    int calories[n];
    for(int i=0;i<n;i++){
        scanf("%d",&calories[i]);
    }
    int min=minDistance(calories,n);
    printf("%d",min);
    return 0;
}
```

	Test	Input	Expected	Got	
✔	Test Case 1	3 1 3 2	18	18	✔
✔	Test Case 2	4 7 4 9 6	389	389	✔
✔	Test Case 3	3 5 10 7	76	76	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

Finish review

CS23331-Design and Analysis of Algorithms-2023 Batch-CSE

Quiz navigation

1

Finish review

Started on	Wednesday, 20 November 2024, 11:39 AM
State	Finished
Completed on	Wednesday, 20 November 2024, 11:53 AM
Time taken	13 mins 51 secs
Marks	1.00/1.00
Grade	**10.00** out of 10.00 (**100**%)

Question **1**

Correct

Mark 1.00 out of 1.00

Flag question

Given an array of N integer, we have to maximize the sum of arr[i] * i, where i is the index of the element (i = 0, 1, 2, ..., N).Write an algorithm based on Greedy technique with a Complexity O(nlogn).

Input Format:

First line specifies the number of elements-n

The next n lines contain the array elements.

Output Format:

Maximum Array Sum to be printed.

Sample Input:

5

2 5 3 4 0

Sample output:

40

Answer: (penalty regime: 0 %)

```c
#include <stdio.h>

void swap(int* a, int* b) {
    int temp = *a;
    *a = *b;
    *b = temp;
}

int partition(int arr[], int low, int high) {
    int pivot = arr[high];
    int i = (low - 1);

    for (int j = low; j <= high - 1; j++) {
        if (arr[j] < pivot) {
            i++;
            swap(&arr[i], &arr[j]);
        }
    }

    swap(&arr[i + 1], &arr[high]);
    return (i + 1);
}


void quickSort(int arr[], int low, int high) {
    if (low < high) {
        int pivot = partition(arr, low, high);

        quickSort(arr, low, pivot - 1);
        quickSort(arr, pivot + 1, high);
    }
}

// Function to print the array
/*void printArray(int arr[], int size) {
    for (int i = 0; i < size; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");
}
*/
int maximizeSum(int arr[], int n) {
    quickSort(arr, 0,n-1);

    int sum = 0;
    for (int i = 0; i < n; i++) {
        sum += arr[i] * i;
    }

    return sum;
}
int main() {
```

	Input	Expected	Got	
✔	5 2 5 3 4 0	40	40	✔
✔	10 2 2 2 4 4 3 3 5 5 5	191	191	✔
✔	2 45 3	45	45	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

Finish review

◄ 3-G-Burger Problem

Jump to...

5-G-Product of Array elements-Minimum ►

CS23331-Design and Analysis of Algorithms-2023 Batch-CSE

Quiz navigation

1

Finish review

Started on	Wednesday, 20 November 2024, 11:53 AM
State	Finished
Completed on	Wednesday, 20 November 2024, 12:06 PM
Time taken	13 mins 9 secs
Marks	1.00/1.00
Grade	**10.00** out of 10.00 (**100**%)

Question **1**

Correct

Mark 1.00 out of 1.00

Flag question

Given two arrays array_One[] and array_Two[] of same size N. We need to first rearrange the arrays such that the sum of the product of pairs(1 element from each) is minimum. That is SUM (A[i] * B[i]) for all i is minimum.

For example:

Input	Result
3 1 2 3 4 5 6	28

Answer: (penalty regime: 0 %)

