Name : Anish Patial

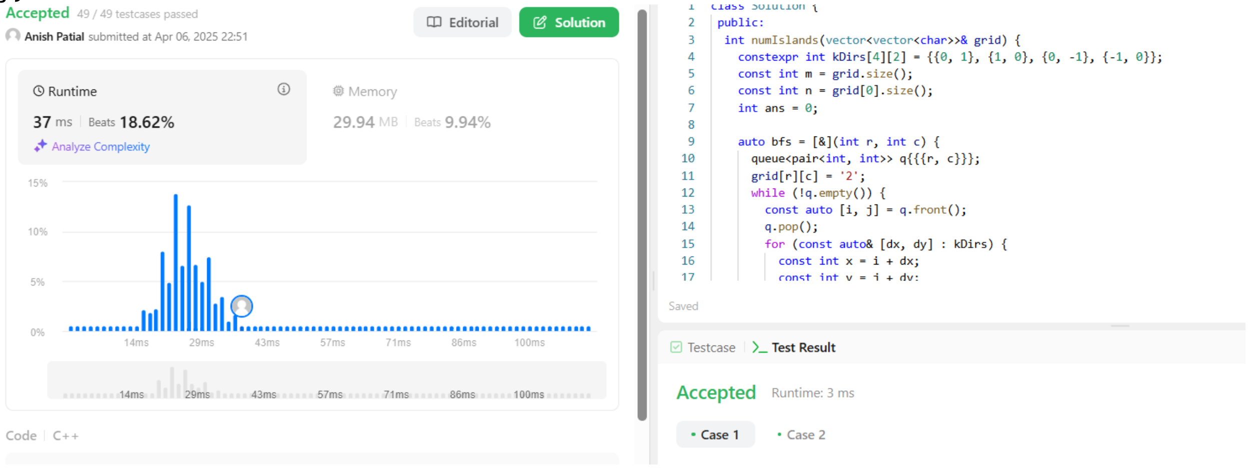
UID : 22BCS15029

SEC : Fl_Iot 601 'A'

Ap experiment 9

1. Number of Islands

```
class Solution {
 public:
  int numIslands(vector<vector<char>>& grid) {
    constexpr int kDirs[4][2] = {{0, 1}, {1, 0}, {0, -1}, {-1, 0}};
    const int m = grid.size();
    const int n = grid[0].size();
    int ans = 0;

    auto bfs = [&](int r, int c) {
      queue<pair<int, int>> q{{{r, c}}};
      grid[r][c] = '2';
      while (!q.empty()) {
        const auto [i, j] = q.front();
        q.pop();
        for (const auto& [dx, dy] : kDirs) {
          const int x = i + dx;
          const int y = j + dy;
          if (x < 0 || x == m || y < 0 || y == n)
            continue;
          if (grid[x][y] != '1')
            continue;
          q.emplace(x, y);
          grid[x][y] = '2';
        }
      }
    };

    for (int i = 0; i < m; ++i)
      for (int j = 0; j < n; ++j)
        if (grid[i][j] == '1') {
          bfs(i, j);
          ++ans;
        }

    return ans;
  }
};
```



Accepted 49 / 49 testcases passed

Anish Patial submitted at Apr 06, 2025 22:51

Runtime 37 ms Beats 18.62%

Memory 29.94 MB Beats 9.94%

Code | C++

```
class Solution {
 public:
  int numIslands(vector<vector<char>>& grid) {
    constexpr int kDirs[4][2] = {{0, 1}, {1, 0}, {0, -1}, {-1, 0}};
    const int m = grid.size();
    const int n = grid[0].size();
    int ans = 0;

    auto bfs = [&](int r, int c) {
      queue<pair<int, int>> q{{{r, c}}};
      grid[r][c] = '2';
      while (!q.empty()) {
        const auto [i, j] = q.front();
        q.pop();
        for (const auto& [dx, dy] : kDirs) {
          const int x = i + dx;
          const int y = j + dy;
```

