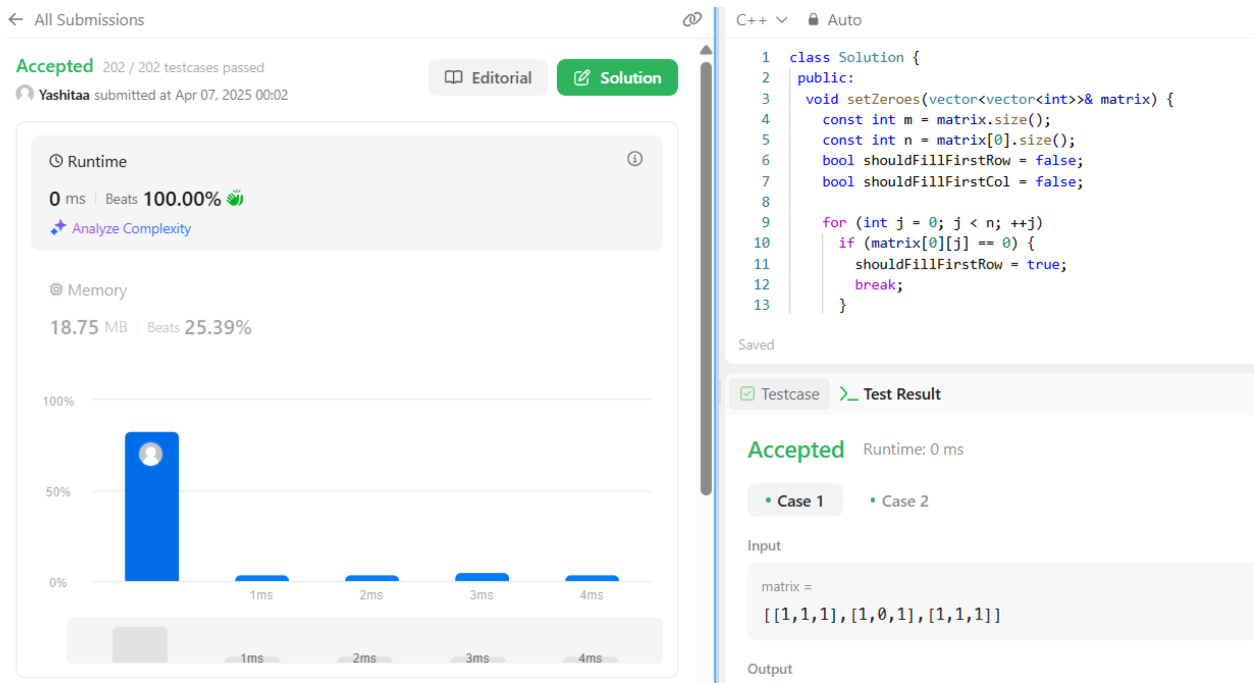
NAME : Yashita
UID : 22BCS15024
SECTION : FL_IoT 601 ‘A’
Ap lab assignment

1. **Set Matrix Zeroes**

```
class Solution {
 public:
  void setZeroes(vector<vector<int>>& matrix) {
    const int m = matrix.size();
    const int n = matrix[0].size();
    bool shouldFillFirstRow = false;
    bool shouldFillFirstCol = false;

    for (int j = 0; j < n; ++j)
      if (matrix[0][j] == 0) {
        shouldFillFirstRow = true;
        break;
      }

    for (int i = 0; i < m; ++i)
      if (matrix[i][0] == 0) {
        shouldFillFirstCol = true;
        break;
      }
    for (int i = 1; i < m; ++i)
```

```
      for (int j = 1; j < n; ++j)
        if (matrix[i][j] == 0) {
          matrix[i][0] = 0;
          matrix[0][j] = 0;
        }
    for (int i = 1; i < m; ++i)
      for (int j = 1; j < n; ++j)
        if (matrix[i][0] == 0 || matrix[0][j] == 0)
          matrix[i][j] = 0;
    if (shouldFillFirstRow)
      for (int j = 0; j < n; ++j)
        matrix[0][j] = 0;
    if (shouldFillFirstCol)
      for (int i = 0; i < m; ++i)
        matrix[i][0] = 0;
  }
};
```



Accepted 202 / 202 testcases passed

Yashitaa submitted at Apr 07, 2025 00:02

Runtime

0 ms | Beats 100.00%

Memory

18.75 MB | Beats 25.39%

```
class Solution {
public:
  void setZeroes(vector<vector<int>>& matrix) {
    const int m = matrix.size();
    const int n = matrix[0].size();
    bool shouldFillFirstRow = false;
    bool shouldFillFirstCol = false;

    for (int j = 0; j < n; ++j)
      if (matrix[0][j] == 0) {
        shouldFillFirstRow = true;
        break;
      }
```

