第八次作业

张睿恒 2024302182001

完成题目：208-简单，211-简单，139-中等，648-中等

T1 LeetCode208

```
class TrieNode {
public:
    TrieNode *child[26];
    bool isWord;
    TrieNode() {
        isWord = false;
        for (int i = 0; i < 26; i++) {
            child[i] = nullptr;
        }
    }
};

class Trie {
    TrieNode* root;
public:
    Trie() {
        root = new TrieNode();
    }

    void insert(string s) {
        TrieNode *p = root;
        for (int j = 0; j < s.length(); j++) {
            int i = s[j] - 'a';
            if (!p->child[i]) {
                p->child[i] = new TrieNode();
            }
            p = p->child[i];
        }
        p->isWord = true;
    }

    bool search(string key, bool prefix = false) {
        TrieNode *p = root;
        for (int j = 0; j < key.length(); j++) {
            int i = key[j] - 'a';
            if (!p->child[i]) {
                return false;
            }
            p = p->child[i];
        }
        if (prefix == false) {
            return p->isWord;
        }
        return true;
    }

    bool startsWith(string prefix) {
```

```
        return search(prefix, true);
    }
};
```

Accepted 16 / 16 testcases passed

ltheng submitted at Nov 26, 2025 17:25

Editorial | Solution

⏳ LIMITED TIME OFFER - $40 off Annual Subscription

LeetCode's Thanksgiving Sale IS NOW LIVE! Get $40 OFF...

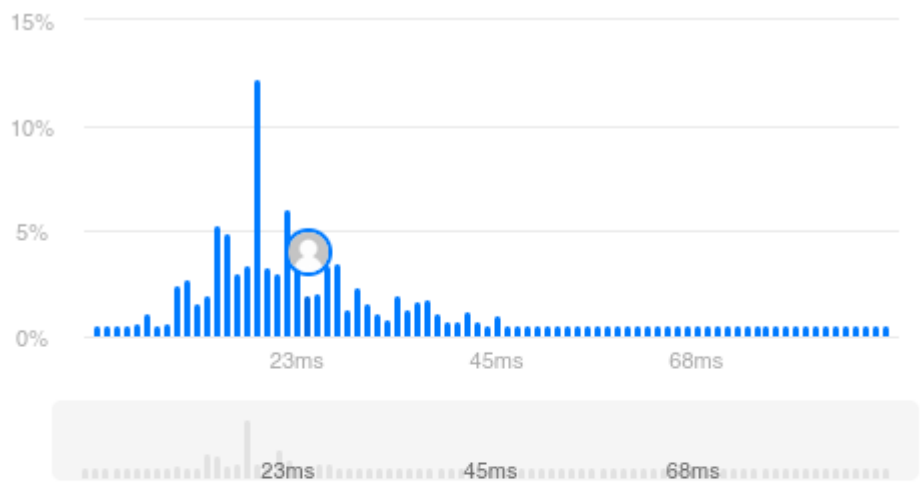
Runtime

24 ms | Beats **45.93%**

Analyze Complexity

Memory

50.58 MB | Beats **53.24%**



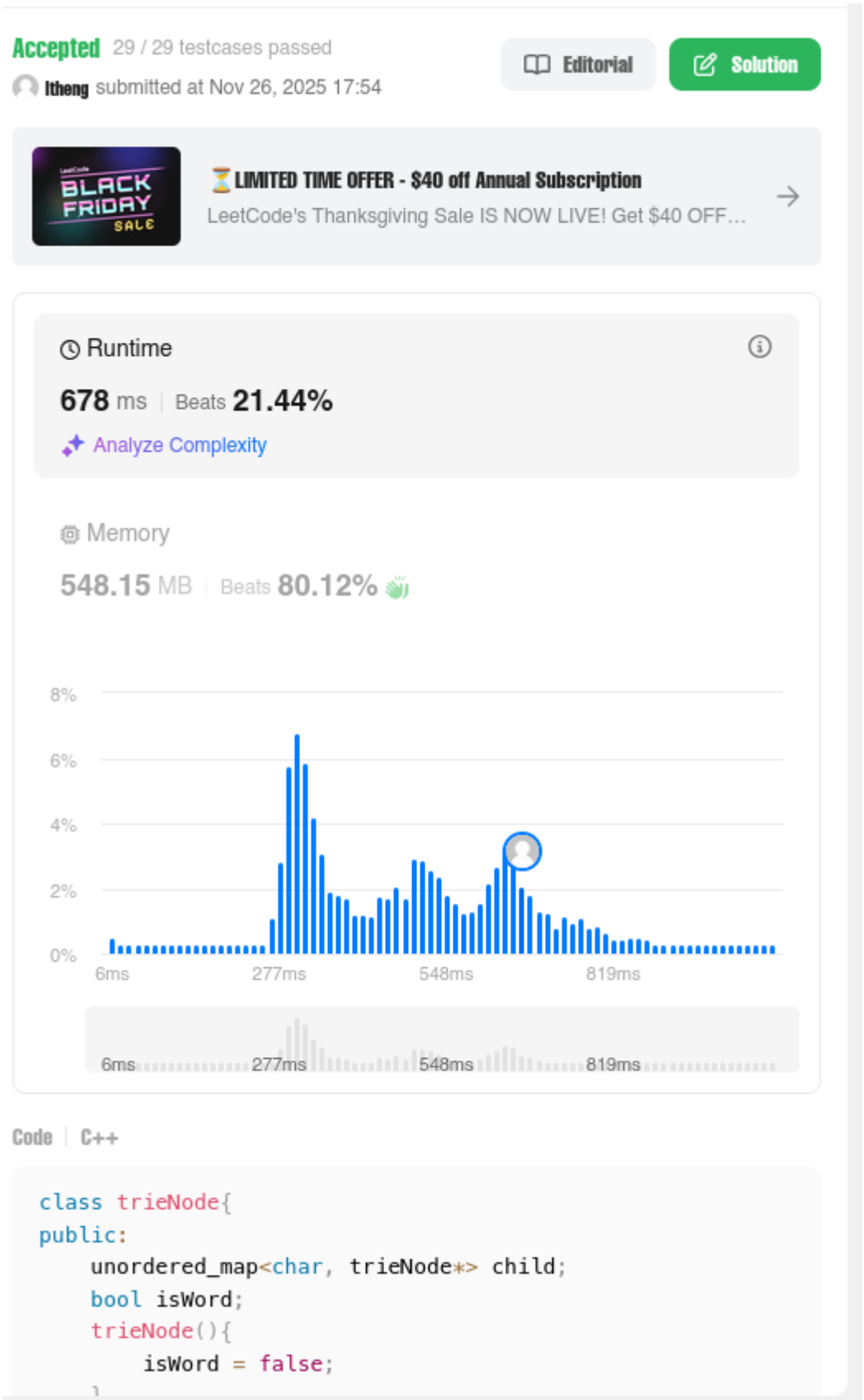
Code | C++

```
class TrieNode {
public:
    TrieNode *child[26];
    bool isWord;
    TrieNode() {
        isWord = false;
        for (int i = 0; i < 26; i++) {
```

T2 LeetCode211

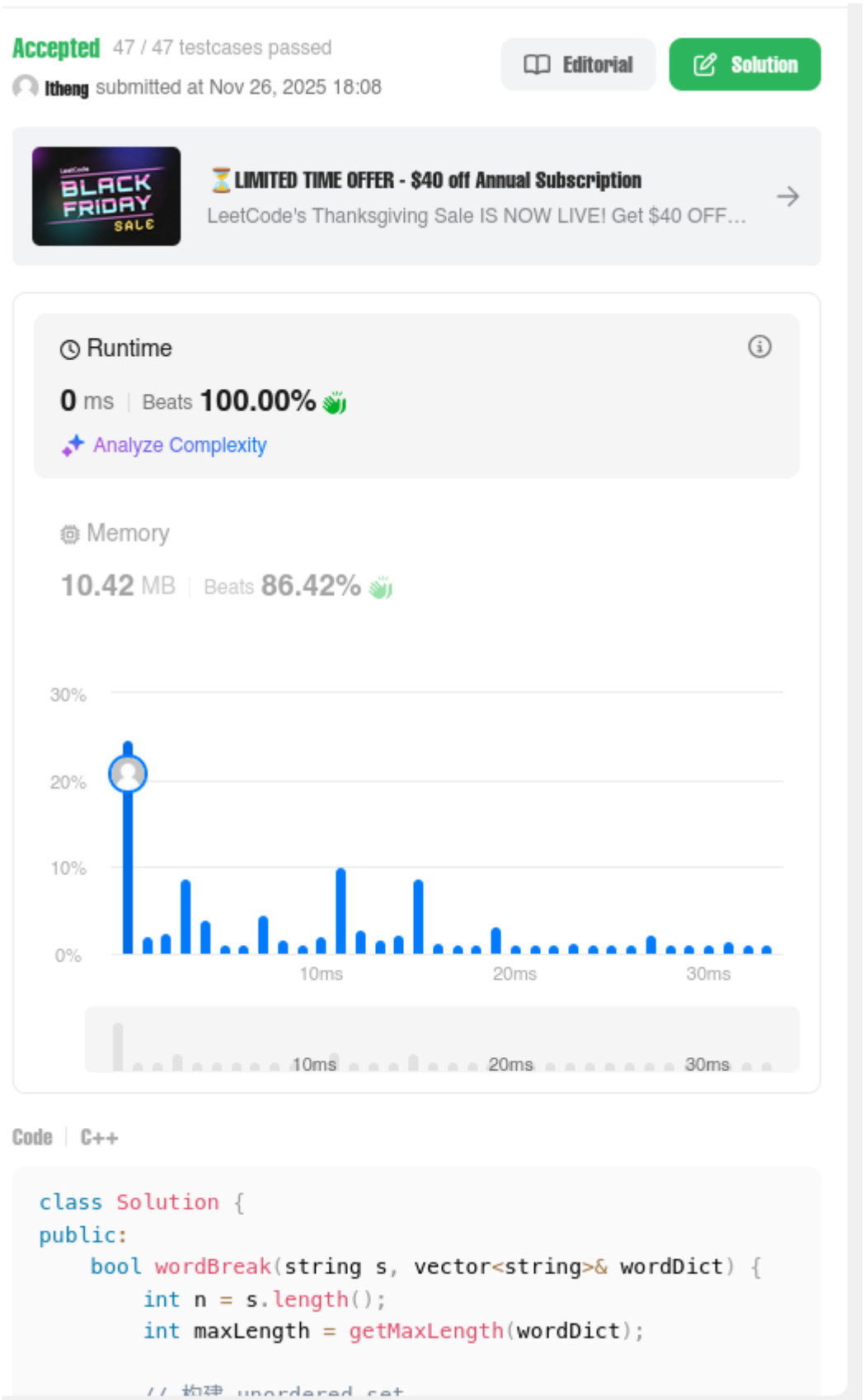
```
class trieNode{
public:
    unordered_map<char, trieNode*> child;
    bool isWord;
    trieNode(){
        isWord = false;
    }
};

class WordDictionary {
public:
    trieNode* root;

    WordDictionary() {
        root = new trieNode();
    }

    void addWord(string word) {
        trieNode* node = root;
        int len = word.length();
        for(int i = 0; i < len; ++i){
            char ch1 = word[i];
            if(node->child.find(ch1) == node->child.end()) {
                node->child[ch1] = new trieNode();
            }
            node = node->child[ch1];
        }
        node->isWord = true;
    }

    bool search(string word) {
        return search1(root, word, 0);
    }

    bool search1(trieNode* node, string& word, int index){
        if(index == word.length()) return node->isWord;

        char ch1 = word[index];

        if(ch1 == '.'){
            for(unordered_map<char, trieNode*>::iterator it = node->child.begin();
                it != node->child.end(); ++it){
                if(search1(it->second, word, index + 1)) {
                    return true;
                }
            }
            return false;
```

```
        }
        else{
            if(node->child.find(ch1) == node->child.end()) {
                return false;
            }
            return search1(node->child[ch1], word, index + 1);
        }
    }
};
```



T3 LeetCode139

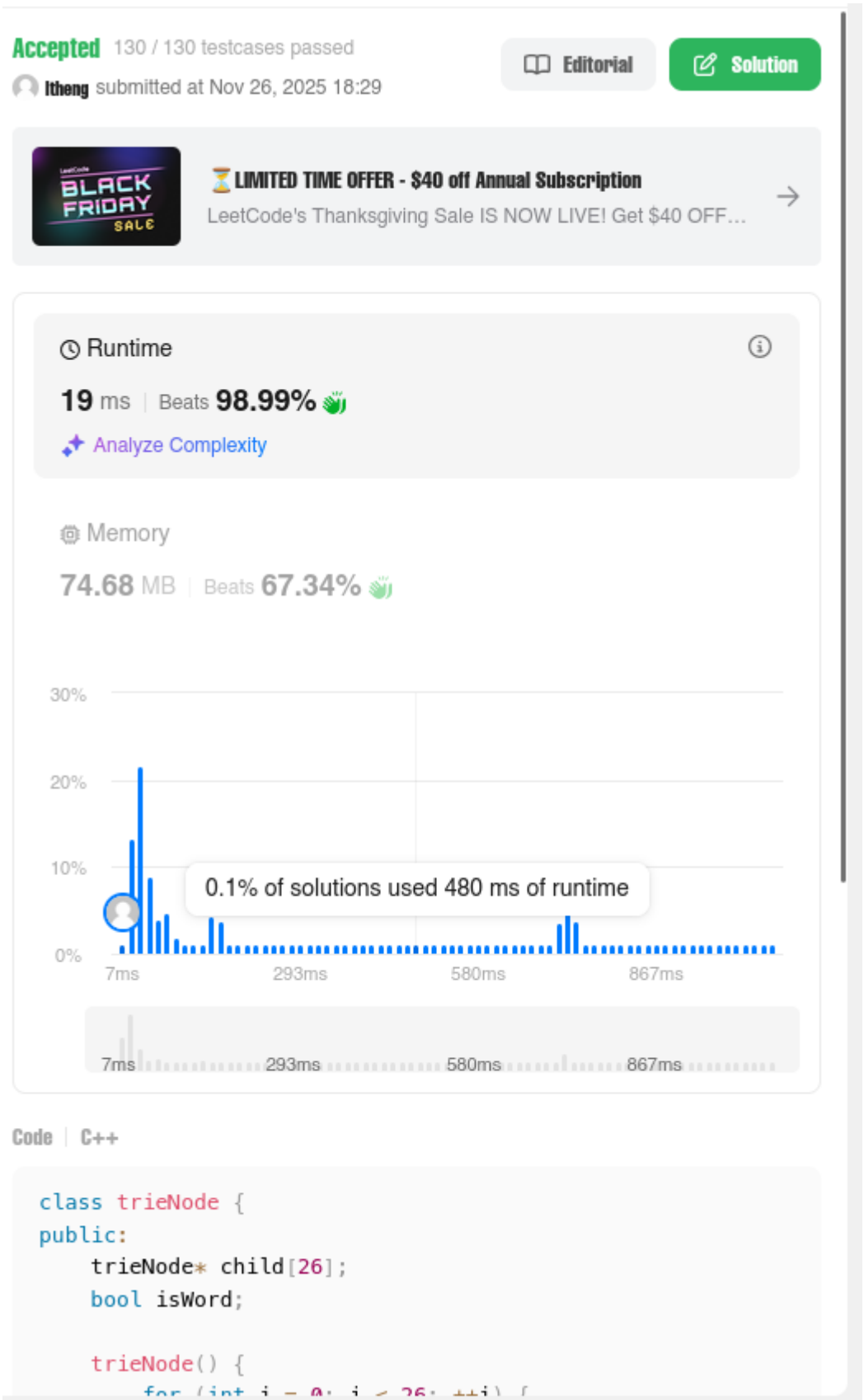
```
class Solution {
public:
    bool wordBreak(string s, vector<string>& wordDict) {
        int n = s.length();
        int maxLength = getMaxLength(wordDict);

        unordered_set<string> wordSet;
        for (int i = 0; i < wordDict.size(); i++) {
            wordSet.insert(wordDict[i]);
        }

        vector<bool> dp(n + 1, false);
        dp[0] = true;

        for (int i = 1; i <= n; ++i) {
            for (int j = i - 1; j >= 0; --j) {
                if (i - j > maxLength) {
                    break;
                }
                if (dp[j] && wordSet.count(s.substr(j, i - j))) {
                    dp[i] = true;
                    break;
                }
            }
        }

        return dp[n];
    }

private:
    int getMaxLength(const vector<string>& wordDict) {
        int maxLen = 0;
        for (int i = 0; i < wordDict.size(); i++) {
            if (wordDict[i].length() > maxLen) {
                maxLen = wordDict[i].length();
            }
        }
        return maxLen;
    }
};
```



T4 Leetcode648

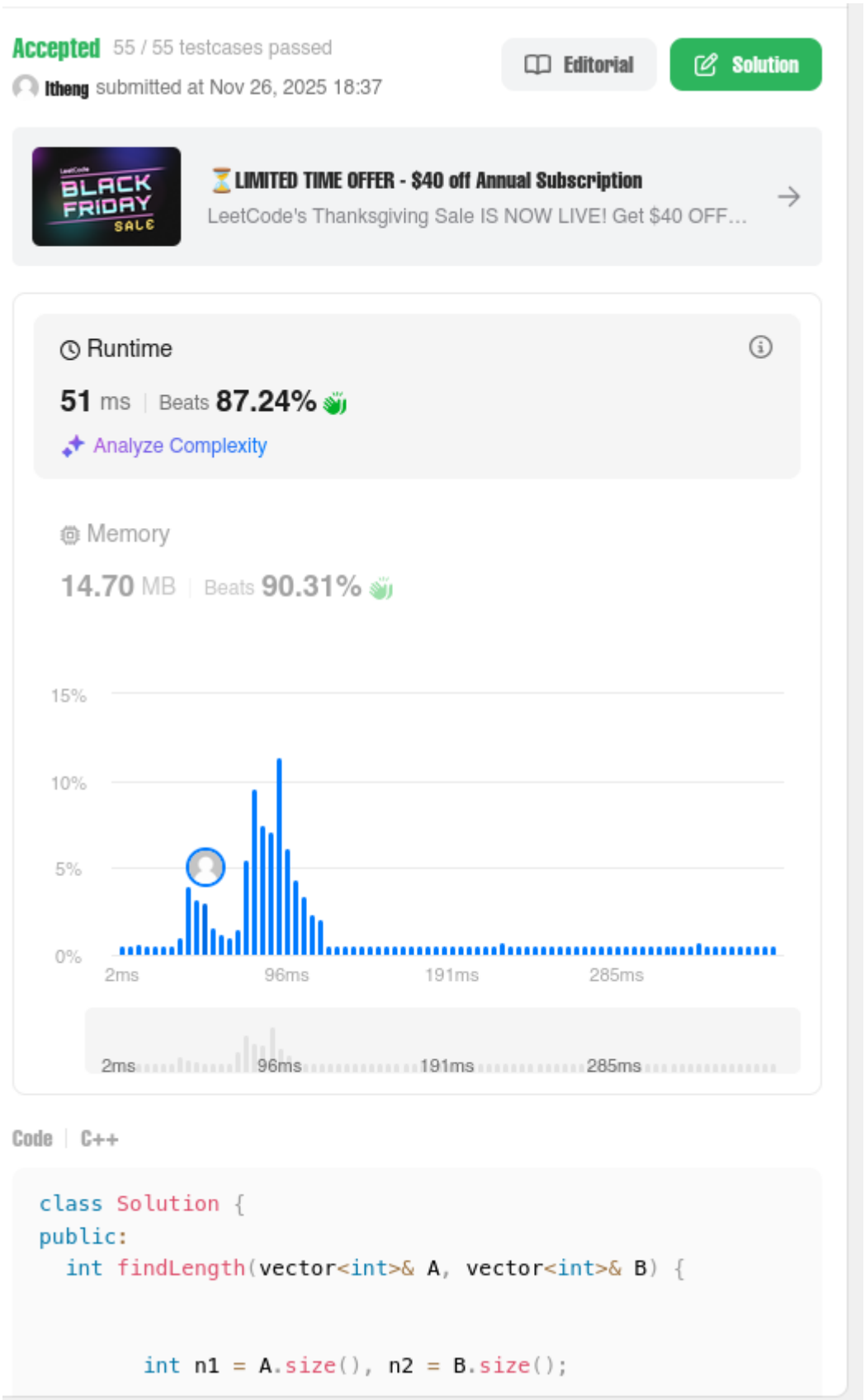
```
class trieNode {
public:
    trieNode* child[26];
    bool isWord;

    trieNode() {
        for (int i = 0; i < 26; ++i) {
            child[i] = NULL;
        }
        isWord = false;
    }
};

class Solution {
private:
    trieNode* root;

    void insert(string word) {
        trieNode* node = root;
        int len = word.length();
        for (int i = 0; i < len; ++i) {
            int num = word[i] - 'a';
            if (node->child[num] == NULL) {
                node->child[num] = new trieNode();
            }
            node = node->child[num];
        }
        node->isWord = true;
    }

    string search(string word) {
        trieNode* node = root;
        string res;
        int len = word.length();
        for (int i = 0; i < len; ++i) {
            int num = word[i] - 'a';
            if (node->child[num] == NULL) {
                return word;
            }
            res += word[i];
            if (node->child[num]->isWord == true) {
                return res;
            }
            node = node->child[num];
        }
        return word;
    }
```

```
public:
    string replaceWords(vector<string>& dictionary, string sentence) {
        root = new trieNode();


        for (int i = 0; i < dictionary.size(); i++) {
            insert(dictionary[i]);
        }


        stringstream ss(sentence);
        string word;
        string res;

        while (ss >> word) {
            if (!res.empty()) {
                res += " ";
            }
            res += search(word);
        }

        return res;
    }
};
```



T5 LeetCode718

```
class Solution {
public:
  int findLength(vector<int>& A, vector<int>& B) {


        int n1 = A.size(), n2 = B.size();

        vector<int> dp(n2+1, 0);
        int ans = 0;

        for(int i = 0; i < n1; ++i) {
            for(int j = n2 - 1; j >= 0; --j) {

                if(A[i] == B[j]) dp[j + 1] = 1 + dp[j];
                else dp[j + 1] = 0;

                ans = max(ans, dp[j + 1]);
            }

        }

        return ans;
    }
};
```



T6 LeetCode1138

```
class Solution {
public:
    pair<int, int> findPos(char c1, char c2) {
        int posC1 = c1 - 'a';
        int posC2 = c2 - 'a';
        int x = posC2 % 5 - posC1 % 5;   // 注意：c2 - c1，表示需要移动的方向
        int y = posC2 / 5 - posC1 / 5;
        return make_pair(x, y);
    }

    string alphabetBoardPath(string target) {
        char curr = 'a';
        string ans = "";

        for (int i = 0; i < target.length(); i++) {
            char c = target[i];
            pair<int, int> pos = findPos(curr, c);

            // 关键：移动顺序很重要！
            // 先向上（避免从z出发时越界）
            while (pos.second < 0) {
                ans += "U";
                pos.second++;
            }

            // 再向左（避免移动到z时越界）
            while (pos.first < 0) {
                ans += "L";
                pos.first++;
            }

            // 再向右
            while (pos.first > 0) {
                ans += "R";
                pos.first--;
            }

            // 最后向下（确保到z时最后才下移）
            while (pos.second > 0) {
                ans += "D";
                pos.second--;
            }

            ans += "!";
            curr = c;
        }
```

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

Accepted 61 / 61 testcases passed

Itheng submitted at Nov 26, 2025 18:39

Solution

⏳ LIMITED TIME OFFER - $40 off Annual Subscription

LeetCode's Thanksgiving Sale IS NOW LIVE! Get $40 OFF...

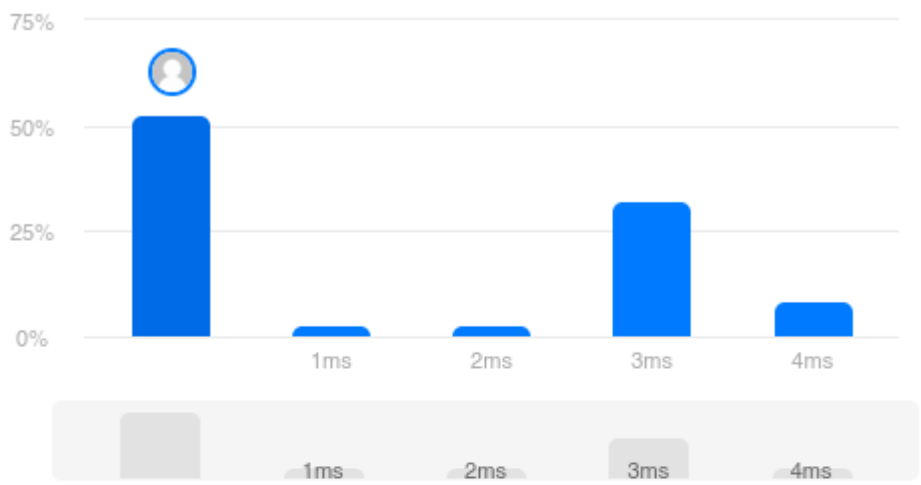
Runtime

0 ms | Beats **100.00%**

Analyze Complexity

Memory

7.99 MB | Beats **96.88%**



Code | C++

```
class Solution {
public:
    pair<int, int> findPos(char c1, char c2) {
        int posC1 = c1 - 'a';
        int posC2 = c2 - 'a';
        int x = posC2 % 5 - posC1 % 5;   // 注意: c2 - c1, 
        int y = posC2 / 5 - posC1 / 5;
```