Saved

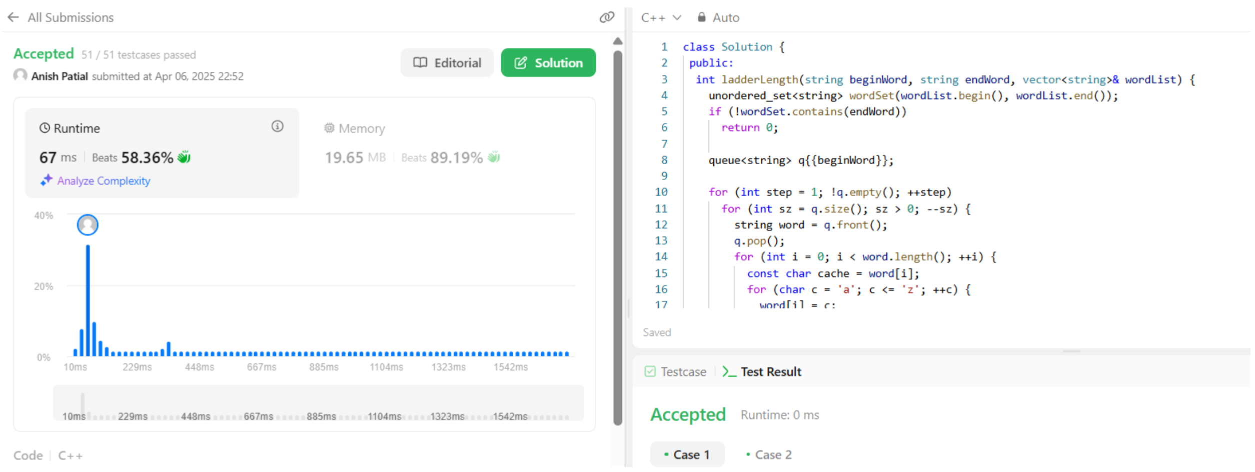
Testcase | Test Result

Accepted Runtime: 3 ms

Case 1 Case 2

2. Word Ladder

```
class Solution {
 public:
  int ladderLength(string beginWord, string endWord, vector<string>& wordList) {
    unordered_set<string> wordSet(wordList.begin(), wordList.end());
    if (!wordSet.contains(endWord))
      return 0;

    queue<string> q{{beginWord}};

    for (int step = 1; !q.empty(); ++step)
      for (int sz = q.size(); sz > 0; --sz) {
        string word = q.front();
        q.pop();
        for (int i = 0; i < word.length(); ++i) {
          const char cache = word[i];
          for (char c = 'a'; c <= 'z'; ++c) {
            word[i] = c;
            if (word == endWord)
              return step + 1;
            if (wordSet.contains(word)) {
              q.push(word);
              wordSet.erase(word);
            }
          }
          word[i] = cache;
        }
      }

    return 0;
  }
};
```


All Submissions

Accepted 51 / 51 testcases passed

Anish Patial submitted at Apr 06, 2025 22:52

Runtime

67 ms | Beats 58.36%

Memory

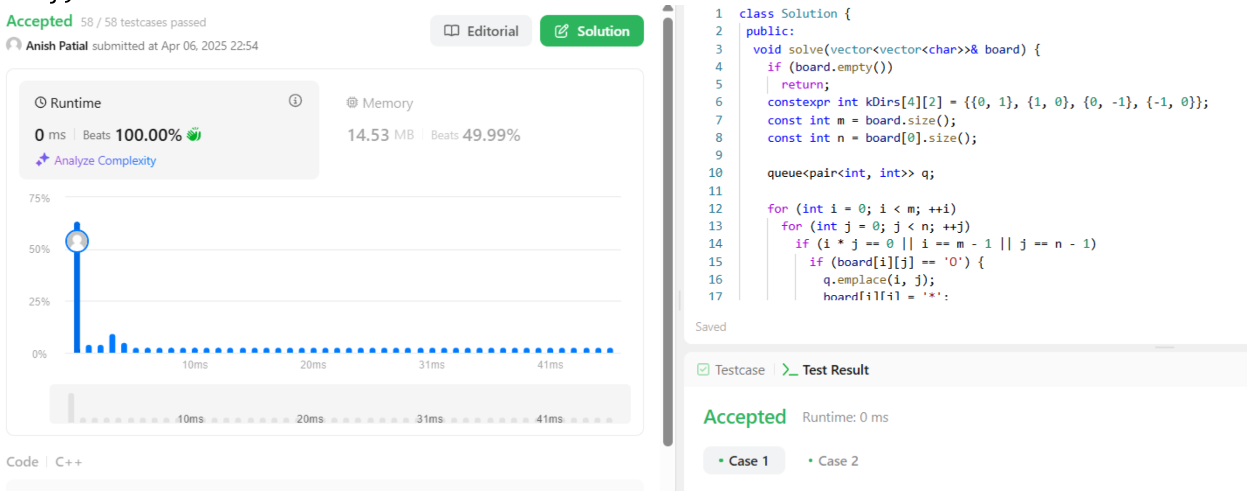
19.65 MB | Beats 89.19%

```cpp
class Solution {
 public:
  int ladderLength(string beginWord, string endWord, vector<string>& wordList) {
    unordered_set<string> wordSet(wordList.begin(), wordList.end());
    if (!wordSet.contains(endWord))
      return 0;

    queue<string> q{{beginWord}};

    for (int step = 1; !q.empty(); ++step)
      for (int sz = q.size(); sz > 0; --sz) {
        string word = q.front();
        q.pop();
        for (int i = 0; i < word.length(); ++i) {
          const char cache = word[i];
          for (char c = 'a'; c <= 'z'; ++c) {
            word[i] = c;
```