Accepted Runtime: 0 ms

Input

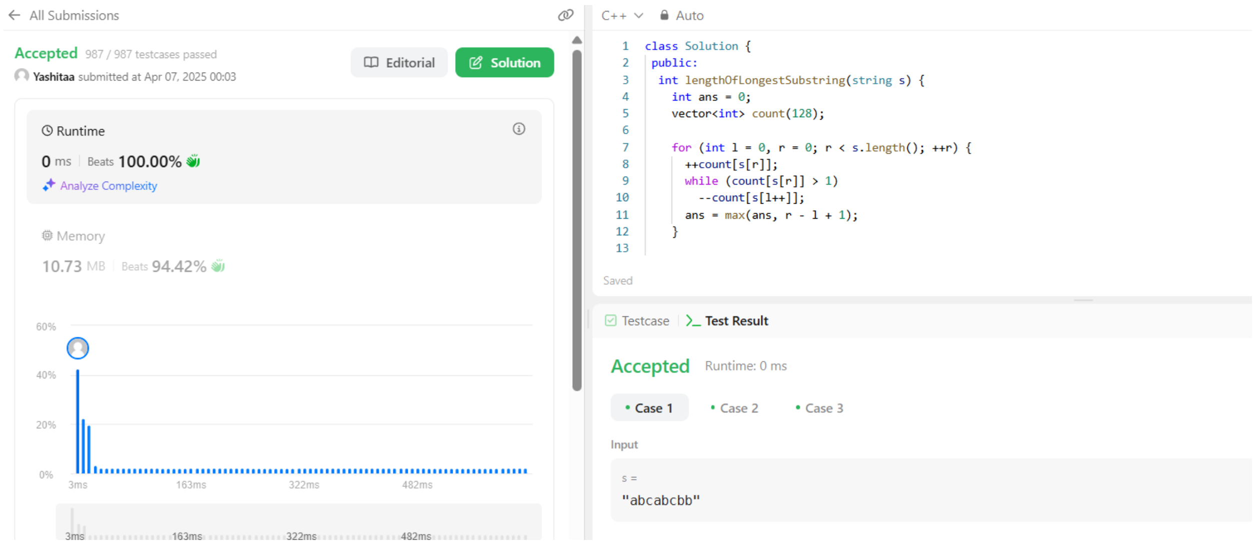
matrix =

```
[[1,1,1],[1,0,1],[1,1,1]]
```

Output

2. Longest Substring Without Repeating Characters

```
class Solution {
 public:
  int lengthOfLongestSubstring(string s) {
    int ans = 0;
    vector<int> count(128);

    for (int l = 0, r = 0; r < s.length(); ++r) {
      ++count[s[r]];
      while (count[s[r]] > 1)
        --count[s[l++]];
      ans = max(ans, r - l + 1);
    }

    return ans;
  }
};
```



All Submissions

Accepted 987 / 987 testcases passed

Yashitaa submitted at Apr 07, 2025 00:03

Runtime

0 ms | Beats 100.00%

Memory

10.73 MB | Beats 94.42%

C++ Auto

```
class Solution {
 public:
  int lengthOfLongestSubstring(string s) {
    int ans = 0;
    vector<int> count(128);

    for (int l = 0, r = 0; r < s.length(); ++r) {
      ++count[s[r]];
      while (count[s[r]] > 1)
        --count[s[l++]];
      ans = max(ans, r - l + 1);
    }

