Assignment - 8

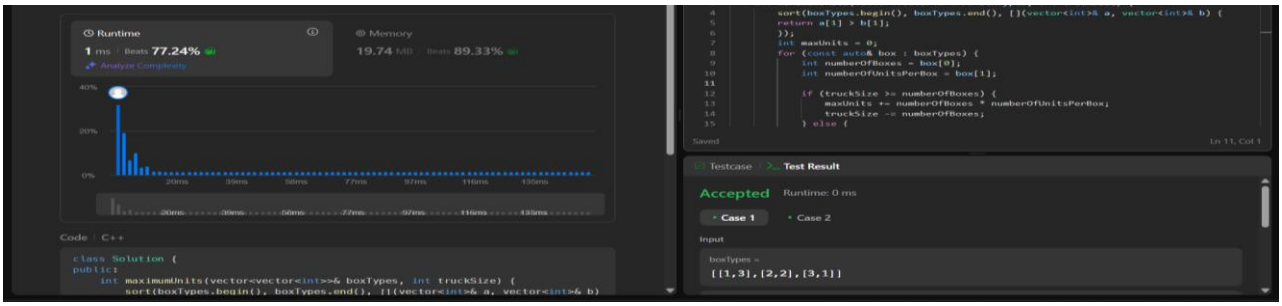
Student Name: Nikhil Kumar Tiwari
UID: 22BCS10471
Branch :BE-CSE
Section/Group:22BCS-IOT-FL-601 A
Semester: 6 th
Subject Code: 22CSP-351
Subject Name: Advanced Programming Lab- 2

Problem 1: Maximum Units on a Truck (https://leetcode.com/problems/maximum-units-on-a-truck/

Code:

```
class Solution {
public:
  int maximumUnits(vector<vector<int>>& boxTypes, int truckSize) {
    sort(boxTypes.begin(), boxTypes.end(), [](vector<int>& a, vector<int>& b) {
    return a[1] > b[1];
    });
    int maxUnits = 0;
    for (const auto& box : boxTypes) {
      int numberOfBoxes = box[0];
      int numberOfUnitsPerBox = box[1];
      if (truckSize >= numberOfBoxes) {
        maxUnits += numberOfBoxes * numberOfUnitsPerBox;
        truckSize -= numberOfBoxes;
      } else {
        maxUnits += truckSize * numberOfUnitsPerBox;
        break;
      }}
    return maxUnits;
  };
```

Screenshot:

