Longest Zig-Zag Subsequence

The longest Zig-Zag subsequence problem is to find length of the longest subsequence of given sequence such that all elements of this are alternating.

If a sequence {x1, x2, .. xn} is alternating sequence then its element satisfy one of the following relation :

```
x1 < x2 > x3 < x4 > x5 < …. xn or
x1 > x2 < x3 > x4 < x5 > …. xn
```

Examples:

```
Input: arr[] = {1, 5, 4}
Output: 3
The whole arrays is of the form  x1 < x2 > x3

Input: arr[] = {1, 4, 5}
Output: 2
All subsequences of length 2 are either of the form
x1 < x2; or x1 > x2

Input: arr[] = {10, 22, 9, 33, 49, 50, 31, 60}
Output: 6
The subsequences {10, 22, 9, 33, 31, 60} or
{10, 22, 9, 49, 31, 60} or {10, 22, 9, 50, 31, 60}
are longest Zig-Zag of length 6.
```

Recommended: Please try your approach on *{IDE}* first, before moving on to the solution.

This problem is an extension of longest increasing subsequence problem, but requires more thinking for finding optimal substructure property in this.

We will solve this problem by dynamic Programming method, Let A is given array of length n of integers. We define a 2D array Z[n][2] such that Z[i][0] contains longest Zig-Zag subsequence ending at index i and last element is greater than its previous element and Z[i][1] contains longest Zig-Zag

subsequence ending at index i and last element is smaller than its previous element, then we have following recurrence relation between them,

```
Z[i][0] = Length of the longest Zig-Zag subsequence
          ending at index i and last element is greater
          than its previous element
Z[i][1] = Length of the longest Zig-Zag subsequence
          ending at index i and last element is smaller
          than its previous element

Recursive Formulation:
   Z[i][0] = max (Z[i][0], Z[j][1] + 1);
             for all j < i and A[j] < A[i]
   Z[i][1] = max (Z[i][1], Z[j][0] + 1);
             for all j < i and A[j] > A[i]
```

The first recurrence relation is based on the fact that, If we are at position i and this element has to bigger than its previous element then for this sequence (upto i) to be bigger we will try to choose an element j (< i) such that A[j] < A[i] i.e. A[j] can become A[i]'s previous element and Z[j][1] + 1 is bigger than Z[i][0] then we will update Z[i][0].
Remember we have chosen Z[j][1] + 1 not Z[j][0] + 1 to satisfy alternate property because in Z[j][0] last element is bigger than its previous one and A[i] is greater than A[j] which will break the alternating property if we update. So above fact derives first recurrence relation, similar argument can be made for second recurrence relation also.

```
// C program to find longest Zig-Zag subsequence in
// an array
#include <stdio.h>
#include <stdlib.h>

// function to return max of two numbers
int max(int a, int b) {  return (a > b) ? a : b; }

// Function to return longest Zig-Zag subsequence length
int zzis(int arr[], int n)
{
    /*Z[i][0] = Length of the longest Zig-Zag subsequence
          ending at index i and last element is greater
          than its previous element
     Z[i][1] = Length of the longest Zig-Zag subsequence
          ending at index i and last element is smaller
          than its previous element   */
    int Z[n][2];

    /* Initialize all values from 1  */
    for (int i = 0; i < n; i++)
        Z[i][0] = Z[i][1] = 1;

    int res = 1; // Initialize result

    /* Compute values in bottom up manner */
    for (int i = 1; i < n; i++)
    {
        // Consider all elements as previous of arr[i]
        for (int j = 0; j < i; j++)
        {
            // If arr[i] is greater, then check with Z[j][1]
            if (arr[j] < arr[i] && Z[i][0] < Z[j][1] + 1)
                Z[i][0] = Z[j][1] + 1;

            // If arr[i] is smaller, then check with Z[j][0]
            if( arr[j] > arr[i] && Z[i][1] < Z[j][0] + 1)
```

```
                Z[i][1] = Z[j][0] + 1;
        }

        /* Pick maximum of both values at index i  */
        if (res < max(Z[i][0], Z[i][1]))
            res = max(Z[i][0], Z[i][1]);
    }

    return res;
}

/* Driver program */
int main()
{
    int arr[] = { 10, 22, 9, 33, 49, 50, 31, 60 };
    int n = sizeof(arr)/sizeof(arr[0]);
    printf("Length of Longest Zig-Zag subsequence is %d\n",
            zzis(arr, n) );
    return 0;
}
```

Run on IDE

Output:

```
Length of Longest Zig-Zag subsequence is 6
```

Time Complexity: $O(n^2)$

Auxiliary Space: O(n)

Below is memoization based solution of this problem.

Longest alternating subsequence

This article is contributed by Utkarsh Trivedi. Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GATE CS Corner Company Wise Coding Practice

Dynamic Programming subsequence

Recommended Posts:

Largest sum Zigzag sequence in a matrix
Dynamic Programming | Set 3 (Longest Increasing Subsequence)
Dynamic Programming | Set 17 (Palindrome Partitioning)
Longest alternating subsequence
Partition a set into two subsets such that the difference of subset sums is minimum

(Login to Rate and Mark)

3.8 Average Difficulty : **3.8/5.0**
Based on **24** vote(s)

Add to TODO List
Mark as DONE

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

Load Comments

Share this post!