```

Saved

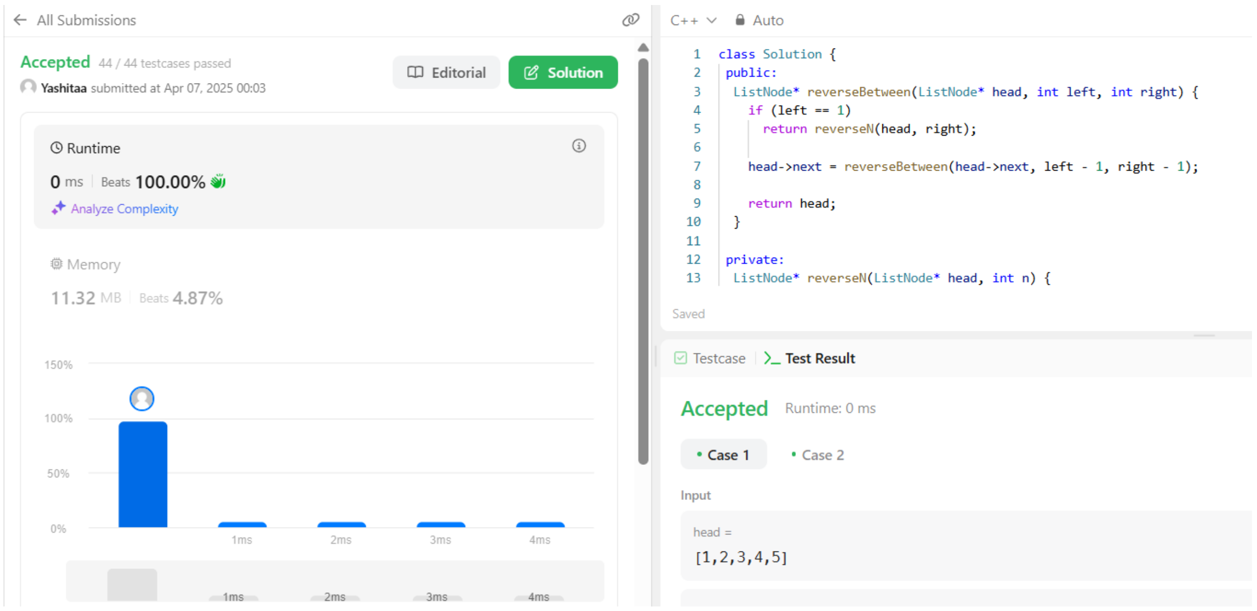
Accepted Runtime: 0 ms

Input

```
s =
"abcabcbb"
```

3. [Reverse Linked List II]

```
class Solution {
 public:
  ListNode* reverseBetween(ListNode* head, int left, int right) {
    if (left == 1)
      return reverseN(head, right);

    head->next = reverseBetween(head->next, left - 1, right - 1);

    return head;
  }

 private:
  ListNode* reverseN(ListNode* head, int n) {
    if (n == 1)
      return head;

    ListNode* newHead = reverseN(head->next, n - 1);
    ListNode* headNext = head->next;
    head->next = headNext->next;
    headNext->next = head;

    return newHead;
  }
};
```



All Submissions

Accepted 44 / 44 testcases passed

Yashitaa submitted at Apr 07, 2025 00:03

Runtime

0 ms | Beats 100.00%

Memory

11.32 MB | Beats 4.87%

C++ Auto

```
class Solution {
 public:
  ListNode* reverseBetween(ListNode* head, int left, int right) {
    if (left == 1)
      return reverseN(head, right);

    head->next = reverseBetween(head->next, left - 1, right - 1);

    return head;
  }

 private:
  ListNode* reverseN(ListNode* head, int n) {
```

Saved

Testcase | Test Result

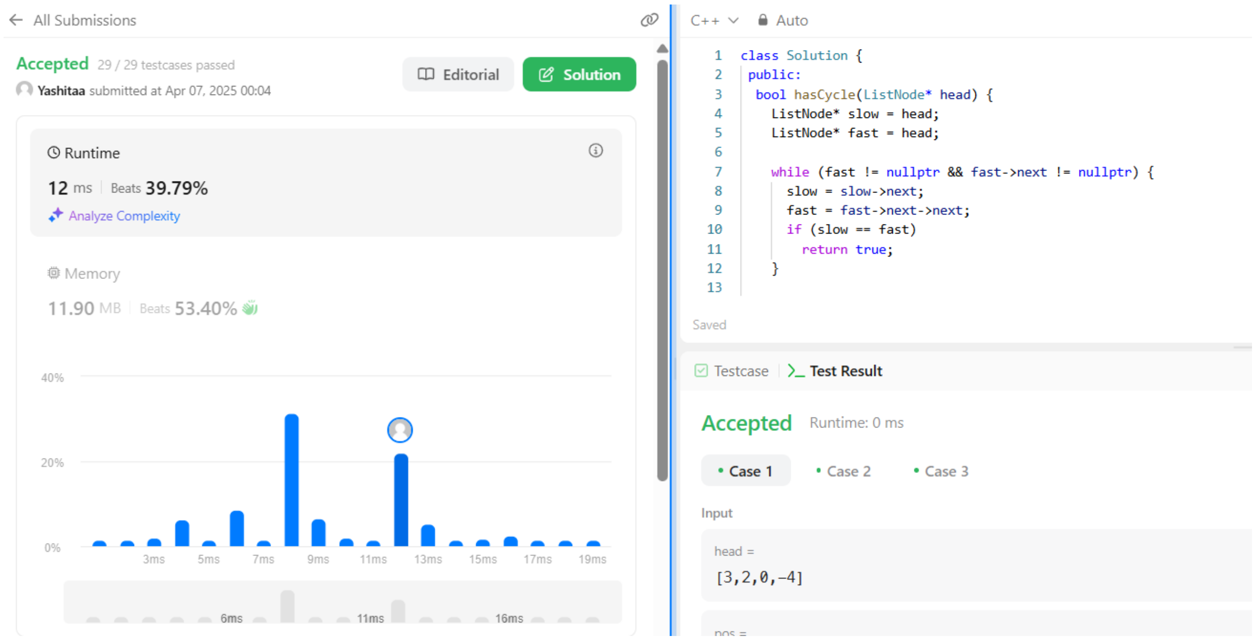
Accepted Runtime: 0 ms

Input

head =
[1,2,3,4,5]

4. **Linked List Cycle**

```
class Solution {
 public:
  bool hasCycle(ListNode* head) {
    ListNode* slow = head;
    ListNode* fast = head;

    while (fast != nullptr && fast->next != nullptr) {
      slow = slow->next;
      fast = fast->next->next;
      if (slow == fast)
        return true;
    }
    return false;
  }
};
```



All Submissions

Accepted 29 / 29 testcases passed

Yashitaa submitted at Apr 07, 2025 00:04

Runtime

12 ms | Beats 39.79%

Memory

11.90 MB | Beats 53.40%

```
class Solution {
 public:
  bool hasCycle(ListNode* head) {
    ListNode* slow = head;
    ListNode* fast = head;

    while (fast != nullptr && fast->next != nullptr) {
      slow = slow->next;
      fast = fast->next->next;
      if (slow == fast)
        return true;
    }
```