```c
#include <stdio.h>

void sortAscending(int arr[], int n) {
    for (int i = 0; i < n - 1; i++) {
        for (int j = i + 1; j < n; j++) {
            if (arr[i] > arr[j]) {
                int temp = arr[i];
                arr[i] = arr[j];
                arr[j] = temp;
            }
        }
    }
}

void sortDescending(int arr[], int n) {
    for (int i = 0; i < n - 1; i++) {
        for (int j = i + 1; j < n; j++) {
            if (arr[i] < arr[j]) {
                int temp = arr[i];
                arr[i] = arr[j];
                arr[j] = temp;
            }
        }
    }
}

int minProductSum(int array_One[], int array_Two[], int n) {
    sortAscending(array_One, n);
    sortDescending(array_Two, n);

    int sum = 0;
    for (int i = 0; i < n; i++) {
        sum += array_One[i] * array_Two[i];
    }
    return sum;
}

int main() {
    int n;

    scanf("%d", &n);

    int array_One[n], array_Two[n];

    for (int i = 0; i < n; i++) {
        scanf("%d", &array_One[i]);
    }


    for (int i = 0; i < n; i++) {
        scanf("%d", &array_Two[i]);
    }
```

	Input	Expected	Got	
✔	3 1 2 3 4 5 6	28	28	✔
✔	4 7 5 1 2 1 3 4 1	22	22	✔
✔	5 20 10 30 10 40 8 9 4 3 10	590	590	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

Finish review

◄ 4-G-Array Sum max problem

Jump to...

1-Number of Zeros in a Given Array ►

WEEK-3

DIVIDE AND CONQURE

RAJALAKSHMI ENGINEERING COLLEGE

Gowtham B R 2023-CSE-C G2

CS23331-Design and Analysis of Algorithms-2023 Batch-CSE

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

Quiz navigation

1

Finish review

Started on	Wednesday, 20 November 2024, 12:07 PM
State	Finished
Completed on	Wednesday, 20 November 2024, 12:18 PM
Time taken	10 mins 51 secs
Marks	1.00/1.00
Grade	**10.00** out of 10.00 (**100**%)

Question **1**

Correct

Mark 1.00 out of 1.00

Flag question

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 %)

```c
#include <stdio.h>

int countZeros(int arr[], int low, int high) {

    if (low > high)
        return 0;

    if (arr[low] == 0 && arr[high] == 0)
        return high - low + 1;

     if (arr[low] == 1 && arr[high] == 1)
        return 0;

     int mid = (low + high) / 2;

     return countZeros(arr, low, mid) + countZeros(arr, mid + 1, high);
}
int main() {
    int m;


    scanf("%d", &m);

    int arr[m];


    for (int i = 0; i < m; i++) {
        scanf("%d", &arr[i]);
    }

     int zeroCount = countZeros(arr, 0, m - 1);
    printf( "%d", zeroCount);

    return 0;
}
```

	Input	Expected	Got	
✔	5 1 1 1 0 0	2	2	✔
✔	10 1 1 1 1 1 1 1 1 1 1	0	0	✔
✔	8 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 0 0 0	2	2	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

Finish review

◄ 5-G-Product of Array elements-Minimum

Jump to...

2-Majority Element ►

CS23331-Design and Analysis of Algorithms-2023 Batch-CSE

Quiz navigation

1

Finish review

Started on	Wednesday, 20 November 2024, 12:18 PM
State	Finished
Completed on	Wednesday, 20 November 2024, 12:27 PM
Time taken	9 mins 3 secs
Marks	1.00/1.00
Grade	**10.00** out of 10.00 (**100**%)

Question **1**

Correct

Mark 1.00 out of 1.00

Flag question

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

The majority element is the element that appears more than `⌊n / 2⌋` 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 * 10^4`
- `-2^31 <= nums[i] <= 2^31 - 1`

For example:

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

Answer: (penalty regime: 0 %)