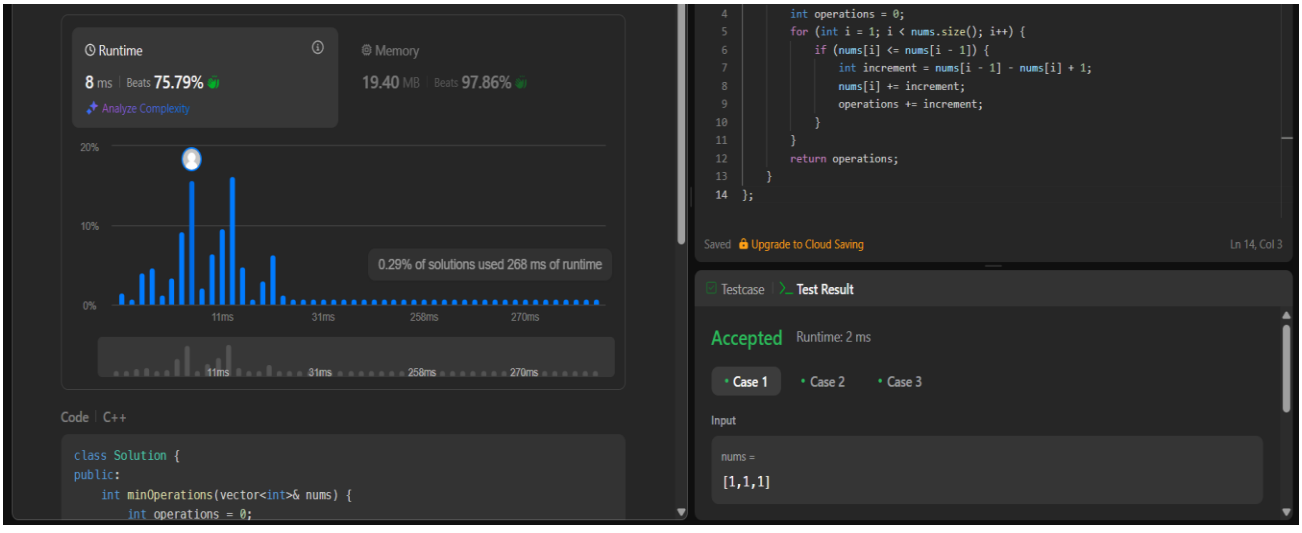


Problem 2: Minimum Operations to Make the Array (https://leetcode.com/problems/minimum-operations-to-make-the-array-increasing/ **)**

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]) {

        int increment = nums[i - 1] - nums[i] + 1;

        nums[i] += increment;

        operations += increment;

      }

    }

    return operations;

  }

};
```

Screenshot:

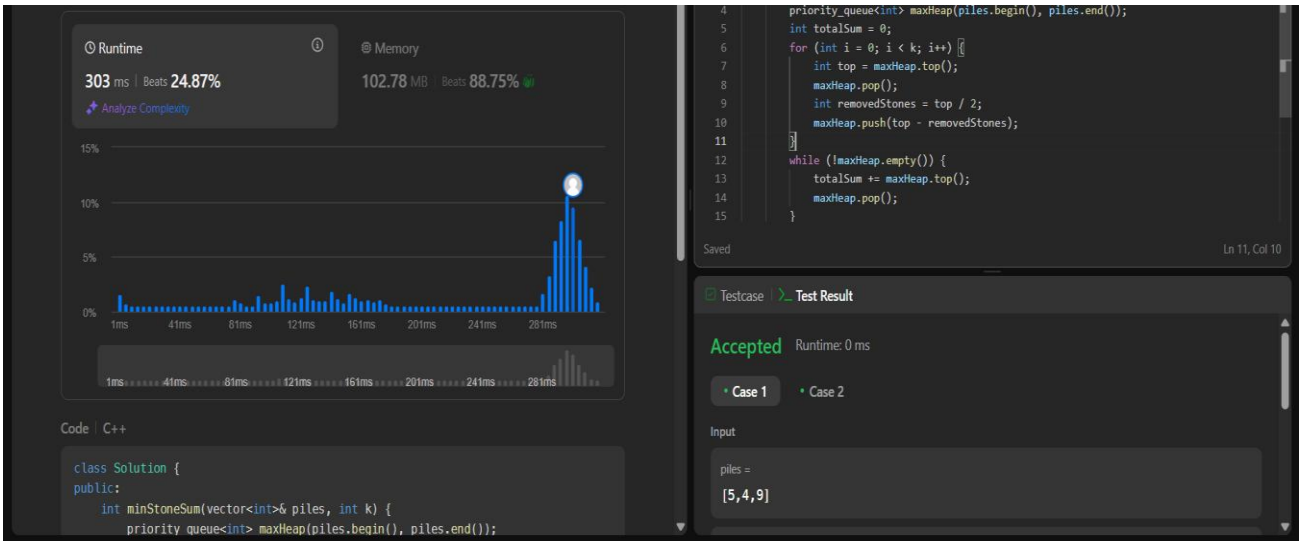


Problem 3: Remove Stones to Minimize the Total (https://leetcode.com/problems/remove-stones-to-minimize-the-total/ **)**

Code:

```
class Solution {

public:

  int minStoneSum(vector<int>& piles, int k) {

    priority_queue<int> maxHeap(piles.begin(), piles.end());

    int totalSum = 0;

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

      int top = maxHeap.top();

      maxHeap.pop();

      int removedStones = top / 2;

      maxHeap.push(top - removedStones);

    }

    while (!maxHeap.empty()) {

      totalSum += maxHeap.top();

      maxHeap.pop();

