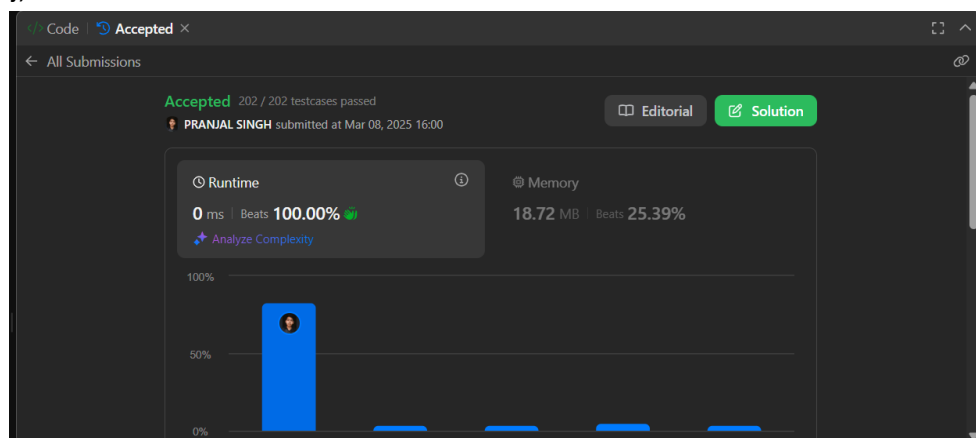


**Name:** Pranjal Singh  
**Sec:** FL\_IOT-601/ A

**UID:** 22BCS13041  
**Sub:** AP Lab -II

### Set Zeros

```
class Solution {
public:
    void setZeroes(vector<vector<int>>& matrix) {
        int n=matrix.size();
        int m=matrix[0].size();
        unordered_set<int> setRows;
        unordered_set<int> setCols;
        for(int i=0;i<n;i++){
            for(int j=0;j<m;j++){
                if(matrix[i][j]==0){
                    setRows.insert(i);
                    setCols.insert(j);
                }
            }
        }
        for(int i=0;i<n;i++){
            for(int j=0;j<m;j++){
                if(setRows.count(i)>0 || setCols.count(j)>0){
                    matrix[i][j]=0;
                }
            }
        }
    }
};
```



## Length of Longest Substring

```
class Solution {
public:
    int lengthOfLongestSubstring(string s) {
        int n = s.length();
        int maxLength = 0;
        unordered_set<char> charSet;
        int left = 0;

        for (int right = 0; right < n; right++) {
            if (charSet.count(s[right]) == 0) {
                charSet.insert(s[right]);
                maxLength = max(maxLength, right - left + 1);
            } else {
                while (charSet.count(s[right])) {
                    charSet.erase(s[left]);
                    left++;
                }
                charSet.insert(s[right]);
            }
        }

        return maxLength;
    }
};
```

The screenshot displays a C++ IDE interface with the following components:

- Problem List:** Shows the problem "Length of Longest Substring" as "Accepted" with 987 / 987 testcases passed. The submission is by user "22bcs13041" on Apr 05, 2025 at 22:03.
- Runtime:** 19 ms, Beats 31.48%.
- Memory:** 14.34 MB, Beats 23.07%.
- Code Editor:** Contains the C++ code for the solution, with line numbers 8 to 24. The code is saved.
- Testcase:** Case 1 is selected, showing the input string "abcabcbb".
- Test Result:** Shows the execution time for Case 1 as 3ms.

## Reverse Linked list

```
class Solution {
```

```
public:
```

```
    ListNode* reverseBetween(ListNode* head, int left, int right) {  
        if (!head || left == right) return head;
```

```
        ListNode dummy(0);  
        dummy.next = head;  
        ListNode* prev = &dummy;
```

```
        for (int i = 0; i < left - 1; ++i) {  
            prev = prev->next;  
        }
```

```
        ListNode* current = prev->next;
```

```
        for (int i = 0; i < right - left; ++i) {  
            ListNode* next_node = current->next;  
            current->next = next_node->next;  
            next_node->next = prev->next;  
            prev->next = next_node;  
        }
```

```
        return dummy.next;
```

```
    }  
}
```

The screenshot displays a code editor interface for a C++ solution. The 'Code' tab is active, showing the following code:

```
9  
10 for (int i = 0; i < left - 1; ++i) {  
11     prev = prev->next;  
12 }  
13  
14 ListNode* current = prev->next;  
15  
16 for (int i = 0; i < right - left; ++i) {  
17     ListNode* next_node = current->next;  
18     current->next = next_node->next;  
19     next_node->next = prev->next;  
20     prev->next = next_node;  
21 }  
22  
23 return dummy.next;  
24 }  
25 ;
```

The 'Testcase' tab is also visible, showing the input: `head = [1,2,3,4,5]`.