```c
#include <stdio.h>

int main() {
    int n;
    scanf("%d", &n);

    int nums[n];

    for (int i = 0; i < n; i++) {
        scanf("%d", &nums[i]);
    }

    int count = 0;
    int candidate = 0;

    for (int i = 0; i < n; i++) {
        if (count == 0) {
            candidate = nums[i];
        }
        if (nums[i] == candidate) {
            count++;
        } else {
            count--;
        }
    }

    printf("%d\n", candidate);

    return 0;
}
```

	Input	Expected	Got	
✔	3 3 2 3	3	3	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

Finish review

CS23331-Design and Analysis of Algorithms-2023 Batch-CSE

Quiz navigation

1

Finish review

Started on	Wednesday, 20 November 2024, 12:27 PM
State	Finished
Completed on	Wednesday, 20 November 2024, 12:36 PM
Time taken	8 mins 6 secs
Marks	1.00/1.00
Grade	**10.00** out of 10.00 (**100**%)

Question **1**

Correct

Mark 1.00 out of 1.00

Flag question

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 %)

```c
#include <stdio.h>

int main() {
    int n, x;
    scanf("%d", &n);
    int arr[n];

    for (int i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }

    scanf("%d", &x);

    int left = 0, right = n - 1;
    int floor = -1;

    while (left <= right) {
        int mid = left + (right - left) / 2;

        if (arr[mid] == x) {
            floor = arr[mid];
            break;
        }

        if (arr[mid] < x) {
            floor = arr[mid];
            left = mid + 1;
        } else {
            right = mid - 1;
        }
    }

    if (floor != -1) {
        printf("%d\n", floor);
    } else {
        printf("No floor value found for %d in the array.\n", x);
    }

    return 0;
}

```

	Input	Expected	Got	
✔	6 1 2 8 10 12 19 5	2	2	✔
✔	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.

Finish review

◄ 2-Majority Element

Jump to...

4-Two Elements sum to x ►

CS23331-Design and Analysis of Algorithms-2023 Batch-CSE

Quiz navigation

1

Finish review

Started on	Wednesday, 20 November 2024, 12:36 PM
State	Finished
Completed on	Wednesday, 20 November 2024, 12:40 PM
Time taken	3 mins 45 secs
Marks	1.00/1.00
Grade	**10.00** out of 10.00 (**100**%)

Question **1**

Correct

Mark 1.00 out of 1.00

Flag question

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 %)

```c
#include <stdio.h>

int main() {
    int n, x;
    scanf("%d", &n);
    int arr[n];
    for (int i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }

    scanf("%d", &x);

    int left = 0, right = n - 1;
    int found = 0;

    while (left < right) {
        int sum = arr[left] + arr[right];

        if (sum == x) {

            printf("%d\n", arr[left]);
            printf("%d\n", arr[right]);
            found = 1;
            break;
        }

        if (sum < x) {
            left++;
        } else {
            right--;
        }
    }

    if (!found) {
        printf("No");
    }

    return 0;
}
```

	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.

Finish review

◄ 3-Finding Floor Value

Jump to...

5-Implementation of Quick Sort ►

CS23331-Design and Analysis of Algorithms-2023 Batch-CSE

Quiz navigation

1

Finish review

Started on	Wednesday, 20 November 2024, 12:40 PM
State	Finished
Completed on	Wednesday, 20 November 2024, 12:50 PM
Time taken	9 mins 22 secs
Marks	1.00/1.00
Grade	**10.00** out of 10.00 (**100**%)

Question **1**

Correct

Mark 1.00 out of 1.00

Flag question

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:

```c
#include <stdio.h>

int main() {
    int n;
    scanf("%d", &n);

    int a[n];


    for (int i = 0; i < n; i++) {
        scanf("%d", &a[i]);
    }


    for (int i = 0; i < n; i++) {
        for (int j = i + 1; j < n; j++) {
            if (a[j] < a[i]) {
                // Swap a[i] and a[j]
                int temp = a[i];
                a[i] = a[j];
                a[j] = temp;
            }
        }
    }


    for (int i = 0; i < n; i++) {
        printf("%d ", a[i]);
    }
    printf("\n");

