GeeksforGeeks
A computer science portal for geeks

Practice IDE Q&A GeeksQuiz

Find a sorted subsequence of size 3 in linear time

Given an array of n integers, find the 3 elements such that a[i] < a[j] < a[k] and i < j < k in 0(n) time. If there are multiple such triplets, then print any one of them.

Examples:

```
Input:  arr[] = {12, 11, 10, 5, 6, 2, 30}
Output: 5, 6, 30

Input:  arr[] = {1, 2, 3, 4}
Output: 1, 2, 3 OR 1, 2, 4 OR 2, 3, 4

Input:  arr[] = {4, 3, 2, 1}
Output: No such triplet
```

Source: Amazon Interview Question

Hint: Use Auxiliary Space

Solution:

1) Create an auxiliary array smaller[0..n-1]. smaller[i] should store the index of a number which is smaller than arr[i] and is on left side of arr[i]. smaller[i] should contain -1 if there is no such element.

2) Create another auxiliary array greater[0..n-1]. greater[i] should store the index of a number which is greater than arr[i] and is on right side of arr[i]. greater[i] should contain -1 if there is no such element.

3) Finally traverse both smaller[] and greater[] and find the index i for which both smaller[i] and greater[i] are not -1.

C/C++

```
// C/C++ program to find a sorted subsequence of size 3
#include<stdio.h>

// A function to fund a sorted subsequence of size 3
void find3Numbers(int arr[], int n)
{
   int max = n-1; //Index of maximum element from right side
   int min = 0; //Index of minimum element from left side
   int i;

   // Create an array that will store index of a smaller
   // element on left side. If there is no smaller element
   // on left side, then smaller[i] will be -1.
   int *smaller = new int[n];
   smaller[0] = -1;  // first entry will always be -1
   for (i = 1; i < n; i++)
   {
       if (arr[i] <= arr[min])
       {
          min = i;
          smaller[i] = -1;
       }
       else
          smaller[i] = min;
   }

   // Create another array that will store index of a
   // greater element on right side. If there is no greater
   // element on right side, then greater[i] will be -1.
   int *greater = new int[n];
   greater[n-1] = -1;  // last entry will always be -1
   for (i = n-2; i >= 0; i--)
   {
       if (arr[i] >= arr[max])
       {
          max = i;
          greater[i] = -1;
       }
       else
          greater[i] = max;
   }

   // Now find a number which has both a greater number on
   // right side and smaller number on left side
   for (i = 0; i < n; i++)
   {
       if (smaller[i] != -1 && greater[i] != -1)
       {
          printf("%d %d %d", arr[smaller[i]],
                 arr[i], arr[greater[i]]);
          return;
       }
   }

   // If we reach number, then there are no such 3 numbers
   printf("No such triplet found");

   // Free the dynamically alloced memory to avoid memory leak
   delete [] smaller;
   delete [] greater;

   return;
}

// Driver program to test above function
int main()
{
    int arr[] = {12, 11, 10, 5, 6, 2, 30};
    int n = sizeof(arr)/sizeof(arr[0]);
    find3Numbers(arr, n);
    return 0;
}
```

Run on IDE

Java

```
// Java program to find a sorted subsequence of size 3
import java.io.*;

class SortedSubsequence
{
    // A function to find a sorted subsequence of size 3
    static void find3Numbers(int arr[])
    {
        int n = arr.length;
        int max = n-1; //Index of maximum element from right side
        int min = 0; //Index of minimum element from left side
        int i;

        // Create an array that will store index of a smaller
        // element on left side. If there is no smaller element
        // on left side, then smaller[i] will be -1.
        int[] smaller = new int[n];
        smaller[0] = -1;  // first entry will always be -1
        for (i = 1; i < n; i++)
        {
            if (arr[i] <= arr[min])
            {
                min = i;
                smaller[i] = -1;
            }
            else
                smaller[i] = min;
        }

        // Create another array that will store index of a
        // greater element on right side. If there is no greater
        // element on right side, then greater[i] will be -1.
        int[] greater = new int[n];
        greater[n-1] = -1;  // last entry will always be -1
        for (i = n-2; i >= 0; i--)
        {
            if (arr[i] >= arr[max])
            {
                max = i;
                greater[i] = -1;
            }
            else
                greater[i] = max;
        }

        // Now find a number which has both a greater number
        // on right side and smaller number on left side
        for (i = 0; i < n; i++)
        {
            if (smaller[i] != -1 && greater[i] != -1)
            {
                System.out.print(arr[smaller[i]]+" "+
                                 arr[i]+" "+ arr[greater[i]]);
                return;
            }
        }

        // If we reach number, then there are no such 3 numbers
```

```java
        System.out.println("No such triplet found");
        return;
    }

    public static void main (String[] args)
    {
        int arr[] = {12, 11, 10, 5, 6, 2, 30};
        find3Numbers(arr);
    }
}
/* This code is contributed by Devesh Agrawal*/
```

Run on IDE

Python

```python
# Pythion program to fund a sorted subsequence of size 3

def find3numbers(arr):
    n = len(arr)
    max = n-1 # Index of maximum element from right side
    min = 0 # Index of minimum element from left side

    # Create an array that will store index of a smaller
    # element on left side. If there is no smaller element
    # on left side, then smaller[i] will be -1.
    smaller = [0]*10000
    smaller[0] = -1
    for i in range(1,n):
        if (arr[i] <= arr[min]):
            min = i
            smaller[i] = -1
        else:
            smaller[i] = min

    # Create another array that will store index of a
    # greater element on right side. If there is no greater
    # element on right side, then greater[i] will be -1.
    greater = [0]*10000
    greater[n-1] = -1

    for i in range(n-2,-1,-1):
        if (arr[i] >= arr[max]):
            max = i
            greater[i] = -1

        else:
            greater[i] = max

    # Now find a number which has both a greater number on
    # right side and smaller number on left side
    for i in range(0,n):
        if smaller[i] != -1 and greater[i] != -1:
            print arr[smaller[i]], arr[i], arr[greater[i]]
            return

    # If we reach here, then there are no such 3 numbers
    print "No triplet found"
    return


# Driver function to test above function
```

```
arr = [12, 11, 10, 5, 6, 2, 30]
find3numbers(arr)

# This code is contributed by Devesh Agrawal
```

Run on IDE

Output:

```
5 6 30
```

Time Complexity: O(n)

Auxliary Space: O(n)

Source: How to find 3 numbers in increasing order and increasing indices in an array in linear time

Exercise:

1. Find a subsequence of size 3 such that arr[i] < arr[j] > arr[k].

2. Find a sorted subsequence of size 4 in linear time

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above

97 Comments Category: Arrays

Related Posts:

- Longest Span with same Sum in two Binary arrays
- Count Inversions of size three in a give array
- Find the subarray with least average
- Count triplets with sum smaller than a given value

- Find zeroes to be flipped so that number of consecutive 1’s is maximized
- Reorder an array according to given indexes
- Find maximum value of Sum(i*arr[i]) with only rotations on given array allowed
- Find maximum average subarray of k length

(Login to Rate and Mark)

2.5 Average Difficulty : **2.5/5.0**
Based on **4** vote(s)

□ Add to TODO List
□ Mark as DONE

Like Share 4 people like this. Be the first of your friends.

Writing code in comment? Please use code.geeksforgeeks.org, generate link and share the link here.