Accepted Runtime: 0 ms

3. **Surrounded Regions**

```cpp
class Solution {
 public:
  void solve(vector<vector<char>>& board) {
    if (board.empty())
      return;
    constexpr int kDirs[4][2] = {{0, 1}, {1, 0}, {0, -1}, {-1, 0}};
    const int m = board.size();
    const int n = board[0].size();

    queue<pair<int, int>> q;

    for (int i = 0; i < m; ++i)
      for (int j = 0; j < n; ++j)
        if (i * j == 0 || i == m - 1 || j == n - 1)
          if (board[i][j] == 'O') {
            q.emplace(i, j);
            board[i][j] = '*';
          }
    while (!q.empty()) {
      const auto [i, j] = q.front();
      q.pop();
      for (const auto& [dx, dy] : kDirs) {
        const int x = i + dx;
        const int y = j + dy;
        if (x < 0 || x == m || y < 0 || y == n)
          continue;
        if (board[x][y] != 'O')
          continue;
        q.emplace(x, y);
        board[x][y] = '*';
      }
    }

    for (vector<char>& row : board)
      for (char& c : row)
        if (c == '*')
          c = 'O';
        else if (c == 'O')
          c = 'X';
  }
};
```


Accepted 58 / 58 testcases passed

Anish Patial submitted at Apr 06, 2025 22:54

Editorial Solution

Runtime 0 ms Beats 100.00%

Analyze Complexity

Memory 14.53 MB Beats 49.99%

Code | C++

```cpp
class Solution {
 public:
  void solve(vector<vector<char>>& board) {
    if (board.empty())
      return;
    constexpr int kDirs[4][2] = {{0, 1}, {1, 0}, {0, -1}, {-1, 0}};
    const int m = board.size();
    const int n = board[0].size();

    queue<pair<int, int>> q;

    for (int i = 0; i < m; ++i)
      for (int j = 0; j < n; ++j)
        if (i * j == 0 || i == m - 1 || j == n - 1)
          if (board[i][j] == 'O') {
            q.emplace(i, j);
            board[i][j] = '*';
```

Saved

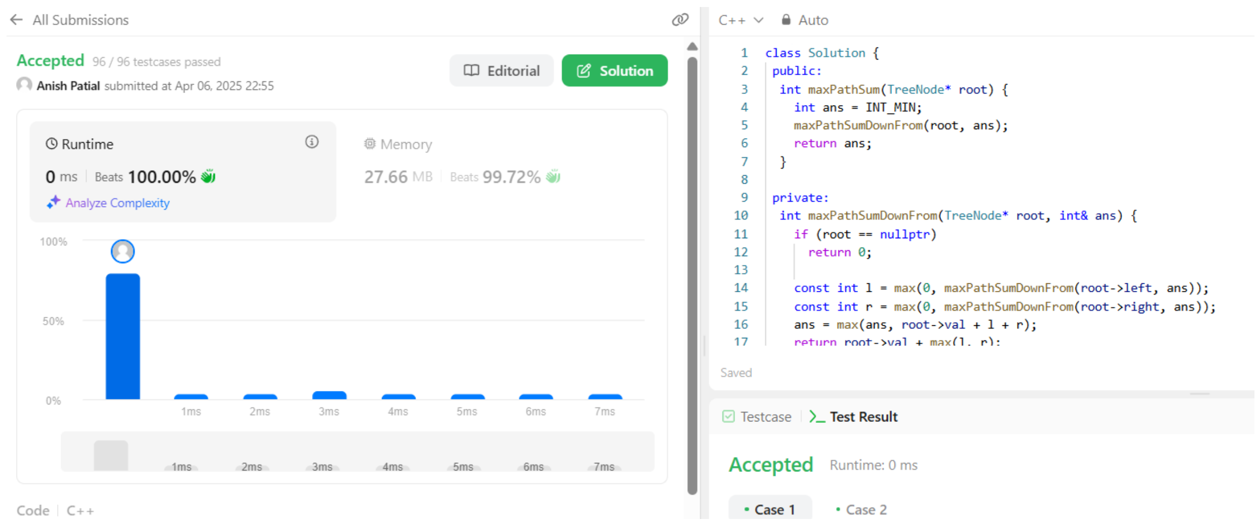
Testcase | Test Result

Accepted Runtime: 0 ms

Case 1 Case 2

4. **Binary Tree Maximum Path Sum**

```cpp
class Solution {
 public:
  int maxPathSum(TreeNode* root) {
    int ans = INT_MIN;
    maxPathSumDownFrom(root, ans);
    return ans;
  }

 private:
  int maxPathSumDownFrom(TreeNode* root, int& ans) {
    if (root == nullptr)
      return 0;

    const int l = max(0, maxPathSumDownFrom(root->left, ans));
    const int r = max(0, maxPathSumDownFrom(root->right, ans));
    ans = max(ans, root->val + l + r);
    return root->val + max(l, r);
  }
};
```



Accepted 96 / 96 testcases passed

Anish Patial submitted at Apr 06, 2025 22:55

Runtime 0 ms | Beats 100.00%

Memory 27.66 MB | Beats 99.72%

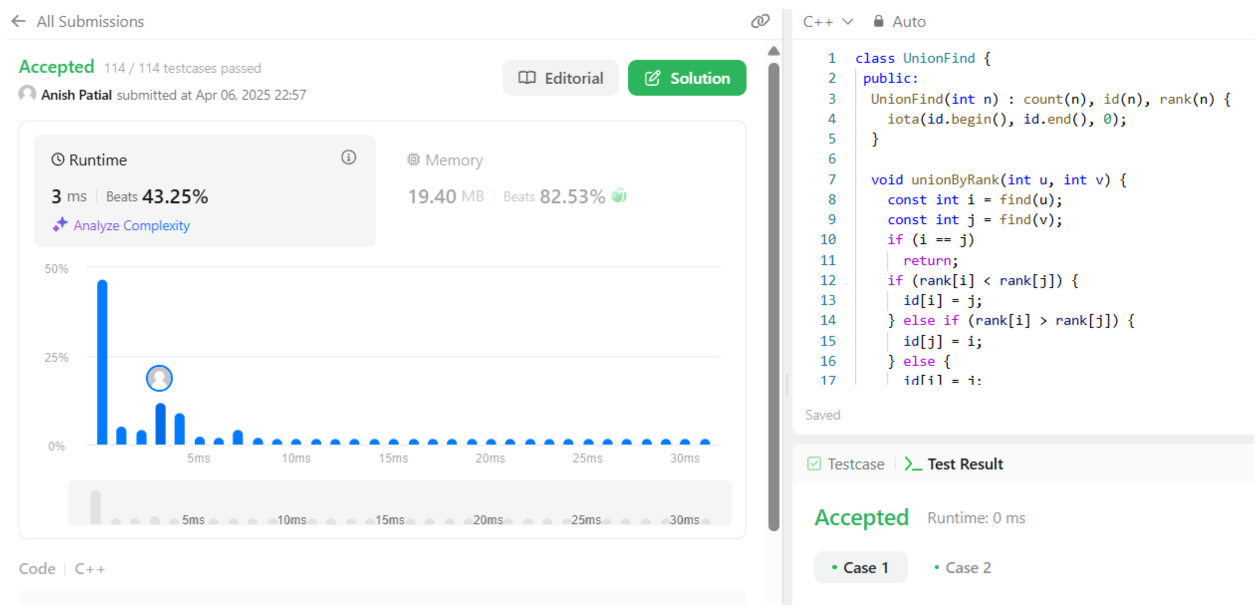
```
class Solution {
public:
  int maxPathSum(TreeNode* root) {
    int ans = INT_MIN;
    maxPathSumDownFrom(root, ans);
    return ans;
  }

public:
private:
  int maxPathSumDownFrom(TreeNode* root, int& ans) {
    if (root == nullptr)
      return 0;

    const int l = max(0, maxPathSumDownFrom(root->left, ans));
    const int r = max(0, maxPathSumDownFrom(root->right, ans));
    ans = max(ans, root->val + l + r);
    return root->val + max(l, r);
```