    return 0;
}

```

	Input	Expected	Got	
✔	5 67 34 12 98 78	12 34 67 78 98	12 34 67 78 98	✔
✔	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.

Finish review

◄ 4-Two Elements sum to x

Jump to...

1-DP-Playing with Numbers ►

WEEK-4

DYNAMIC PROGRAMMING

RAJALAKSHMI ENGINEERING COLLEGE

Gowtham B R 2023-CSE-C G2

CS23331-Design and Analysis of Algorithms-2023 Batch-CSE

Dashboard / My courses / CS23331-DAA-2023-CSE / Dynamic Programming / 1-DP-Playing with Numbers

Quiz navigation

1

Finish review

Started on	Wednesday, 20 November 2024, 12:51 PM
State	Finished
Completed on	Wednesday, 20 November 2024, 12:56 PM
Time taken	5 mins 43 secs
Grade	**10.00** out of 10.00 (**100**%)

Question **1**

Correct

Mark 10.00 out of 10.00

Flag question

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 %)

```c
#include <stdio.h>

long long count_ways(int n) {

    long long dp[n + 1];

    dp[0] = 1;
    dp[1] = 1;
    dp[2] = 1;
    dp[3] = 2;


    for (int i = 4; i <= n; i++) {
        dp[i] = dp[i - 1] + dp[i - 3];
    }

    return dp[n];
}
int main() {
    int n;
    scanf("%d", &n);
    printf("%lld\n", count_ways(n));
    return 0;
}
```

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

Passed all tests! ✔

Correct

Marks for this submission: 10.00/10.00.

Finish review

◄ 5-Implementation of Quick Sort

Jump to...

2-DP-Playing with chessboard ►

CS23331-Design and Analysis of Algorithms-2023 Batch-CSE

Quiz navigation

1

Finish review

Started on	Wednesday, 20 November 2024, 4:59 PM
State	Finished
Completed on	Wednesday, 20 November 2024, 7:00 PM
Time taken	2 hours
Grade	**10.00** out of 10.00 (**100**%)

Question **1**

Correct

Mark 10.00 out of 10.00

Flag question

Playing with Chessboard:

Ram is given with an n*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*n chessboard values

Output Format

Print Maximum monetary value of the path

Answer: (penalty regime: 0 %)

```c
#include <stdio.h>
#define MAX 100


int max(int a, int b) {
    return (a > b) ? a : b;
}


int maxMonetaryPath(int chessboard[MAX][MAX], int n) {
    int dp[MAX][MAX];


    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            dp[i][j] = 0;
        }
    }


    dp[0][0] = chessboard[0][0];


    for (int j = 1; j < n; j++) {
        dp[0][j] = dp[0][j - 1] + chessboard[0][j];
    }


    for (int i = 1; i < n; i++) {
        dp[i][0] = dp[i - 1][0] + chessboard[i][0];
    }

    for (int i = 1; i < n; i++) {
        for (int j = 1; j < n; j++) {
            dp[i][j] = chessboard[i][j] + max(dp[i - 1][j], dp[i][j - 1]);
        }
    }


    return dp[n - 1][n - 1];
}

int main() {
    int n;
    int chessboard[MAX][MAX];


    scanf("%d", &n);


    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
```

	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.

Finish review

◄ 1-DP-Playing with Numbers

Jump to...

3-DP-Longest Common Subsequence ►

CS23331-Design and Analysis of Algorithms-2023 Batch-CSE

Quiz navigation

1

Finish review

Started on	Wednesday, 20 November 2024, 7:00 PM
State	Finished
Completed on	Wednesday, 20 November 2024, 7:40 PM
Time taken	39 mins 34 secs
Marks	1.00/1.00
Grade	**10.00** out of 10.00 (**100**%)

Question **1**

Correct

Mark 1.00 out of 1.00

Flag question

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 azb	2

Answer: (penalty regime: 0 %)

