

AP Assignment 8

**aliza asif**

22BCS50175

22BCS\_FL\_IOT\_601\_A

**AP ASSIGNMENT 8**

**Q1.** **Minimum Operations to Make the Array Increasing**

Implementation Code:

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;

}

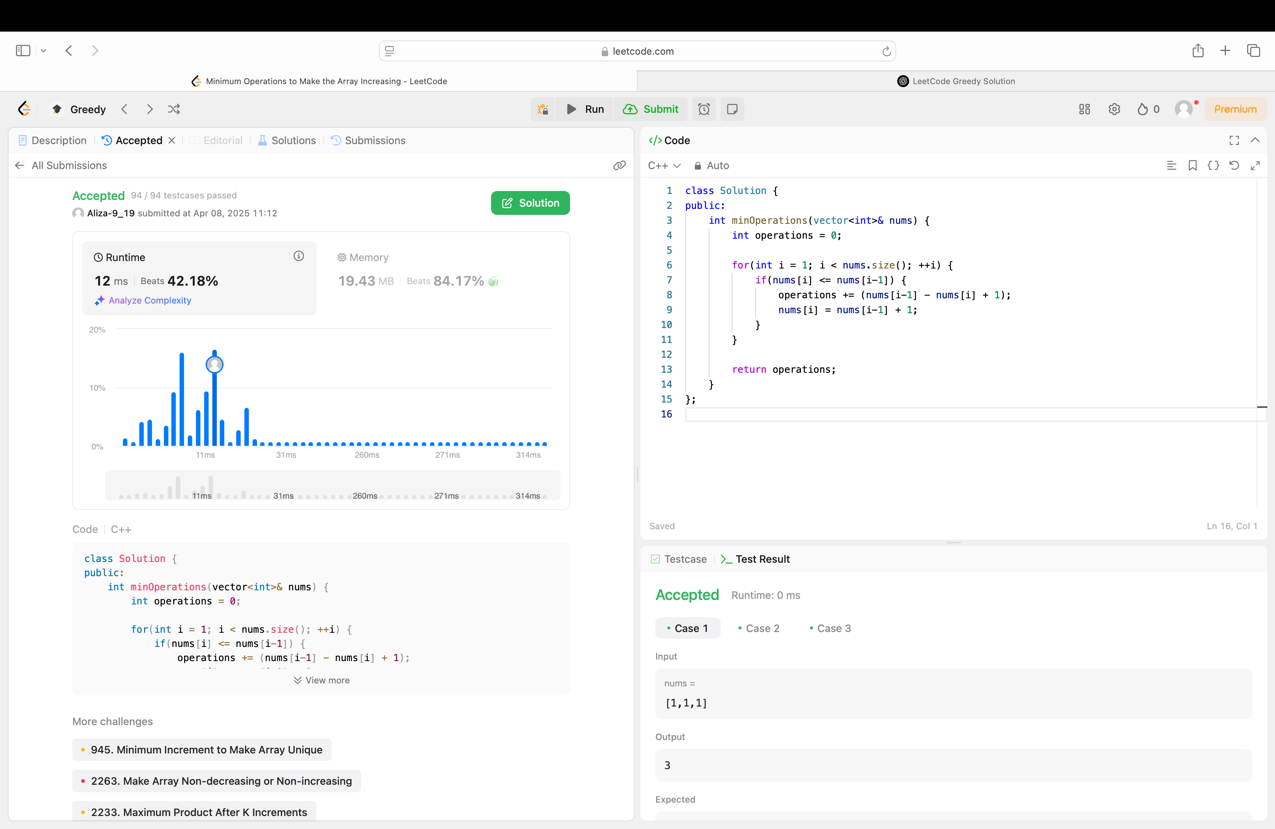
}

return operations;

}

};

Output:



**Q2. Maximum Score From Removing Substrings**

Implementation Code:

class Solution {

public:

int removePair(string& s, char first, char second, int score) {

stack<char> st;

int total = 0;

for (char c : s) {

if (!st.empty() && st.top() == first && c == second) {

st.pop();

total += score;

} else {

st.push(c);

}

}

s = "";

while (!st.empty()) {

s = st.top() + s;

st.pop(); }

return total; }

int maximumGain(string s, int x, int y) {

int totalScore = 0;

if (x > y) {

totalScore += removePair(s, 'a', 'b', x);

totalScore += removePair(s, 'b', 'a', y);

} else {

totalScore += removePair(s, 'b', 'a', y);

totalScore += removePair(s, 'a', 'b', x);

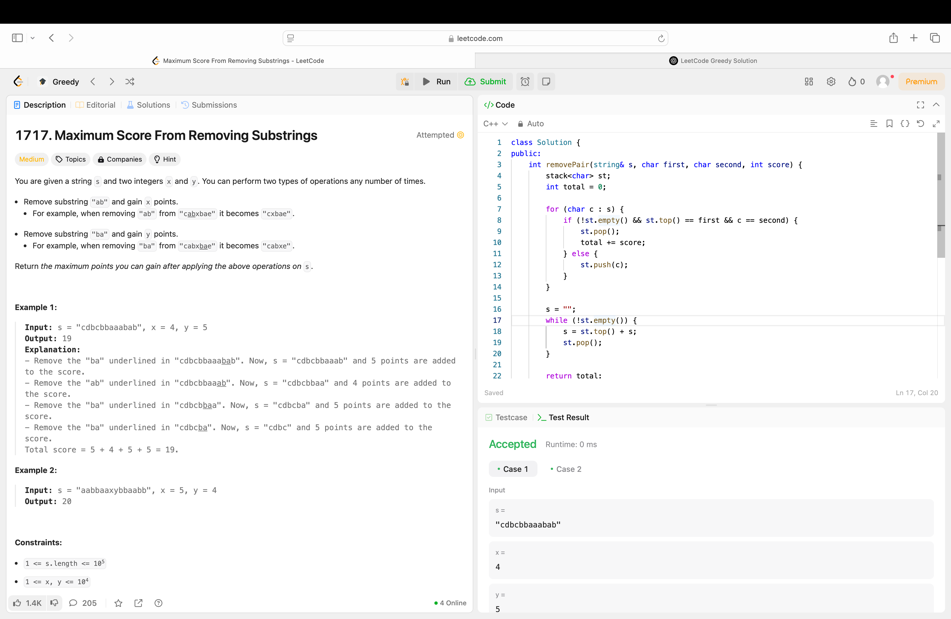
}

return totalScore;

}

};

Output:



**Q3. Minimum Operations to Make a Subsequence**

Implementation Code:

class Solution {

public:

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

unordered\_map<int, int> pos;

for (int i = 0; i < target.size(); ++i) {

pos[target[i]] = i;

}

vector<int> indexSeq;

for (int num : arr) {

if (pos.find(num) != pos.end()) {

indexSeq.push\_back(pos[num]);

}

}

vector<int> lis;

for (int index : indexSeq) {

auto it = lower\_bound(lis.begin(), lis.end(), index);

if (it == lis.end()) lis.push\_back(index);

else \*it = index;

}

return target.size() - lis.size();

}

};

Output:

