ICPC Assiut Community

Newcomers Training

Arrays

ICPC Assiut
community

Revision on Array 1D (Problems)

- Write a program that take an integer ***n*** that represent the size of two array of char S,T. print the first char in **S** and the first char in ***T***, then print the second char in ***S***, and second char in ***T***, and so…

(without using string)

- Write a program that take an array of integer of size N, and print the summation of the even numbers and the summation of odd numbers.

- Write a program that take an array of integer of size N, and print the summation of the prime numbers.

Sort Function

- **sort();**
  - This function sorts an array **in increasing order**
  - It generally takes two parameters :
    - **First one** is the **point of the array from where the sorting needs to begin**
    - **Second one** is the **length up to which we want the array** to get sorted.
- **Library : algorithm**
- **Complexity O(N * log2(N))**
- **Output :**

```
Array after sorting :
0 1 2 3 4 5 6 7 8 9
```

```cpp
#include <iostream>
#include <algorithm>
using namespace std;
int main()
{
    int n = 10; // size of the array
    int arr[10]={1, 5, 8, 9, 6, 7, 3, 4, 2,0};

    sort(arr, arr + n);

    /*Here we take two parameters :
    "array" : point at the beginning of the
array

    "array + n" : point at the size which you
want to sort it
    */
    cout << "Array after sorting :\n";
    for (int i = 0; i < n; ++i)
        cout << arr[i] << " ";

    return 0;
}
```

Sort Function

- **Another Ex :**
  - **Sorting from the 3rd element to the 8th element :**

```
#include <iostream>
#include <algorithm>

using namespace std;

int main()
{
    int n = 10; // size of the array
    int arr[10] = { 1, 5, 8, 9, 6, 7, 3, 4, 2, 0 };

    sort(arr + 3, arr + 8);

    cout << "Array after sorting using : \n";
    for (int i = 0; i < n; ++i)
        cout << arr[i] << " ";

    return 0;
}
```

```
1 5 8 3 4 6 7 9 2 0
```

Reverse Function

- **reverse();**
  - This function reverse the elements an array
  - It generally takes two parameters :
    - **First one** is the **point of the array from where the reversing needs to begin**
    - **Second one** is the **length up to which we want the array** to get reversed.
- **Library : algorithm**
- **Complexity O(N)**
- **Try to reverse the array without using the function**
- **Problem : write a code to sort an array in decreasing order**

```cpp
#include <iostream>
#include <algorithm>
using namespace std;

int main()
{
    int n = 10; // size of the array
    int arr[10]={1, 5, 8, 9, 6, 7, 3, 4, 2,0};

    reverse(arr, arr + n);

    cout << "Array after reversing :\n";
    for (int i = 0; i < n; ++i)
        cout << arr[i] << " ";

    return 0;
```

```
Array after reversing :
0 2 4 3 7 6 9 8 5 1
```

Frequency Array

- **What is the Frequency Array ?**
  - It's a simple Technique the consider some value as indexes of the array.

- **What is the usage of the Frequency Array ?**
  - it's use to count the frequency of numbers in the array.

- Remember that you need **an array whose size is equal to the value of the largest integer in the original array.** Which means that you can't use a frequency array if the values in the original array can be up to 10^9 for example.

- You can use a frequency array to sort an array in O(M) time, where M is the value of the largest integer in the array.

Frequency Array

Examples :

```cpp
#include<iostream>
using namespace std;
int main () {
    int Frequency[100] = {0}; // Initial all the array with 0

    int numbers[8] = {1, 2, 3, 2, 5, 6, 1, 1};

    // Doing the Frequency operation in the array "numbers"
    for(int i=0;i<8;i++){
        Frequency[ numbers[i] ] ++;
    }

    // Printing the number and it`s frequency
    for(int i=1;i<=10;i++){
        cout << i << " : " << Frequency[i] << endl;
    }

    return 0;
}
```

Output :

```
1 : 3
2 : 2
3 : 1
4 : 0
5 : 1
6 : 1
7 : 0
8 : 0
9 : 0
10 : 0
```

Problems

* You are given a string consisting of lowercase and uppercase Latin letters, Check if all the characters of the alphabet of this language appear in it *at least once*

First to solve this Problem take **(100 Points)**

Prefix Sum

- **What is the Prefix Sum ?**
  - It's a simple Technique, it's make every element in the array equal the summation of this element and all the elements before it .

- **What is the usage of the Prefix Sum ?**
  - it's help you to know the summation of the elements in the array in any range .

- In many cases, you don't need the original array after you build the prefix_sum array. In these cases, it's better to "transform" your original array into a prefix_sum array, instead of creating a separate array for the prefix sum.