```c
#include <stdio.h>
#include <string.h>


int longest_common_subsequence(char s1[], char s2[]) {
    int m = strlen(s1);
    int n = strlen(s2);
    int dp[m + 1][n + 1];


    for (int i = 0; i <= m; i++) {
        for (int j = 0; j <= n; j++) {
            if (i == 0 || j == 0) {
                dp[i][j] = 0;
            } else if (s1[i - 1] == s2[j - 1]) {
                dp[i][j] = dp[i - 1][j - 1] + 1;
            } else {
                dp[i][j] = (dp[i - 1][j] > dp[i][j - 1]) ? dp[i - 1][j] : dp[i][j - 1];
            }
        }
    }

    return dp[m][n];
}
int main() {
    char s1[100], s2[100];


    scanf("%s", s1);
    scanf("%s", s2);


    int result = longest_common_subsequence(s1, s2);


    printf("%d", result);

    return 0;
}

```

	Input	Expected	Got	
✔	aab azb	2	2	✔
✔	ABCD ABCD	4	4	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

Finish review

CS23331-Design and Analysis of Algorithms-2023 Batch-CSE

Quiz navigation

1

Finish review

Started on	Wednesday, 20 November 2024, 7:40 PM
State	Finished
Completed on	Wednesday, 20 November 2024, 7:53 PM
Time taken	12 mins 39 secs
Marks	1.00/1.00
Grade	**10.00** out of 10.00 (**100**%)

Question **1**

Correct

Mark 1.00 out of 1.00

Flag question

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 %)

```c
#include <stdio.h>

int longest_non_decreasing_subsequence(int arr[], int n) {
    int dp[n];
    int max_length = 1;


    for (int i = 0; i < n; i++) {
        dp[i] = 1;
    }


    for (int i = 1; i < n; i++) {
        for (int j = 0; j < i; j++) {

            if (arr[j] <= arr[i]) {
                dp[i] = (dp[i] > dp[j] + 1) ? dp[i] : dp[j] + 1;
            }
        }

        if (dp[i] > max_length) {
            max_length = dp[i];
        }
    }

    return max_length;
}
int main() {
    int n;


    scanf("%d", &n);

    int arr[n];


    for (int i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }


    int result = longest_non_decreasing_subsequence(arr, n);
    printf("%d", result);

    return 0;
}

```

	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.

Finish review

◄ 3-DP-Longest Common Subsequence

Jump to...

1-Finding Duplicates-O(n^2) Time Complexity,O(1) Space Complexity ►

WEEK-5

COMPETATIVE PROGRAMMING

RAJALAKSHMI ENGINEERING COLLEGE

Gowtham B R 2023-CSE-C G2

CS23331-Design and Analysis of Algorithms-2023 Batch-CSE

Dashboard / My courses / CS23331-DAA-2023-CSE / Competitive Programming / 1-Finding Duplicates-O(n^2) Time Complexity,O(1) Space Complexity

Quiz navigation

1

Finish review

Started on	Wednesday, 20 November 2024, 7:55 PM
State	Finished
Completed on	Wednesday, 20 November 2024, 8:16 PM
Time taken	20 mins 26 secs
Marks	1.00/1.00
Grade	**4.00** out of 4.00 (**100**%)

Question **1**

Correct

Mark 1.00 out of 1.00

Flag question

Find Duplicate in Array.

Given a read only array of n integers between 1 and n, find one number that repeats.

Input Format:

First Line - Number of elements

n Lines - n Elements

Output Format:

Element x - That is repeated

For example:

Input	Result
5 1 1 2 3 4	1

Answer: (penalty regime: 0 %)

