Tuesday, January 4, 2022 5:55 PM

# ANARC05B - The Double HeLiX

#dynamic-programming #binary-search
Two finite, strictly increasing, integer sequences are given. Any common integer between the two sequences constitute an intersection point. Take for example the following two sequences where intersection points are

printed in bold:

- First= 3 5 **7** 9 20 **25** 30 40 **55** 56 **57** 60 62
- Second= 1 4 **7** 11 14 **25** 44 47 **55 57** 100

You can 'walk" over these two sequences in the following way:

1. You may start at the beginning of any of the two sequences. Now start moving forward.
2. At each intersection point, you have the choice of either continuing with the same sequence you're currently on, or switching to the other sequence.

The objective is finding a path that produces the maximum sum of data you walked over. In the above example, the largest possible sum is 450, which is the result of adding 3, 5, 7, 9, 20, 25, 44, 47, 55, 56, 57, 60, and 62

## Input

Your program will be tested on a number of test cases. Each test case will be specified on two separate lines. Each line denotes a sequence and is specified using the following format:

```
n v1 v2 ... vn
```

Where n is the length of the sequence and vi is the ith element in that sequence. Each sequence will have at least one element but no more than 10,000. All elements are between -10,000 and 10,000 (inclusive).

The last line of the input includes a single zero, which is not part of the test cases.

## Output

For each test case, write on a separate line, the largest possible sum that can be produced.

## Sample

```
Input:
13 3 5 7 9 20 25 30 40 55 56 57 60 62
11 1 4 7 11 14 25 44 47 55 57 100
4 -5 100 1000 1005
3 -12 1000 1001
0

Output:
450
2100
```

Submit solution!

hide comments

From <https://www.spoj.com/problems/ANARC05B/>

```java
import java.io.BufferedWriter;
import java.io.File;
import java.util.*;
import java.io.*;
class Body {
    int indexI;
    int indexJ;
    int sum;
    public void setValues(int i, int j, int sum) {
        this.indexI = i;
        this.indexJ = j;
        this.sum = sum;
    }
}
public class doubleHelix {
```

```
    public static void main(String[] args) {
        try {
            File inputFile = new File("input.txt");
            Scanner fileReader = new Scanner(inputFile);
            while (fileReader.hasNext()) {
                String lineOne[] = fileReader.nextLine().trim().split("\\s");
                int arr1[] = new int[lineOne.length];
                String lineTwo[] = fileReader.nextLine().trim().split("\\s");
                int arr2[] = new int[lineTwo.length];
                for (int i = 0; i < arr1.length; i++) {
                    arr1[i] = Integer.parseInt(lineOne[i]);
                }
                for (int i = 0; i < arr2.length; i++) {
                    arr2[i] = Integer.parseInt(lineTwo[i]);
                }
                outputMaker(arr1, arr2);
            }
            fileReader.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
    private static void outputMaker(int[] arr1, int[] arr2) throws IOException {
        try {
            FileWriter outputFile = new FileWriter("output.txt", true);
            BufferedWriter writer = new BufferedWriter(outputFile);
            DoubleHelixSolution solution = new DoubleHelixSolution();
            int maxSum = solution.maxSum(arr1, arr2);
            writer.write(Integer.toString(maxSum));
            writer.newLine();
            writer.newLine();
            writer.close();
        } catch (IOException e) {
        }
    }
}
class DoubleHelixSolution {
    public int maxSum(int arr1[], int arr2[]) {
        int m = arr1.length;
        int n = arr2.length;
        int dpLength = Math.max(m, n);
        int dp[] = new int[dpLength];
        int i, j;
        i = j = 0;
        int maxSum = 0;
        for (int k = 0; k < dp.length; k++) {
            // System.out.println();
            // System.out.println(" i : " + i + "  j : " + j);
            if (i == m || j == n) {
                break;
            }
            Body body = SumTillNextCommonPoint(arr1, arr2, i, j);
            maxSum += body.sum;
            // System.out.println(" ---- MAX SUM TILL NOW  : " + maxSum +
" -----");
            i = body.indexI;
            j = body.indexJ;
            // System.out.println(" i : " + i + "  j : " + j);
        }

        // System.out.println("maxSum " + maxSum);
        return maxSum;
    }
    public Body SumTillNextCommonPoint(
            int arr1[], int arr2[], int i, int j) {
        int sumForOne = 0;
        int sumForTwo = 0;
        while (arr1[i] == arr2[j]) {
            // System.out.println(" indide eual checker : ");
            // System.out.println(" i : " + i + " j : " + j + " arr1[i] : " +
arr1[i] + "  arr2[j]  :" + arr2[j]);
            sumForOne += arr1[i];
            sumForTwo += arr2[j];
            i++;
            j++;
        }
        // System.out.println("SumForOne :  " + sumForOne + "  Sum For Two : " +
sumForTwo);
        while ((arr1[i] != arr2[j])
        ) {
            // System.out
            //         .println("Currently :  arr[i] :" + arr1[i] + "  arr2 [j] :
" + arr2[j] + "i : " + i + "  j : " + j);
            if (arr1[i] > arr2[j]) {
                // System.out.print(arr2[j] + " -->");
                sumForTwo += arr2[j];
                j++;
            } else {
                // System.out.print(arr1[i] + " -->");
```

```
                sumForOne += arr1[i];
                i++;
            }
            if (i == arr1.length || j == arr2.length) {
                int m = arr1.length;
                int n = arr2.length;
                if (i == m) {
                    // System.out.println("helix two is pending");
                    while (j < n) {
                        // System.out.print("---"+arr2[j] + "--");
                        sumForTwo += arr2[j];
                        j++;
                    }
                } else {
                    // System.out.println("helix one is pending");

                    while (i < m) {
                        // System.out.print( "------"+arr1[i] + "----");
                        sumForOne += arr1[i];
                        i++;
                    }
                }
                break;
            }
            // System.out.println("SumForOne :  " + sumForOne + "  Sum For Two :
" + sumForTwo);
        }
        // System.out.println(" outside max loop : ");
        // if (i != arr1.length && j != arr2.length) {
        //     System.out.println(" i = " + i + "  j =" + j + "  arr1[i] = " +
arr1[i] + "  arr2 [j]" + arr2[j]);
        // }
        // System.out.println("SumForOne :  " + sumForOne + "  Sum For Two : " +
sumForTwo);
        int maxSum = Math.max(sumForOne, sumForTwo);
        // System.out.println(" MaxSum : " + maxSum);
        Body body = new Body();
        body.setValues(i, j, maxSum);
        return body;
    }
}
```