Accepted Runtime: 0 ms

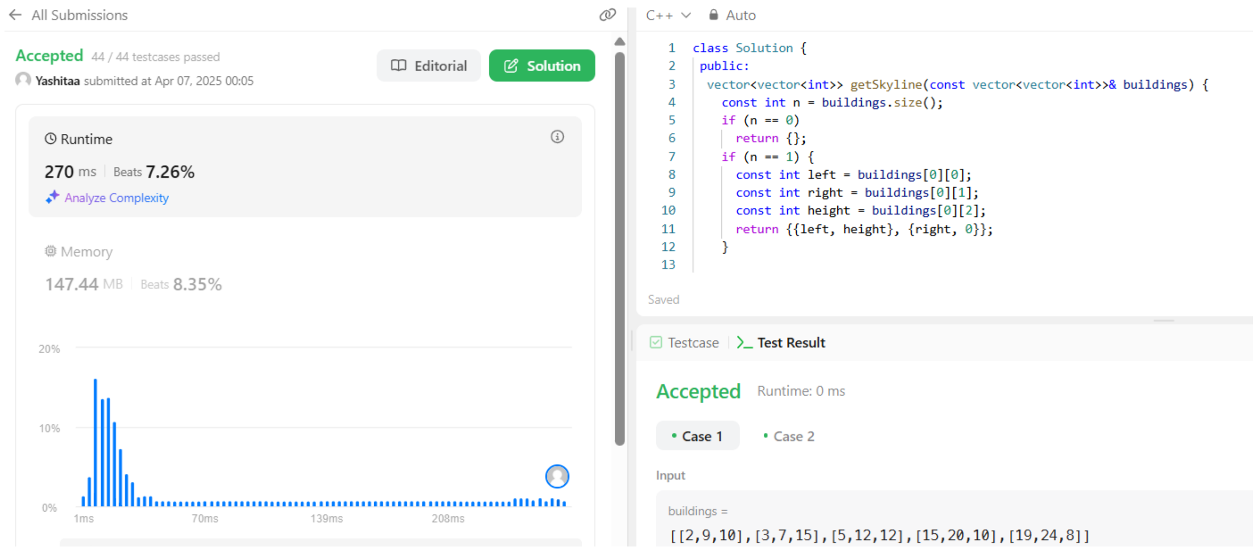
Input

head =
[3,2,0,-4]

5. [The Skyline Problem]

```
class Solution {
 public:
  vector<vector<int>> getSkyline(const vector<vector<int>>& buildings) {
    const int n = buildings.size();
    if (n == 0)
      return {};
    if (n == 1) {
      const int left = buildings[0][0];
      const int right = buildings[0][1];
      const int height = buildings[0][2];
      return {{left, height}, {right, 0}};
    }

    const vector<vector<int>> left =
      getSkyline({buildings.begin(), buildings.begin() + n / 2});
    const vector<vector<int>> right =
      getSkyline({buildings.begin() + n / 2, buildings.end()});
    return merge(left, right);
  }

 private:
  vector<vector<int>> merge(const vector<vector<int>>& left,
                            const vector<vector<int>>& right) {
    vector<vector<int>> ans;
    int i = 0;
    int j = 0;
    int leftY = 0;
    int rightY = 0;

    while (i < left.size() && j < right.size())
      if (left[i][0] < right[j][0]) {
        leftY = left[i][1];
        addPoint(ans, left[i][0], max(left[i++][1], rightY));
      } else {
        rightY = right[j][1];
        addPoint(ans, right[j][0], max(right[j++][1], leftY));
      }

    while (i < left.size())
      addPoint(ans, left[i][0], left[i++][1]);

    while (j < right.size())
      addPoint(ans, right[j][0], right[j++][1]);

    return ans;
  }
```

```
      void addPoint(vector<vector<int>>& ans, int x, int y) {
        if (!ans.empty() && ans.back()[0] == x) {
          ans.back()[1] = y;
          return;
        }
        if (!ans.empty() && ans.back()[1] == y)
          return;
        ans.push_back({x, y});
      }
    };
```



Accepted 44 / 44 testcases passed

Yashitaa submitted at Apr 07, 2025 00:05

Runtime

270 ms | Beats 7.26%

Memory

147.44 MB | Beats 8.35%

```
class Solution {
 public:
  vector<vector<int>> getSkyline(const vector<vector<int>>& buildings) {
    const int n = buildings.size();
    if (n == 0)
      return {};
    if (n == 1) {
      const int left = buildings[0][0];
      const int right = buildings[0][1];
      const int height = buildings[0][2];
      return {{left, height}, {right, 0}};
    }