Accepted Runtime: 0 ms

5. **Number of Provinces**

```
class UnionFind {
 public:
  UnionFind(int n) : count(n), id(n), rank(n) {
    iota(id.begin(), id.end(), 0);
  }

  void unionByRank(int u, int v) {
    const int i = find(u);
    const int j = find(v);
    if (i == j)
      return;
    if (rank[i] < rank[j]) {
      id[i] = j;
    } else if (rank[i] > rank[j]) {
      id[j] = i;
    } else {
      id[i] = j;
      ++rank[j];
    }
    --count;
  }

  int getCount() const {
    return count;
  }

 private:
  int count;
  vector<int> id;
  vector<int> rank;

  int find(int u) {
    return id[u] == u ? u : id[u] = find(id[u]);
  }
};
```

```
class Solution {
 public:
  int findCircleNum(vector<vector<int>>& isConnected) {
    const int n = isConnected.size();
    UnionFind uf(n);

    for (int i = 0; i < n; ++i)
      for (int j = i; j < n; ++j)
        if (isConnected[i][j] == 1)
          uf.unionByRank(i, j);

    return uf.getCount();
  }
};
```



All Submissions

Accepted 114 / 114 testcases passed

Anish Patial submitted at Apr 06, 2025 22:57

Runtime
3 ms | Beats 43.25%

Memory
19.40 MB | Beats 82.53%

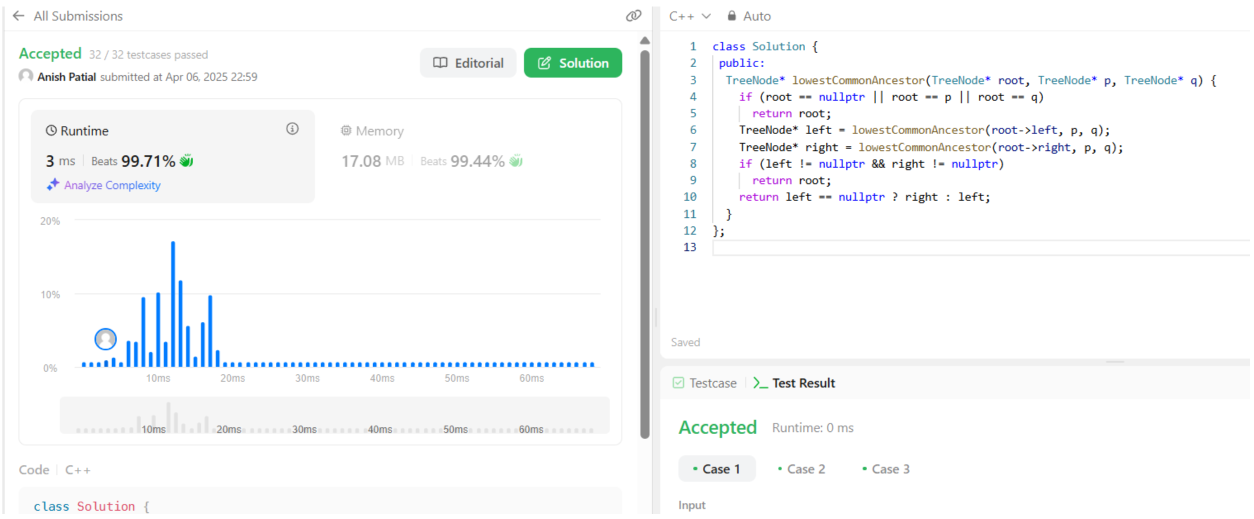
```
class UnionFind {
 public:
  UnionFind(int n) : count(n), id(n), rank(n) {
    iota(id.begin(), id.end(), 0);
  }

  void unionByRank(int u, int v) {
    const int i = find(u);
    const int j = find(v);
    if (i == j)
      return;
    if (rank[i] < rank[j]) {
      id[i] = j;
    } else if (rank[i] > rank[j]) {
      id[j] = i;
    } else {
      id[i] = i;
```