Prefix Sum

Example :

```cpp
#include<iostream>
using namespace std;

int main () {
    int numbers[8] = {1, 2, 3, 2, 5, 6, 1, 1};

    // Doing the Prefix Sum operation in the array "numbers"
    for(int i = 1; i < 8; i++){
        //Note we start from index (1) not (0) to avoid the negative index
        numbers[i] = numbers[i] + numbers[i - 1];
    }

    // Printing the numbers
    for(int i = 0; i < 8; i++){
        cout << numbers[i] << " ";
    }

    return 0;
}
```

Output :
1 3 6 8 13 19 20 21

Problems

- Write a program that take ***N*** numbers from the user in array, and **print 2 Arrays**, **the First one** is a prefix sum of all the even numbers in the array, and **the Second one** is a prefix sum of all the odd numbers in the array. (the new two arrays should have the same size of the original array)

Example :

(Input)

Array = {1, 2, 3, 4, 5, 6, 7, 8}

(Output)

Even = {0, 2, 2, 6, 6, 12, 12, 20}

Odd = {1, 1, 4, 4, 9, 9, 16, 16}

Binary search

- **Binary Search**: search in a sorted array by repeatedly dividing the search **interval in half**.

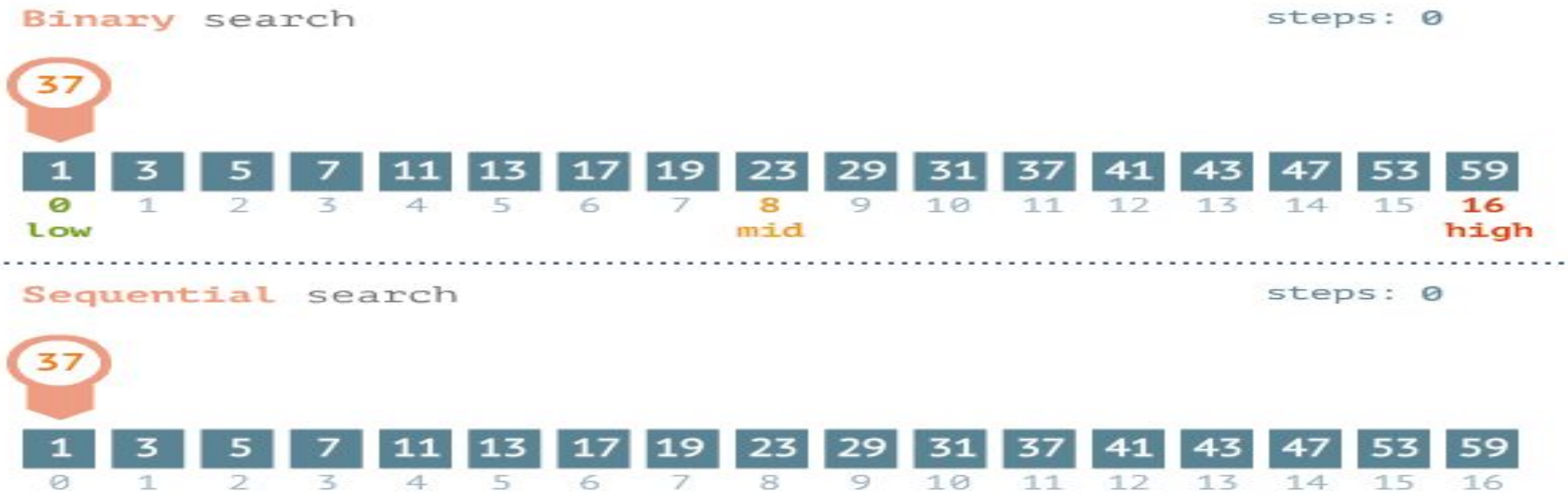
  Begin with an interval covering the whole array.

  If the value of the search key is **less than the item in the middle** of the interval, **narrow the interval to the lower half**.
  Otherwise narrow it **to the upper half**.

  Repeatedly check until the value is found or the interval is empty.

Binary search

- Given a array of ***N*** elements, write a program to search a given element ***x*** in array
- A simple approach is to do linear search. The time complexity of above algorithm **is O(n)**. Another approach to perform the same task is using Binary Search, and it's time complexity **is O(log2(n))**



Problems

- Given an array of ***N*** integers, Find if integer ***X*** **is exist in this array or not.**

Solution

For more information about **2D Arrays** visit this Link

Now it's time to practise and solve the problems of Arrays

Arrays Sheet

Good luck <3