```

Accepted Runtime: 0 ms

• Case 1 • Case 2

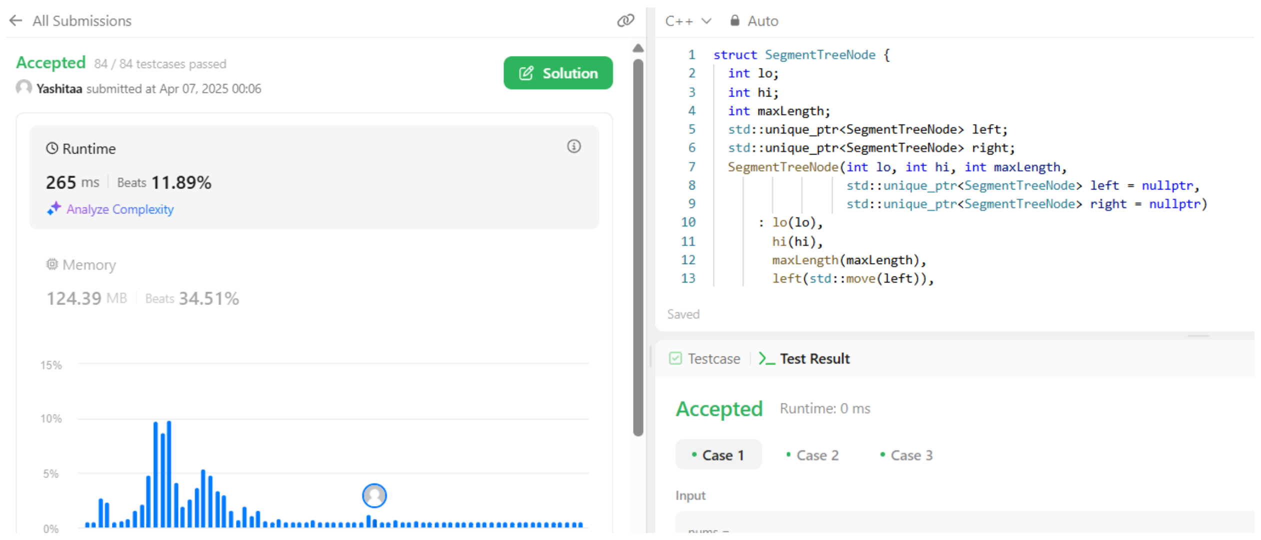
Input

buildings =

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

6. Longest Increasing Subsequence II

```
struct SegmentTreeNode {
  int lo;
  int hi;
  int maxLength;
  std::unique_ptr<SegmentTreeNode> left;
  std::unique_ptr<SegmentTreeNode> right;
  SegmentTreeNode(int lo, int hi, int maxLength,
                  std::unique_ptr<SegmentTreeNode> left = nullptr,
                  std::unique_ptr<SegmentTreeNode> right = nullptr)
      : lo(lo),
        hi(hi),
      maxLength(maxLength),
      left(std::move(left)),
      right(std::move(right)) {}
};

class SegmentTree {
 public:
  explicit SegmentTree() : root(make_unique<SegmentTreeNode>(0, 1e5 + 1, 0)) {}

  void updateRange(int i, int j, int maxLength) {
    update(root, i, j, maxLength);
  }
  int queryRange(int i, int j) {
    return query(root, i, j);
  }

 private:
  std::unique_ptr<SegmentTreeNode> root;

  void update(std::unique_ptr<SegmentTreeNode>& root, int i, int j,
              int maxLength) {
    if (root->lo == i && root->hi == j) {
      root->maxLength = maxLength;
      root->left = nullptr;
      root->right = nullptr;
      return;
    }
    const int mid = root->lo + (root->hi - root->lo) / 2;
    if (root->left == nullptr) {
      root->left = make_unique<SegmentTreeNode>(root->lo, mid, root->maxLength);
      root->right =
          make_unique<SegmentTreeNode>(mid + 1, root->hi, root->maxLength);
    }
    if (j <= mid)
      update(root->left, i, j, maxLength);
    else if (i > mid)
      update(root->right, i, j, maxLength);
    else {
      update(root->left, i, mid, maxLength);
      update(root->right, mid + 1, j, maxLength);
    }
    root->maxLength = merge(root->left->maxLength, root->right->maxLength);
  }

  int query(std::unique_ptr<SegmentTreeNode>& root, int i, int j) {
    if (root->left == nullptr)
      return root->maxLength;
    if (root->lo == i && root->hi == j)
      return root->maxLength;
    const int mid = root->lo + (root->hi - root->lo) / 2;
    if (j <= mid)
      return query(root->left, i, j);
    if (i > mid)
      return query(root->right, i, j);
    return merge(query(root->left, i, mid), query(root->right, mid + 1, j));
  }

  int merge(int left, int right) const {
    return max(left, right);
  };
};

class Solution {
 public:
  int lengthOfLIS(vector<int>& nums, int k) {
    int ans = 1;
    SegmentTree tree;

    for (const int num : nums) {
      const int left = max(1, num - k);
      const int right = num - 1;
      const int maxLength = tree.queryRange(left, right) + 1;
      ans = max(ans, maxLength);
      tree.updateRange(num, num, maxLength);
    }
  return ans;
 }
};
```



All Submissions

Accepted 84 / 84 testcases passed
Yashitaa submitted at Apr 07, 2025 00:06

Runtime
265 ms | Beats 11.89%

Memory
124.39 MB | Beats 34.51%

```
struct SegmentTreeNode {
  int lo;
  int hi;
  int maxLength;
  std::unique_ptr<SegmentTreeNode> left;
  std::unique_ptr<SegmentTreeNode> right;
  SegmentTreeNode(int lo, int hi, int maxLength,
                  std::unique_ptr<SegmentTreeNode> left = nullptr,
                  std::unique_ptr<SegmentTreeNode> right = nullptr)
      : lo(lo),
        hi(hi),
        maxLength(maxLength),
        left(std::move(left)),
```

Saved

Testcase | Test Result

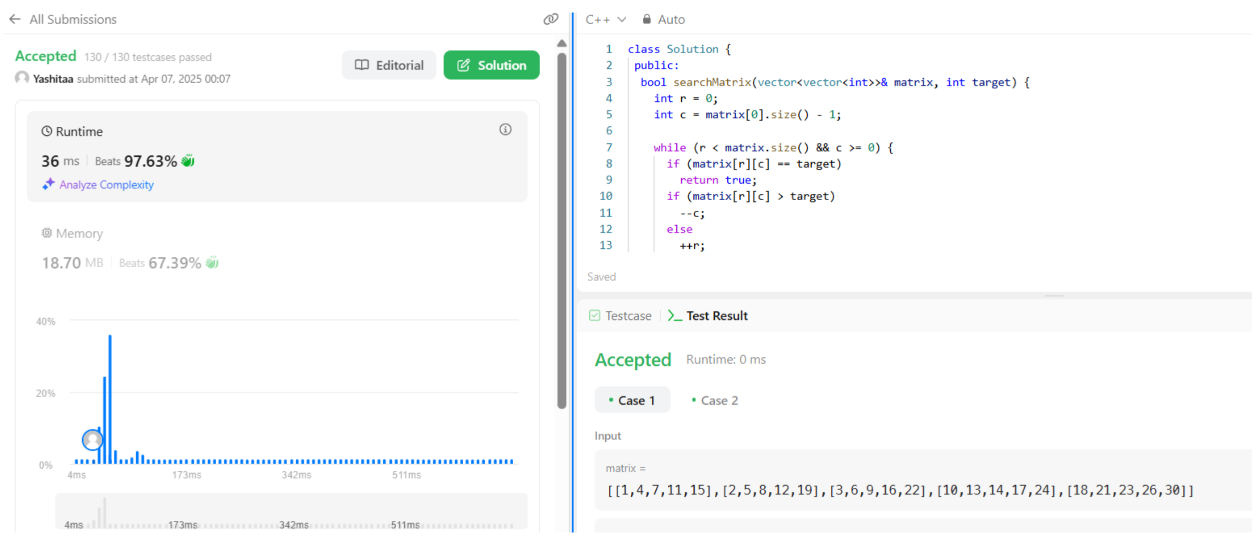
Accepted Runtime: 0 ms

Case 1 Case 2 Case 3

Input

7. **Search a 2D Matrix II**

```
class Solution {
 public:
  bool searchMatrix(vector<vector<int>>& matrix, int target) {
    int r = 0;
    int c = matrix[0].size() - 1;

    while (r < matrix.size() && c >= 0) {
      if (matrix[r][c] == target)
        return true;
      if (matrix[r][c] > target)
        --c;
      else
        ++r;
    }

    return false;
  }
};
```



Accepted 130 / 130 testcases passed
Yashitaa submitted at Apr 07, 2025 00:07

Runtime
36 ms | Beats 97.63%

Memory
18.70 MB | Beats 67.39%

```
class Solution {
 public:
  bool searchMatrix(vector<vector<int>>& matrix, int target) {
    int r = 0;
    int c = matrix[0].size() - 1;

    while (r < matrix.size() && c >= 0) {
      if (matrix[r][c] == target)
        return true;
      if (matrix[r][c] > target)
        --c;
      else
        ++r;
```

Saved

Testcase | Test Result

Accepted Runtime: 0 ms

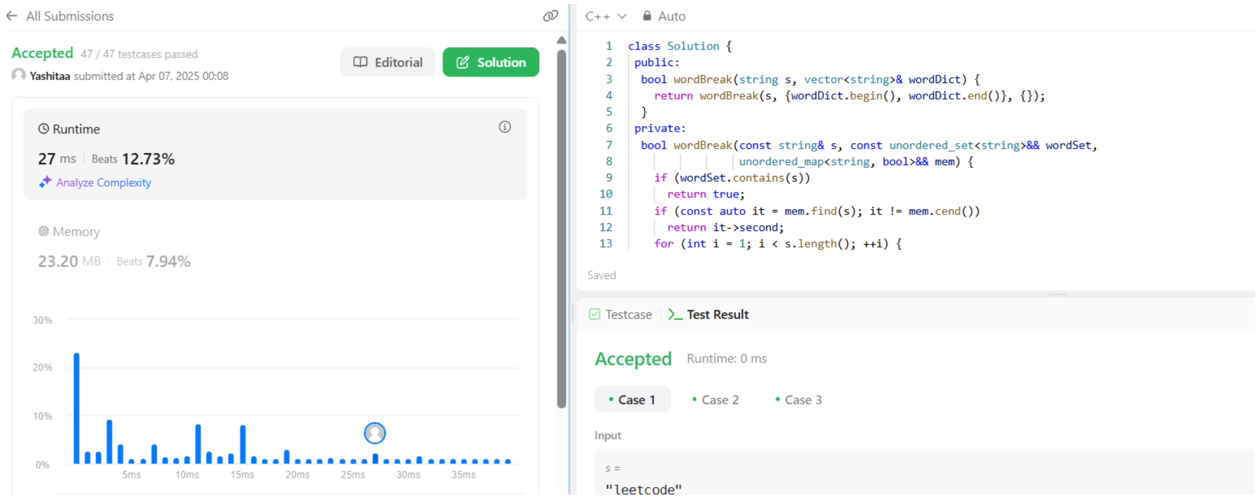
Case 1 Case 2

Input

matrix =
[[1,4,7,11,15],[2,5,8,12,19],[3,6,9,16,22],[10,13,14,17,24],[18,21,23,26,30]]