    }

    return totalSum;

  }

};
```

Screenshot:



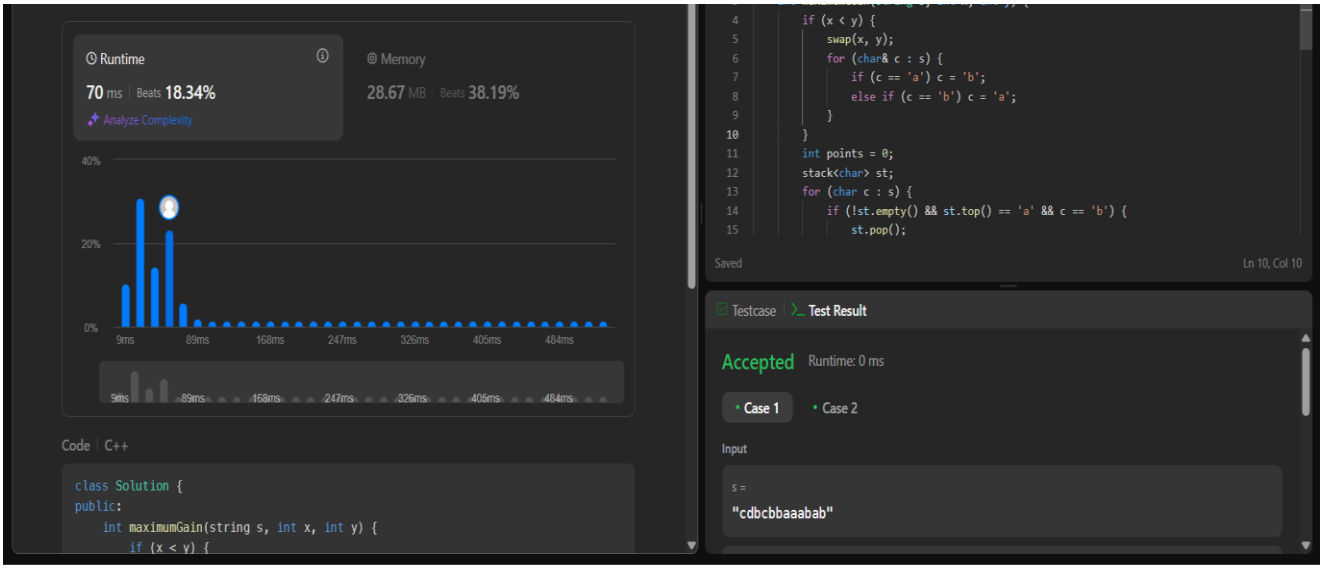
Problem 4: Maximum Score from Removing Substrings
(https://leetcode.com/problems/maximum-score-from-removing-substrings/ **)**

Code:

```
class Solution {
public:
  int maximumGain(string s, int x, int y) {
    if (x < y) {
      swap(x, y);
      for (char& c : s) {
        if (c == 'a') c = 'b';
        else if (c == 'b') c = 'a';
      }
    }
    int points = 0;
    stack<char> st;
    for (char c : s) {
      if (!st.empty() && st.top() == 'a' && c == 'b') {
        st.pop();
        points += x;
      } else {
        st.push(c);
      }
    }
    string remaining;
    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();
            points += y;
        } else {
            st.push(c);
        }
    }
    return points;
  }
};
```

Screenshot:



Problem 5: Minimum Operations to make a Subsequence
(https://leetcode.com/problems/minimum-operations-to-make-a-subsequence/ **)**

Code:

```
class Solution {

public:

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

    int n = target.size(), ans = 0;

    map<int, int> map; //

    for(int i=0;i<n;i++) map[target[i]] = i;

    vector<int> longSubsq;

    for(int num : arr) {

      if(map.find(num) == map.end()) continue;

      auto it = lower_bound(longSubsq.begin(), longSubsq.end(), map[num]);

      if(it == longSubsq.end())

        longSubsq.push_back(map[num]);

      else *it = map[num];

    }

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

  }

};
```

Screenshot: