

Assignment 8 (Advance Programming)

Name – Paras Aggarwal

UID – 22BCS16963

1827. Minimum Operations to Make the Array Increasing

You are given an integer array `nums` (0-indexed). In one operation, you can choose an element of the array and increment it by 1.

For example, if `nums = [1,2,3]`, you can choose to increment `nums[1]` to make `nums = [1,3,3]`.

Return the minimum number of operations needed to make `nums` strictly increasing.

An array `nums` is strictly increasing if `nums[i] < nums[i+1]` for all $0 \leq i < \text{nums.length} - 1$. An array of length 1 is trivially strictly increasing.

Example 1:

Input: `nums = [1,1,1]`

Output: 3

Explanation: You can do the following operations:

- 1) Increment `nums[2]`, so `nums` becomes `[1,1,2]`.
- 2) Increment `nums[1]`, so `nums` becomes `[1,2,2]`.
- 3) Increment `nums[2]`, so `nums` becomes `[1,2,3]`.

Solution:

```
class Solution {  
public:  
    int minOperations(vector<int>& nums) {  
        int operations = 0;
```

```

        for (int i = 1; i < nums.size(); ++i) {
            if (nums[i] <= nums[i - 1]) {
                operations += nums[i - 1] - nums[i] + 1;
                nums[i] = nums[i - 1] + 1;
            }
        }

        return operations;
    }
};

```

The screenshot shows the LeetCode interface for problem 1827. On the left, the problem description states: "You are given an integer array `nums` (0-indexed). In one operation, you can choose an element of the array and increment it by 1." It provides an example: if `nums = [1, 2, 3]`, you can increment `nums[1]` to make `nums = [1, 3, 3]`. The goal is to return the minimum number of operations to make `nums` strictly increasing. The problem is marked as "Solved" and "Easy".

On the right, the C++ code is displayed in a dark-themed editor. It defines a function `minOperations` that takes a vector of integers `nums` and returns an integer. The code implements the logic shown in the previous code block, using a loop to iterate through the array and calculate the required operations.

Below the code, the "Testcase" tab is selected, showing "Accepted" status with a runtime of 0 ms. The input for Case 1 is `nums = [1, 1, 1]`.

1717. Maximum Score From Removing Substrings

You are given a string `s` and two integers `x` and `y`. You can perform two types of operations any number of times.

Remove substring "ab" and gain `x` points.

For example, when removing "ab" from "cabxbae" it becomes "cxbae".

Remove substring "ba" and gain `y` points.

For example, when removing "ba" from "cabxbae" it becomes "cabxe".

Return the maximum points you can gain after applying the above operations on s.

Example 1:

Input: s = "cdbcbbaaabab", x = 4, y = 5

Output: 19

Explanation:

- Remove the "ba" underlined in "cdbcbbaaabab". Now, s = "cdbcbbaaab" and 5 points are added to the score.
- Remove the "ab" underlined in "cdbcbbaaab". Now, s = "cdbcbbaa" and 4 points are added to the score.
- Remove the "ba" underlined in "cdbcbbaa". Now, s = "cdbcba" and 5 points are added to the score.
- Remove the "ba" underlined in "cdbcba". Now, s = "cdabc" and 5 points are added to the score.

Total score = 5 + 4 + 5 + 5 = 19.

Solution:

```
class Solution {
public:
    int maximumGain(string s, int x, int y) {
        if (y > x) {
            swap(x, y);
            for (char &c : s) {
                if (c == 'a') c = 'b';
                else if (c == 'b') c = 'a';
            }
        }
        int score = 0;
        stack<char> st;
        string remaining;
```

```
    for (char c : s) {  
        if (!st.empty() && st.top() == 'a' && c == 'b') {  
            st.pop();  
            score += x;  
        } else {  
            st.push(c);  
        }  
    }  
    while (!st.empty()) {  
        remaining += st.top();  
        st.pop();  
    }  
    reverse(remaining.begin(), remaining.end());  
    for (char c : remaining) {  
        if (!st.empty() && st.top() == 'b' && c == 'a') {  
            st.pop();  
            score += y;  
        } else {  
            st.push(c);  
        }  
    }  
    return score;  
}  
};
```

Description

Accepted

Editorial

Solutions

Submissions

1717. Maximum Score From Removing Substrings

Solved

Medium

Topics Companies Hint

You are given a string `s` and two integers `x` and `y`. You can perform two types of operations any number of times.

- Remove substring `"ab"` and gain `x` points.
For example, when removing `"ab"` from `"cabxbae"` it becomes `"cxbae"`.
- Remove substring `"ba"` and gain `y` points.
For example, when removing `"ba"` from `"cabxbae"` it becomes `"cabxe"`.

Return the maximum points you can gain after applying the above operations on `s`.

Example 1:

Input: `s = "cdbcbbaaabab"`, `x = 4`, `y = 5`
Output: 19

1.4K 205 21 Online

Code

```

1 class Solution {
2 public:
3     int maximumGain(string s, int x, int y) {
4         if (y > x) {
5             swap(x, y);
6             for (char &c : s) {
7                 if (c == 'a') c = 'b';
8                 else if (c == 'b') c = 'a';
9             }
10        }
11
12        int score = 0;
13        startercharry et...

```

Ln 15, Col 1 Saved

Run Submit

Testcase

Test Result

Accepted Runtime: 0 ms

Case 1 Case 2

Input

`s = "cdbcbbaaabab"`

1713. Minimum Operations to Make a Subsequence

You are given an array `target` that consists of distinct integers and another integer array `arr` that can have duplicates.

In one operation, you can insert any integer at any position in `arr`. For example, if `arr = [1,4,1,2]`, you can add 3 in the middle and make it `[1,4,3,1,2]`. Note that you can insert the integer at the very beginning or end of the array.

Return the minimum number of operations needed to make `target` a subsequence of `arr`.

A subsequence of an array is a new array generated from the original array by deleting some elements (possibly none) without changing the remaining elements' relative order. For example, `[2,7,4]` is a subsequence of `[4,2,3,7,2,1,4]` (the underlined elements), while `[2,4,2]` is not.

Example 1:

Input: `target = [5,1,3]`, `arr = [9,4,2,3,4]`

Output: 2

Explanation: You can add 5 and 1 in such a way that makes `arr = [5,9,4,1,2,3,4]`, then `target` will be a subsequence of `arr`.

Solution:

```
class Solution {
```

```
public:
```

```
    int minOperations(vector<int>& target, vector<int>& arr) {
```

```

unordered_map<int, int> indexMap;

for (int i = 0; i < target.size(); ++i) {
    indexMap[target[i]] = i;
}

vector<int> lisSequence;

for (int num : arr) {
    if (indexMap.find(num) != indexMap.end()) {
        int idx = indexMap[num];
        auto it = lower_bound(lisSequence.begin(), lisSequence.end(), idx);
        if (it == lisSequence.end()) {
            lisSequence.push_back(idx);
        } else {
            *it = idx;
        }
    }
}

return target.size() - lisSequence.size();
}

};

```

DescriptionAccepted X EditorialSolutionsSubmissions

1713. Minimum Operations to Make a Subsequence

Solved

HardTopicsCompaniesHint

You are given an array `target` that consists of **distinct** integers and another integer array `arr` that **can** have duplicates.

In one operation, you can insert any integer at any position in `arr`. For example, if `arr = [1,4,1,2]`, you can add `3` in the middle and make it `[1,4,3,1,2]`. Note that you can insert the integer at the very beginning or end of the array.

Return the **minimum** number of operations needed to make `target` a **subsequence** of `arr`.

A **subsequence** of an array is a new array generated from the original array by deleting some elements (possibly none) without changing the remaining elements' relative order. For example, `[2,7,4]` is a subsequence of `[4,2,3,7,2,1,4]` (the underlined elements), while `[2,4,2]` is not.

Example 1:

7439☆📌🔖

7 Online

Code

C++Auto

```
1 class Solution {
2 public:
3     int minOperations(vector<int>& target, vector<int>& arr) {
4         unordered_map<int, int> indexMap;
5         for (int i = 0; i < target.size(); ++i) {
6             indexMap[target[i]] = i;
7         }
8
9         vector<int> lisSequence;
10
11         for (int num : arr) {
12             if (indexMap.find(num) != indexMap.end()) {
13                 int idx = indexMap[num];
14             }
15         }
16     }
17 }
```

Ln 23, Col 51 | Saved

RunSubmit

TestcaseTest Result

AcceptedRuntime: 0 ms

Case 1

Case 2

Input

target =
[5, 1, 3]