**Algorithmic Problem Solving 2021**

**17ECSE309**

**Q-Box Assignment Set**

Student Name: Pawan Kumar Singh

SRN: 01FE18BEC097

Branch: ECE

**Question 01**

Title: Minimum Stickers Needed

Level: Easy

Concepts Tested: Greedy, Two Pointers

**Problem Statement:**

Prakash Sir decided to host a contest on Hackerrank platform for all APS students. He asked the students to decide what they want as prizes for the contest based on their scores. Everyone knows Prakash Sir has a cool collection of Coding Stickers which are very dear to him, which he doesn’t share with anyone. Every student wants to get some of the stickers from his collection.

So, the students put the following conditions for the contest:

* Each student will get atleast 1 sticker.
* Sudent which solves more questions than its neighbours gets more stickers than the neighbours.

However, Prakash Sir really likes to collect cool coding stickers and wants to minimize the number of stickers he will have to give at the end of the contest.

Given the number of students and their scores in the contest, find the minimum number of stickers which Prakash Sir has to give away.

**Input Format:**

The input contains of two lines. The first line contains a single value N, denoting the number of students in APS class. The next line contains N space seperated values denoting the score of the students.

**Constraints:**

1 <= N <= 100000

1 <= score[i] <= 1000

**Output Format:**

Print a single integer which is the minimum number of stickers Prakash Sir has to give away.

**Solution:**

(Solution in C++)

#include<bits/stdc++.h>

using namespace std;

// Funtion to find minimum stickers to be given

int findMinimumStickers(vector<int> scores, int n) {

vector<int>left(n, 1);

vector<int>right(n, 1);

// If the score of current student is greater than its left neighbour

for (int i = 1; i < n; i++) {

if (scores[i] > scores[i - 1]) left[i] = left[i - 1] + 1;

}

// If the score of current student is greater than its right neighbour

for (int i = n - 2; i >= 0; i--) {

if (scores[i] > scores[i + 1]) right[i] = right[i + 1] + 1;

}

int ans = 0;

// We need to get the maximum of both values as we have to satisfy

// both left and right neighbour conditions.

for (int i = 0; i < n; i++) {

ans += max(left[i], right[i]);

}

return ans;

}

int main() {

int n;

cin >> n;

vector<int>scores(n);

for (int i = 0; i < n; i++) cin >> scores[i];

cout << findMinimumStickers(scores, n) << endl;

return 0;

}

**Sample Test Cases:**

**Input -**

3

2 1 3

**Output -**

5

**Input -**

8

12 4 3 11 34 34 1 67

**Output -**

16

**Test Cases:**

**Input -**

10

1 2 3 4 5 5 6 7 10 25

**Output -**

30

**Input -**

12

9 7 12 33 4 553 212 122 434 345 34 553

**Output -**

23

**Input -**

10

233 123 15 456 345 654 355 346 643 674

**Output -**

20

**Input -**

8

12 4 3 11 34 34 1 67

**Output -**

16

**Input -**

5

131 1234 4542 144 355

**Output -**

9

**Input -**

12

3432 4254 3446 1231 3455 12455 4131 53411 41345 5543 2134 3541

**Output -**

25

**Input -**

3

2 1 3

**Output -**

5

**Output -**

16

**Input -**

10

2 5 8 3 6 4 6 6 4 8

**Output -**

17

**Input -**

15

17 14 9 16 43 52 8 0 76 53 64 67 29 73 11

**Output -**

30

**Input -**

5

9 9 9 9 9

**Output -**

5

**Question 02**

Title: Tough Challenge

Level: Medium

Concepts Tested: Recursion, Dynamic Programming

**Problem Statement:**

Prakash Sir has challenged Pawan for a game. He gives Pawan two not necessarily distinct string str1 and str2. He challenges Pawan to convert str1 into str2 in minimum number of moves.

In each move, Pawan is allowed to perform one of the following operations :

* Insert a character in str1
* Remove a character from str1
* Replace a character in str1 to any other character

Find minimum number of moves Pawan requires to convert str1 to str2.

**Input Format:**

The first and only line contains two space seperated strings str1 and str2

**Constraints:**

0 <= str1.length() <= 100

0 <= str2.length() <= 100

**Output Format:**

Print a single integer denoting the minimum number of moves required to convert str1 to str2.

**Solution:**

//(C++ top-down approach)

#include<bits/stdc++.h>

using namespace std;

int solve(string str1, string str2, int m, int n) {

// Memoization table

vector < vector<int>>dp(m + 1, vector<int>(n + 1));

for (int i = 0; i <= m; i++) {

for (int j = 0; j <= n; j++) {

// If str1 is an empty string

if (i == 0) dp[i][j] = j;

//if str2 is an empty string

else if (j == 0) dp[i][j] = i;

else {

// If the current characters of both the strings match (no need to perform any operation)

if (str1[i - 1] == str2[j - 1]) {

dp[i][j] = dp[i - 1][j - 1];

}

else {

// 1 + min(Insert, Remove, Replace)

dp[i][j] = 1 + min(dp[i][j - 1], min(dp[i - 1][j], dp[i - 1][j - 1]));

}

}

}

}

return dp[m][n];

}

int main() {

string str1, str2;

cin >> str1 >> str2;

cout << solve(str1, str2, str1.length(), str2.length());

return 0;

}

**Sample Test Cases:**

**Input -**

“abc” “bcd”

**Output -**

2

**Test Cases:**

**Input -**

“abc” “bcd”

**Output -**

2

**Input -**

“abc” “cba”

**Output -**

2

**Input -**

“” “sdnasjkndfjdnflanlnfoiedqdlkdadnafnjdfndlknfaldnaldnamsakfjdnmkjebffhewohfifoajflakf”

**Output -**

84

**Input -**

“adjffoiflkrlfkfnkdmadadlankds” “sdnasjkndfjdnflanlnfoiedqdlkdadnafnjdfndlknfaldnaldnamsakfjdnmkjebffhewohfifoajflakf”

**Output -**

64

**Input -**

“aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa”

“bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb”

**Output -**

93

**Input -**

“abcdefghijklrstuvwxyz” “abcdefghijklmnopqrstuvwxyz”

**Output -**

5

**Input -**

“abcdefghijklmnopqrstuvwxyz” “abcdefghijklmnopqrstuvwxyz”

**Output -**

0

**Input -**

“abcdefghijklmnopqrstuvwxyz” “”

**Output -**

26

**Input -**

“erofhowefffmiwneendasldlafnoifnoafnmdmmdfnn” “ownfenkffnpjienl”

**Output -**

34

**Input -**

“abcdefghijkl” “ancfgl”

**Output -**

7