```c
#include <stdio.h>


int findDuplicate(int arr[], int n) {
    int slow = arr[0];
    int fast = arr[arr[0]];


    while (slow != fast) {
        slow = arr[slow];
        fast = arr[arr[fast]];
    }


    fast = 0;
    while (slow != fast) {
        slow = arr[slow];
        fast = arr[fast];
    }

    return slow;
}
int main() {
    int n;


    scanf("%d", &n);

    int arr[n];


    for (int i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }


    int duplicate = findDuplicate(arr, n);
    printf("%d\n", duplicate);

    return 0;
}

```

	Input	Expected	Got	
✔	11 10 9 7 6 5 1 2 3 8 4 7	7	7	✔
✔	5 1 2 3 4 4	4	4	✔
✔	5 1 1 2 3 4	1	1	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

Finish review

◄ 4-DP-Longest non-decreasing Subsequence

Jump to...

2-Finding Duplicates-O(n) Time Complexity,O(1) Space Complexity ►

CS23331-Design and Analysis of Algorithms-2023 Batch-CSE

Quiz navigation

1

Finish review

Started on	Wednesday, 20 November 2024, 8:16 PM
State	Finished
Completed on	Wednesday, 20 November 2024, 8:27 PM
Time taken	11 mins 11 secs
Marks	1.00/1.00
Grade	**4.00** out of 4.00 (**100**%)

Question **1**

Correct

Mark 1.00 out of 1.00

Flag question

Find Duplicate in Array.

Given a read only array of n integers between 1 and n, find one number that repeats.

Input Format:

First Line - Number of elements

n Lines - n Elements

Output Format:

Element x - That is repeated

For example:

Input	Result
5 1 1 2 3 4	1

Answer: (penalty regime: 0 %)

```c
#include <stdio.h>

int findDuplicate(int arr[], int n) {
    int slow = arr[0];
    int fast = arr[arr[0]];

    // Phase 1: Detect cycle
    while (slow != fast) {
        slow = arr[slow];
        fast = arr[arr[fast]];
    }

    // Phase 2: Find entry point of cycle (duplicate)
    fast = 0;
    while (slow != fast) {
        slow = arr[slow];
        fast = arr[fast];
    }

    return slow;
}

int main() {
    int n;
    scanf("%d", &n);
    int arr[n];

    for (int i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }

    int duplicate = findDuplicate(arr, n);
    printf("%d", duplicate);

    return 0;
}
```

	Input	Expected	Got	
✔	11 10 9 7 6 5 1 2 3 8 4 7	7	7	✔
✔	5 1 2 3 4 4	4	4	✔
✔	5 1 1 2 3 4	1	1	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

Finish review

◄ 1-Finding Duplicates-O(n^2) Time Complexity,O(1) Space Complexity

Jump to...

3-Print Intersection of 2 sorted arrays-O(m*n)Time Complexity,O(1) Space Complexity ►

CS23331-Design and Analysis of Algorithms-2023 Batch-CSE

Quiz navigation

1

Finish review

Started on	Wednesday, 20 November 2024, 8:28 PM
State	Finished
Completed on	Wednesday, 20 November 2024, 8:46 PM
Time taken	17 mins 51 secs
Marks	1.00/1.00
Grade	**30.00** out of 30.00 (**100**%)

Question **1**

Correct

Mark 1.00 out of 1.00

Flag question

Find the intersection of two sorted arrays.

OR in other words,

Given 2 sorted arrays, find all the elements which occur in both the arrays.

Input Format

- The first line contains T, the number of test cases. Following T lines contain:

1. Line 1 contains N1, followed by N1 integers of the first array
2. Line 2 contains N2, followed by N2 integers of the second array

Output Format

The intersection of the arrays in a single line

Example

Input:

1

3 10 17 57

6 2 7 10 15 57 246

Output:

10 57

Input:

1

6 1 2 3 4 5 6

2 1 6

Output:

1 6

For example:

Input	Result
1 3 10 17 57 6 2 7 10 15 57 246	10 57

Answer: (penalty regime: 0 %)

```c
void findIntersection(int arr1[], int n1, int arr2[], int n2) {
    int i = 0, j = 0;
    int found = 0;

    while (i < n1 && j < n2) {
        if (arr1[i] < arr2[j]) {
            i++;
        } else if (arr2[j] < arr1[i]) {
            j++;
        } else {
            // Intersection found
            printf("%d ", arr1[i]);
            found = 1;
            i++;
            j++;
        }
    }

    if (!found) {
        printf("No Intersection");
    }
    printf("\n");
}

int main() {
    int T;
    scanf("%d", &T);

    while (T--) {
        int n1, n2;


        scanf("%d", &n1);
        int arr1[n1];
        for (int i = 0; i < n1; i++) {
            scanf("%d", &arr1[i]);
        }


        scanf("%d", &n2);
        int arr2[n2];
        for (int i = 0; i < n2; i++) {
            scanf("%d", &arr2[i]);
        }


        findIntersection(arr1, n1, arr2, n2);
    }

    return 0;
}
```

	Input	Expected	Got	
✔	1 3 10 17 57 6 2 7 10 15 57 246	10 57	10 57	✔
✔	1 6 1 2 3 4 5 6 2 1 6	1 6	1 6	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

Finish review

◄ 2-Finding Duplicates-O(n) Time Complexity,O(1) Space Complexity

Jump to...

4-Print Intersection of 2 sorted arrays-O(m+n)Time Complexity,O(1) Space Complexity ►

CS23331-Design and Analysis of Algorithms-2023 Batch-CSE

Quiz navigation

1

Finish review

Started on	Wednesday, 20 November 2024, 8:46 PM
State	Finished
Completed on	Wednesday, 20 November 2024, 9:36 PM
Time taken	50 mins 14 secs
Marks	1.00/1.00
Grade	**30.00** out of 30.00 (**100**%)

Question **1**

Correct

Mark 1.00 out of 1.00

Flag question

Find the intersection of two sorted arrays.

OR in other words,

Given 2 sorted arrays, find all the elements which occur in both the arrays.

Input Format

- The first line contains T, the number of test cases. Following T lines contain:

1. Line 1 contains N1, followed by N1 integers of the first array
2. Line 2 contains N2, followed by N2 integers of the second array

Output Format

The intersection of the arrays in a single line

Example

Input:

1

3 10 17 57

6 2 7 10 15 57 246

Output:

10 57

Input:

1

6 1 2 3 4 5 6

2 1 6

Output:

1 6

For example:

Input	Result
1 3 10 17 57 6 2 7 10 15 57 246	10 57

Answer: (penalty regime: 0 %)

```c
#include <stdio.h>

void findIntersection(int arr1[], int n1, int arr2[], int n2) {
    int i = 0, j = 0;
    int found = 0;

    while (i < n1 && j < n2) {
        if (arr1[i] < arr2[j]) {
            i++;
        } else if (arr2[j] < arr1[i]) {
            j++;
        } else {

            printf("%d ", arr1[i]);
            found = 1;
            i++;
            j++;
        }
    }

    if (!found) {
        printf("No Intersection");
    }
    printf("\n");
}

int main() {
    int T;
    scanf("%d", &T);

    while (T--) {
        int n1, n2;


        scanf("%d", &n1);
        int arr1[n1];
        for (int i = 0; i < n1; i++) {
            scanf("%d", &arr1[i]);
        }


        scanf("%d", &n2);
        int arr2[n2];
        for (int i = 0; i < n2; i++) {
            scanf("%d", &arr2[i]);
        }


        findIntersection(arr1, n1, arr2, n2);
    }

    return 0;
```

	Input	Expected	Got	
✔	1 3 10 17 57 6 2 7 10 15 57 246	10 57	10 57	✔
✔	1 6 1 2 3 4 5 6 2 1 6	1 6	1 6	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

Finish review

◄ 3-Print Intersection of 2 sorted arrays-O(m*n)Time Complexity,O(1) Space Complexity

Jump to...

5-Pair with Difference-O(n^2)Time Complexity,O(1) Space Complexity ►

CS23331-Design and Analysis of Algorithms-2023 Batch-CSE

Quiz navigation

1

Finish review

Started on	Wednesday, 20 November 2024, 9:37 PM
State	Finished
Completed on	Wednesday, 20 November 2024, 10:03 PM
Time taken	26 mins 8 secs
Marks	1.00/1.00
Grade	**4.00** out of 4.00 (**100**%)

Question **1**

Correct

Mark 1.00 out of 1.00

Flag question

Given an array A of sorted integers and another non negative integer k, find if there exists 2 indices i and j such that A[j] - A[i] = k, i != j.

Input Format:

First Line n - Number of elements in an array

Next n Lines - N elements in the array

k - Non - Negative Integer

Output Format:

1 - If pair exists

0 - If no pair exists

Explanation for the given Sample Testcase:

YES as 5 - 1 = 4

So Return 1.

For example:

Input	Result
3 1 3 5 4	1

Answer: (penalty regime: 0 %)