Saved

Accepted Runtime: 0 ms

6. **Lowest Common Ancestor of a Binary Tree**

```
class Solution {
 public:
  TreeNode* lowestCommonAncestor(TreeNode* root, TreeNode* p, TreeNode* q) {
    if (root == nullptr || root == p || root == q)
      return root;
    TreeNode* left = lowestCommonAncestor(root->left, p, q);
    TreeNode* right = lowestCommonAncestor(root->right, p, q);
  if (left != nullptr && right != nullptr)
    return root;
  return left == nullptr ? right : left;
 }
};
```



All Submissions

Accepted 32 / 32 testcases passed
Anish Patial submitted at Apr 06, 2025 22:59

Runtime
3 ms | Beats 99.71%
Analyze Complexity

Memory
17.08 MB | Beats 99.44%

Code | C++

```
class Solution {
```

C++ Auto

```
class Solution {
 public:
  TreeNode* lowestCommonAncestor(TreeNode* root, TreeNode* p, TreeNode* q) {
    if (root == nullptr || root == p || root == q)
      return root;
    TreeNode* left = lowestCommonAncestor(root->left, p, q);
    TreeNode* right = lowestCommonAncestor(root->right, p, q);
    if (left != nullptr && right != nullptr)
      return root;
    return left == nullptr ? right : left;
  }
};
```