8. Word Break

```
class Solution {
 public:
  bool wordBreak(string s, vector<string>& wordDict) {
    return wordBreak(s, {wordDict.begin(), wordDict.end()}, {});
  }
 private:
  bool wordBreak(const string& s, const unordered_set<string>&& wordSet,
                 unordered_map<string, bool>&& mem) {
    if (wordSet.contains(s))
      return true;
    if (const auto it = mem.find(s); it != mem.cend())
      return it->second;
    for (int i = 1; i < s.length(); ++i) {
      const string& prefix = s.substr(0, i);
      const string& suffix = s.substr(i);
      if (wordSet.contains(prefix) &&
        wordBreak(suffix, std::move(wordSet), std::move(mem)))
      return mem[s] = true;
    }

    return mem[s] = false;
  }
};
```



All Submissions

Accepted 47 / 47 testcases passed
Yashitaa submitted at Apr 07, 2025 00:08

Editorial Solution

Runtime
27 ms | Beats 12.73%
Analyze Complexity

Memory
23.20 MB | Beats 7.94%

C++ Auto

```
class Solution {
 public:
  bool wordBreak(string s, vector<string>& wordDict) {
    return wordBreak(s, {wordDict.begin(), wordDict.end()}, {});
  }
 private:
  bool wordBreak(const string& s, const unordered_set<string>&& wordSet,
                 unordered_map<string, bool>&& mem) {
    if (wordSet.contains(s))
      return true;
    if (const auto it = mem.find(s); it != mem.cend())
      return it->second;
    for (int i = 1; i < s.length(); ++i) {
```

Saved

Testcase | Test Result

Accepted Runtime: 0 ms

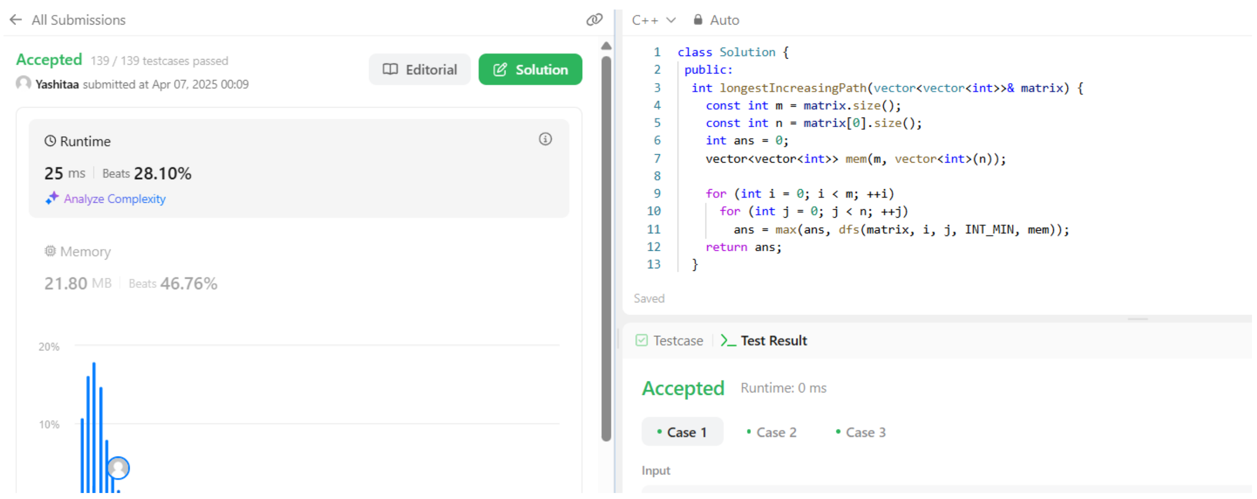
Case 1 Case 2 Case 3

Input

s =
"leetcode"

9. Longest Increasing Path in a Matrix

```
class Solution {
 public:
  int longestIncreasingPath(vector<vector<int>>& matrix) {
    const int m = matrix.size();
    const int n = matrix[0].size();
    int ans = 0;
    vector<vector<int>> mem(m, vector<int>(n));

    for (int i = 0; i < m; ++i)
      for (int j = 0; j < n; ++j)
        ans = max(ans, dfs(matrix, i, j, INT_MIN, mem));
    return ans;
  }
 private:
  int dfs(const vector<vector<int>>& matrix, int i, int j, int prev,
          vector<vector<int>>& mem) {
    if (i < 0 || i == matrix.size() || j < 0 || j == matrix[0].size())
      return 0;
    if (matrix[i][j] <= prev)
      return 0;
    int& ans = mem[i][j];
    if (ans > 0)
      return ans;

    const int curr = matrix[i][j];
    return ans = 1 + max({dfs(matrix, i + 1, j, curr, mem),
                          dfs(matrix, i - 1, j, curr, mem),
                          dfs(matrix, i, j + 1, curr, mem),
                          dfs(matrix, i, j - 1, curr, mem)});
  }
};
```



All Submissions

Accepted 139 / 139 testcases passed

Yashitaa submitted at Apr 07, 2025 00:09

Runtime

25 ms | Beats 28.10%

Memory

21.80 MB | Beats 46.76%

```
class Solution {
 public:
  int longestIncreasingPath(vector<vector<int>>& matrix) {
    const int m = matrix.size();
    const int n = matrix[0].size();
    int ans = 0;
    vector<vector<int>> mem(m, vector<int>(n));

    for (int i = 0; i < m; ++i)
      for (int j = 0; j < n; ++j)
        ans = max(ans, dfs(matrix, i, j, INT_MIN, mem));
    return ans;
  }
```

Saved

Testcase | Test Result

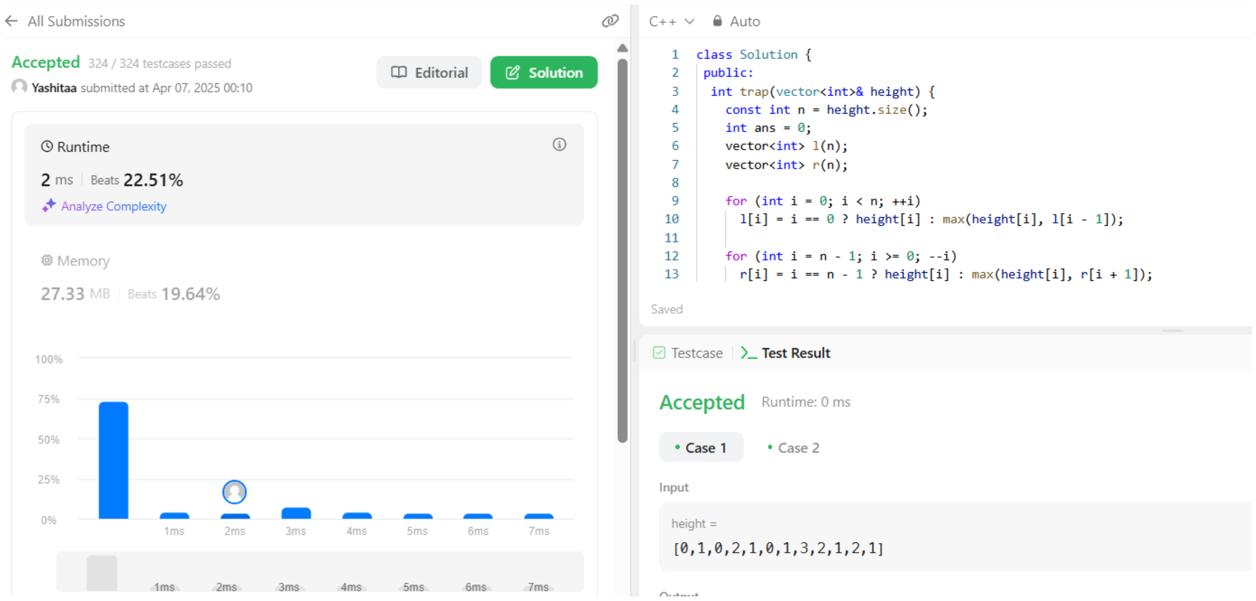
Accepted Runtime: 0 ms

Case 1 Case 2 Case 3

Input

10. **Trapping Rain Water**

```
class Solution {
 public:
  int trap(vector<int>& height) {
    const int n = height.size();
    int ans = 0;
    vector<int> l(n);
    vector<int> r(n);

    for (int i = 0; i < n; ++i)
      l[i] = i == 0 ? height[i] : max(height[i], l[i - 1]);

    for (int i = n - 1; i >= 0; --i)
      r[i] = i == n - 1 ? height[i] : max(height[i], r[i + 1]);

    for (int i = 0; i < n; ++i)
      ans += min(l[i], r[i]) - height[i];

    return ans;
  }
};
```



All Submissions

Accepted 324 / 324 testcases passed

Yashitaa submitted at Apr 07, 2025 00:10

Runtime

2 ms | Beats 22.51%

Memory

27.33 MB | Beats 19.64%

C++ Auto

```cpp
class Solution {
 public:
  int trap(vector<int>& height) {
    const int n = height.size();
    int ans = 0;
    vector<int> l(n);
    vector<int> r(n);

    for (int i = 0; i < n; ++i)
      l[i] = i == 0 ? height[i] : max(height[i], l[i - 1]);

    for (int i = n - 1; i >= 0; --i)
      r[i] = i == n - 1 ? height[i] : max(height[i], r[i + 1]);
```

Saved

Testcase | Test Result

Accepted Runtime: 0 ms

Case 1 Case 2

Input

height =

[0,1,0,2,1,0,1,3,2,1,2,1]