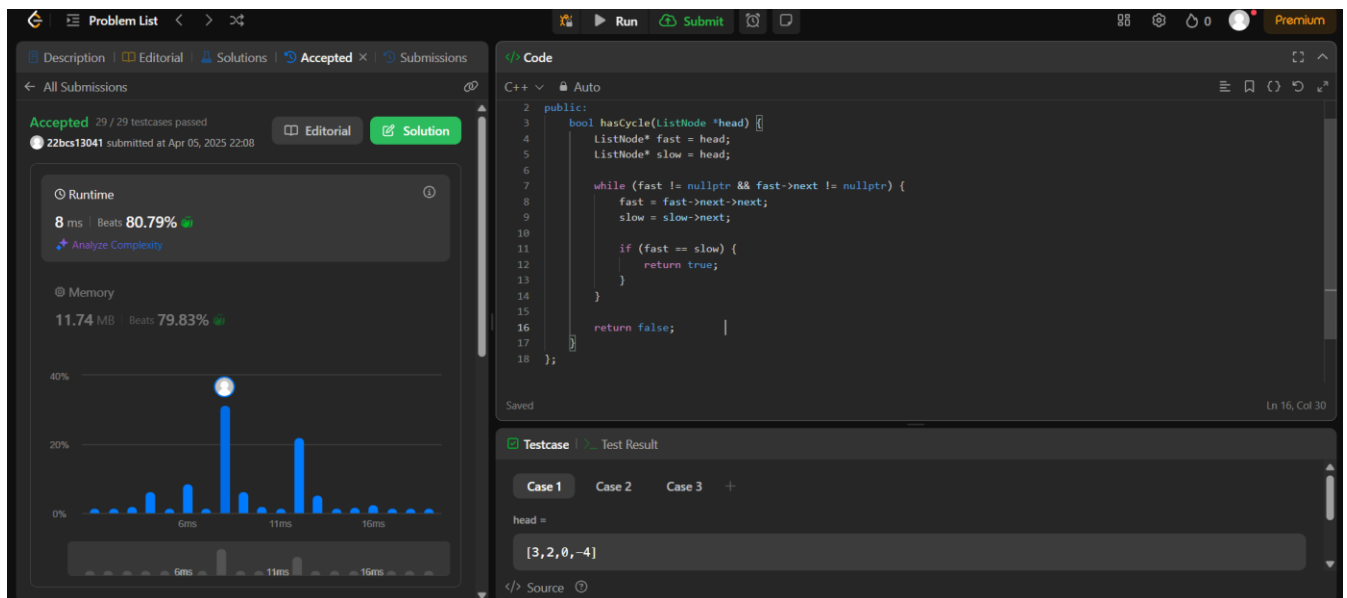
The left sidebar shows submission statistics:

- Accepted 44 / 44 testcases passed
- 22bcs13041 submitted at Apr 05, 2025 22:06
- Runtime: 0 ms, Beats 100.00%
- Memory: 11.09 MB, Beats 96.85%

A bar chart shows the runtime performance across different test cases, with the first case (1ms) being the fastest.

## Linked list has no cycle

```
class Solution {  
  
public:  
  
    bool hasCycle(ListNode *head) {  
  
        ListNode* fast = head;  
  
        ListNode* slow = head;  
  
        while (fast != nullptr && fast->next != nullptr) {  
  
            fast = fast->next->next;  
  
            slow = slow->next;  
  
            if (fast == slow) {  
  
                return true;  
  
            }  
  
        }  
  
        return false;  
  
    }  
  
};
```



## **Skyline problem**

```
class Solution {
public:
    vector<vector<int>> getSkyline(vector<vector<int>>& buildings) {
        vector<vector<int>> ans;
        multiset<int> pq{0};

        vector<pair<int, int>> points;

        for(auto b: buildings){
            points.push_back({b[0], -b[2]});
            points.push_back({b[1], b[2]});
        }

        sort(points.begin(), points.end());

        int ongoingHeight = 0;

        // points.first = x coordinate, points.second = height
        for(int i = 0; i < points.size(); i++){
            int currentPoint = points[i].first;
            int heightAtCurrentPoint = points[i].second;

            if(heightAtCurrentPoint < 0){
                pq.insert(-heightAtCurrentPoint);
            } else {
                pq.erase(pq.find(heightAtCurrentPoint));
            }

            // after inserting/removing heightAt, if there's a change
            auto pqTop = *pq.rbegin();
            if(ongoingHeight != pqTop){
                ongoingHeight = pqTop;
                ans.push_back({currentPoint, ongoingHeight});
            }
        }
    }
}
```

```
    return ans;
}
};
```

The screenshot displays a LeetCode submission page for a problem. The left sidebar shows the submission status: "Accepted 44 / 44 testcases passed" and "22bcs13041 submitted at Apr 05, 2025 22:09". Below this, the "Runtime" section indicates "11 ms" and "Beats 86.80%". The "Memory" section shows "28.80 MB" and "Beats 42.66%". A bar chart visualizes the runtime distribution. The main area shows the C++ code, which uses a priority queue to track the height of buildings and calculate the trapped water. The code is as follows:

```
23     if(heightAtCurrentPoint < 0){
24         pq.insert(-heightAtCurrentPoint);
25     } else {
26         pq.erase(pq.find(heightAtCurrentPoint));
27     }
28
29     // after inserting/removing heightAtI, if there's a change
30     auto pqTop = *pq.rbegin();
31     if(ongoingHeight != pqTop){
32         ongoingHeight = pqTop;
33         ans.push_back({currentPoint, ongoingHeight});
34     }
35 }
36
37 return ans;
38
39 };
```

The right sidebar shows the "Testcase" section with "Case 1" selected. The input for Case 1 is:

```
buildings =
[[2, 9, 10], [3, 7, 15], [5, 12, 12], [15, 20, 10], [19, 24, 8]]
```