Saved

Testcase | Test Result

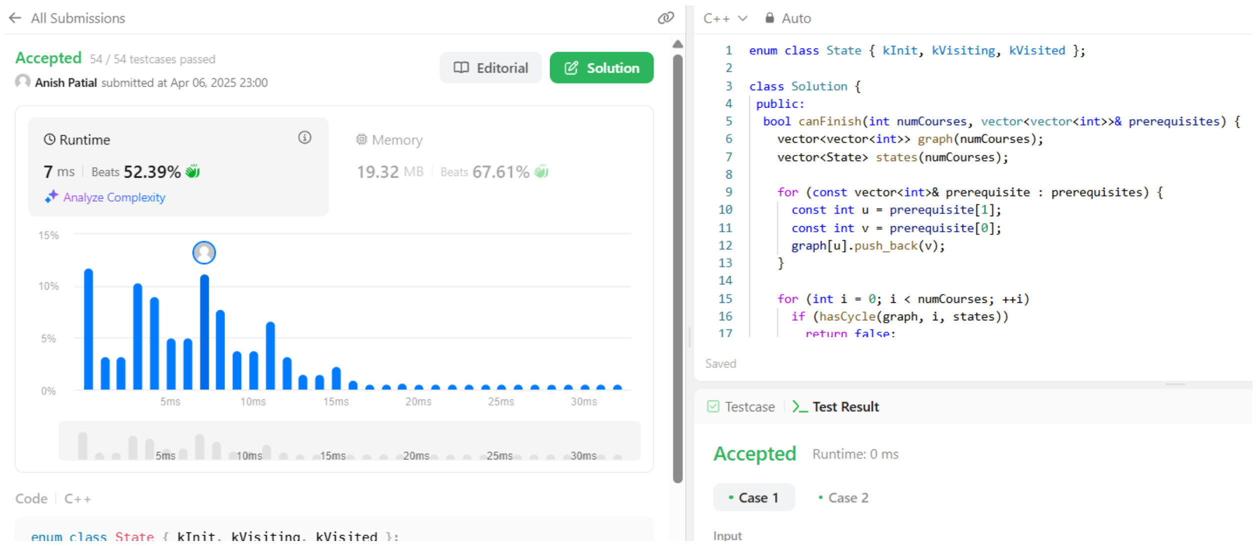
Accepted Runtime: 0 ms

Case 1 · Case 2 · Case 3

Input

7. Course Schedule

```
enum class State { kInit, kVisiting, kVisited };

class Solution {
 public:
  bool canFinish(int numCourses, vector<vector<int>>& prerequisites) {
    vector<vector<int>> graph(numCourses);
    vector<State> states(numCourses);

    for (const vector<int>& prerequisite : prerequisites) {
      const int u = prerequisite[1];
      const int v = prerequisite[0];
      graph[u].push_back(v);
    }

    for (int i = 0; i < numCourses; ++i)
      if (hasCycle(graph, i, states))
        return false;

    return true;
  }

 private:
  bool hasCycle(const vector<vector<int>>& graph, int u,
             vector<State>& states) {
    if (states[u] == State::kVisiting)
     return true;
    if (states[u] == State::kVisited)
     return false;
    states[u] = State::kVisiting;
    for (const int v : graph[u])
     if (hasCycle(graph, v, states))
      return true;
    states[u] = State::kVisited;
    return false;
  }
};
```



All Submissions

Accepted 54 / 54 testcases passed
Anish Patial submitted at Apr 06, 2025 23:00

Runtime
7 ms | Beats 52.39%

Memory
19.32 MB | Beats 67.61%

Code | C++

```
enum class State { kInit, kVisiting, kVisited };
```

C++ Auto

```
enum class State { kInit, kVisiting, kVisited };

class Solution {
 public:
  bool canFinish(int numCourses, vector<vector<int>>& prerequisites) {
    vector<vector<int>> graph(numCourses);
    vector<State> states(numCourses);

    for (const vector<int>& prerequisite : prerequisites) {
      const int u = prerequisite[1];
      const int v = prerequisite[0];
      graph[u].push_back(v);
    }

    for (int i = 0; i < numCourses; ++i)
      if (hasCycle(graph, i, states))
        return false;
```

Saved

Testcase | Test Result

Accepted Runtime: 0 ms

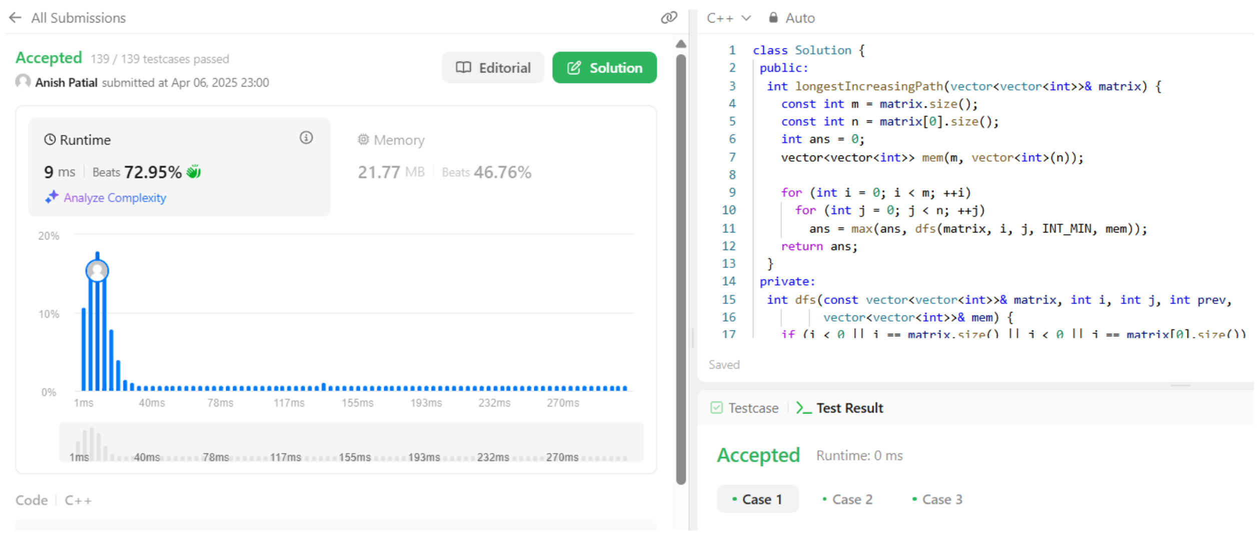
Case 1 · Case 2

Input

8. 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

Anish Patial submitted at Apr 06, 2025 23:00

Runtime 9 ms | Beats 72.95%

Memory 21.77 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;
  }
 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())
```

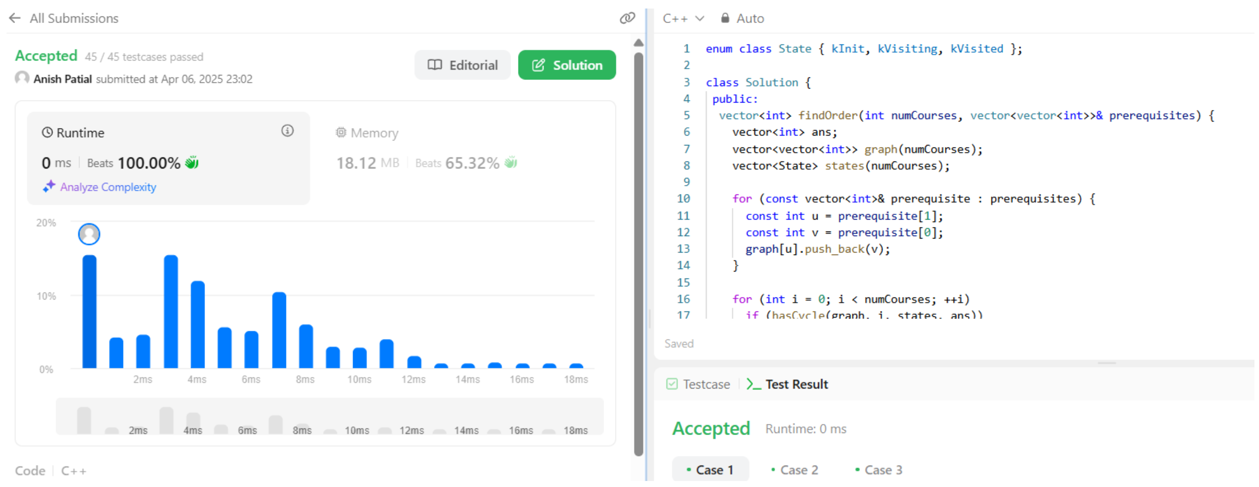
Test Result

Accepted Runtime: 0 ms

9. Course Schedule II

```
enum class State { kInit, kVisiting, kVisited };

class Solution {
 public:
  vector<int> findOrder(int numCourses, vector<vector<int>>& prerequisites) {
    vector<int> ans;
    vector<vector<int>> graph(numCourses);
    vector<State> states(numCourses);

    for (const vector<int>& prerequisite : prerequisites) {
      const int u = prerequisite[1];
      const int v = prerequisite[0];
      graph[u].push_back(v);
    }

    for (int i = 0; i < numCourses; ++i)
```

```
      if (hasCycle(graph, i, states, ans))
        return {};

    ranges::reverse(ans);
    return ans;
  }

 private:
  bool hasCycle(const vector<vector<int>>& graph, int u, vector<State>& states,
                vector<int>& ans) {
    if (states[u] == State::kVisiting)
      return true;
    if (states[u] == State::kVisited)
      return false;
    states[u] = State::kVisiting;
    for (const int v : graph[u])
      if (hasCycle(graph, v, states, ans))
        return true;
    states[u] = State::kVisited;
    ans.push_back(u);
    return false;
  }
};
```



Accepted 45 / 45 testcases passed

Anish Patial submitted at Apr 06, 2025 23:02

Runtime: 0 ms | Beats 100.00%

Memory: 18.12 MB | Beats 65.32%

```
enum class State { kInit, kVisiting, kVisited };

class Solution {
 public:
  vector<int> findOrder(int numCourses, vector<vector<int>>& prerequisites) {
    vector<int> ans;
    vector<vector<int>> graph(numCourses);
    vector<State> states(numCourses);

    for (const vector<int>& prerequisite : prerequisites) {
      const int u = prerequisite[1];
      const int v = prerequisite[0];
      graph[u].push_back(v);
    }

    for (int i = 0; i < numCourses; ++i)
      if (hasCycle(graph, i, states, ans))
```

Accepted Runtime: 0 ms