```c
#include <stdio.h>

int findPairWithDifference(int arr[], int n, int k) {
    int i = 0, j = 1;

    while (i < n && j < n) {
        int diff = arr[j] - arr[i];

        if (i != j && diff == k) {

            return 1;
        }
        else if (diff < k) {
            j++;
        }
        else {
            i++;
        }
    }

    return 0;
}

int main() {
    int n, k;
    scanf("%d", &n);
    int arr[n];

    for (int i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }

    scanf("%d", &k);
    int result = findPairWithDifference(arr, n, k);
    printf("%d\n", result);

    return 0;
}

```

	Input	Expected	Got	
✔	3 1 3 5 4	1	1	✔
✔	10 1 4 6 8 12 14 15 20 21 25 1	1	1	✔
✔	10 1 2 3 5 11 14 16 24 28 29 0	0	0	✔
✔	10 0 2 3 7 13 14 15 20 24 25 10	1	1	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

Finish review

◄ 4-Print Intersection of 2 sorted arrays-O(m+n)Time Complexity,O(1) Space Complexity

Jump to...

6-Pair with Difference -O(n) Time Complexity,O(1) Space Complexity ►

CS23331-Design and Analysis of Algorithms-2023 Batch-CSE

Quiz navigation

1

Finish review

Started on	Wednesday, 20 November 2024, 10:03 PM
State	Finished
Completed on	Wednesday, 20 November 2024, 10:41 PM
Time taken	37 mins 16 secs
Marks	1.00/1.00
Grade	**4.00** out of 4.00 (**100**%)

Question **1**

Correct

Mark 1.00 out of 1.00

Flag question

Given an array A of sorted integers and another non negative integer k, find if there exists 2 indices i and j such that A[j] - A[i] = k, i != j.

Input Format:

First Line n - Number of elements in an array

Next n Lines - N elements in the array

k - Non - Negative Integer

Output Format:

1 - If pair exists

0 - If no pair exists

Explanation for the given Sample Testcase:

YES as 5 - 1 = 4

So Return 1.

For example:

Input	Result
3 1 3 5 4	1

Answer: (penalty regime: 0 %)

```c
#include <stdio.h>

int findPairWithDifference(int arr[], int n, int k) {
    int i = 0, j = 1;

    while (j < n) {
        int diff = arr[j] - arr[i];

        if (i != j && diff == k) {

            return 1;
        }
        else if (diff < k) {

            j++;
        }
        else {

            i++;

            if (i == j) {
                j++;
            }
        }
    }

    return 0;
}
int main() {
    int n, k;
    scanf("%d", &n);
    int arr[n];

    for (int i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }

    scanf("%d", &k);
    int result = findPairWithDifference(arr, n, k);
    printf("%d", result);

    return 0;
}

```

	Input	Expected	Got	
✔	3 1 3 5 4	1	1	✔
✔	10 1 4 6 8 12 14 15 20 21 25 1	1	1	✔
✔	10 1 2 3 5 11 14 16 24 28 29 0	0	0	✔
✔	10 0 2 3 7 13 14 15 20 24 25 10	1	1	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

Finish review

◄ 5-Pair with Difference-O(n^2)Time Complexity,O(1) Space Complexity

Jump to...