# Divide And Conquer

## 1. Number of zeroes in a given Array

**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.

```c
#include <stdio.h>
int countZeros(int arr[], int left, int right) {
    if (left > right) {
        return 0;
    }
    if (left == right) {
        return arr[left] == 0 ? 1 : 0;
    }
    int mid = (left + right) / 2;
    int leftZeros = countZeros(arr, left, mid);
    int rightZeros = countZeros(arr, mid + 1, right);
    if (arr[mid] == 1) {
        return rightZeros;
    } else {
        return leftZeros + rightZeros;
    }
}
int findZeroCount(int arr[], int size) {
    return countZeros(arr, 0, size - 1);
}

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

    return 0;
}
```

| | Input | Expected | Got | |
|---|---|---|---|---|
| ✔ | 5<br>1<br>1<br>1<br>0<br>0 | 2 | 2 | ✔ |
| ✔ | 10<br>1<br>1<br>1<br>1<br>1<br>1<br>1<br>1<br>1<br>1 | 0 | 0 | ✔ |
| ✔ | 8<br>0<br>0<br>0<br>0<br>0<br>0<br>0<br>0 | 8 | 8 | ✔ |
| ✔ | 17<br>1<br>1<br>1<br>1<br>1<br>1<br>1<br>1<br>1<br>1<br>1<br>1<br>1<br>1<br>1<br>0<br>0 | 2 | 2 | ✔ |

Passed all tests! ✔

# 2. Majority Element

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<br>3 2 3 | 3 |
| 7<br>2 2 1 1 1 2 2 | 2 |

```c
#include <stdio.h>
int countOccurrences(int arr[], int left, int right, int element) {
    int count = 0;
    for (int i = left; i <= right; i++) {
        if (arr[i] == element) {
            count++;
        }
    }
    return count;
}

int findMajorityElement(int arr[], int left, int right) {
    if (left == right) {
        return arr[left];
    }
    int mid = (left + right) / 2;
    int leftMajority = findMajorityElement(arr, left, mid);
    int rightMajority = findMajorityElement(arr, mid + 1, right);
    if (leftMajority == rightMajority) {
        return leftMajority;
    }
    int leftCount = countOccurrences(arr, left, right, leftMajority);
    int rightCount = countOccurrences(arr, left, right, rightMajority);
    int n = right - left + 1;
    if (leftCount > n / 2) {
        return leftMajority;
    } else if (rightCount > n / 2) {
        return rightMajority;
    } else {
        return -1; // No majority element
    }
}

int main() {
    int n;
    scanf("%d", &n);
    int arr[n];
    for (int i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }
    int majorityElement = findMajorityElement(arr, 0, n - 1);
    if (majorityElement != -1) {
        printf("%d", majorityElement);
    }
    return 0;
}
```

| | Input | Expected | Got | |
|---|---|---|---|---|
| ✔ | 3<br>3 2 3 | 3 | 3 | ✔ |

Passed all tests! ✔

# 3. Finding Floor value

**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

```c
#include <stdio.h>

int findFloor(int arr[], int left, int right, int key){
    if(left > right) return -1;
    if(key < arr[left]) return -1;
    if(key >= arr[right]) return arr[right];
    int mid=(left+right)/2;
    if(key == arr[mid]){
        return arr[mid];
    }else if(key > arr[mid]){
        if(mid+1 <= right && arr[mid+1] > key) return arr[mid];
        return findFloor(arr,mid+1,right,key);
    }else{
        return findFloor(arr,left,mid-1,key);
    }
}

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 ans=findFloor(arr,0,n-1,x);
    printf("%d", ans);
    return 0;
}
```

| | Input | Expected | Got | |
|---|---|---|---|---|
| ✔ | 6<br>1<br>2<br>8<br>10<br>12<br>19<br>5 | 2 | 2 | ✔ |
| ✔ | 5<br>10<br>22<br>85<br>108<br>129<br>100 | 85 | 85 | ✔ |
| ✔ | 7<br>3<br>5<br>7<br>9<br>11<br>13<br>15<br>10 | 9 | 9 | ✔ |

Passed all tests! ✔

# 4. Two elements sum to x

**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")

```c
#include <stdio.h>

void findPair(int arr[],int i,int j,int x){
    if (i>=j){
        printf("No\n");
        return;}
    int sum=arr[i]+arr[j];
    if (sum == x) {
        printf("%d\n%d",arr[i],arr[j]);
        return;}
    else if(sum<x)findPair(arr,i+1,j,x);
    else findPair(arr,i,j-1,x);}

int main(){
    int n;scanf("%d",&n);int arr[n];
    for (int i=0;i<n;i++)scanf("%d", &arr[i]);
    int x;scanf("%d", &x);
    findPair(arr,0,n-1,x);
    return 0;}
```

| | Input | Expected | Got | |
|---|---|---|---|---|
| ✔ | 4<br>2<br>4<br>8<br>10<br>14 | 4<br>10 | 4<br>10 | ✔ |
| ✔ | 5<br>2<br>4<br>6<br>8<br>10<br>100 | No | No | ✔ |

Passed all tests! ✔

# 5. Implementation of Quick Sort

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<br>67 34 12 98 78 | 12 34 67 78 98 |

```c
#include<stdio.h>
void quick(int a[],int left,int right){
    if(left<right){
        int i=left,j=right;
        int pivot=a[left];
        while(i<j){
            while(a[j]>pivot)j--;
            while(i<j&&a[i]<=pivot)i++;
            if(i<j){
                int temp=a[i];
                a[i]=a[j];
                a[j]=temp;}}
        a[left]=a[j];a[j]=pivot;
        quick(a,left,j-1);
        quick(a,j+1,right);}}
int main(){
    int a;scanf("%d",&a);int arr[a];
    for(int i=0;i<a;i++)scanf("%d",&arr[i]);
    quick(arr,0,a-1);
    for(int i=0;i<a;i++)printf("%d ",arr[i]);
}
```

| | Input | Expected | Got | |
|---|---|---|---|---|
| ✔ | 5<br>67 34 12 98 78 | 12 34 67 78 98 | 12 34 67 78 98 | ✔ |
| ✔ | 10<br>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<br>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! ✔