ASSIGNMENT 10

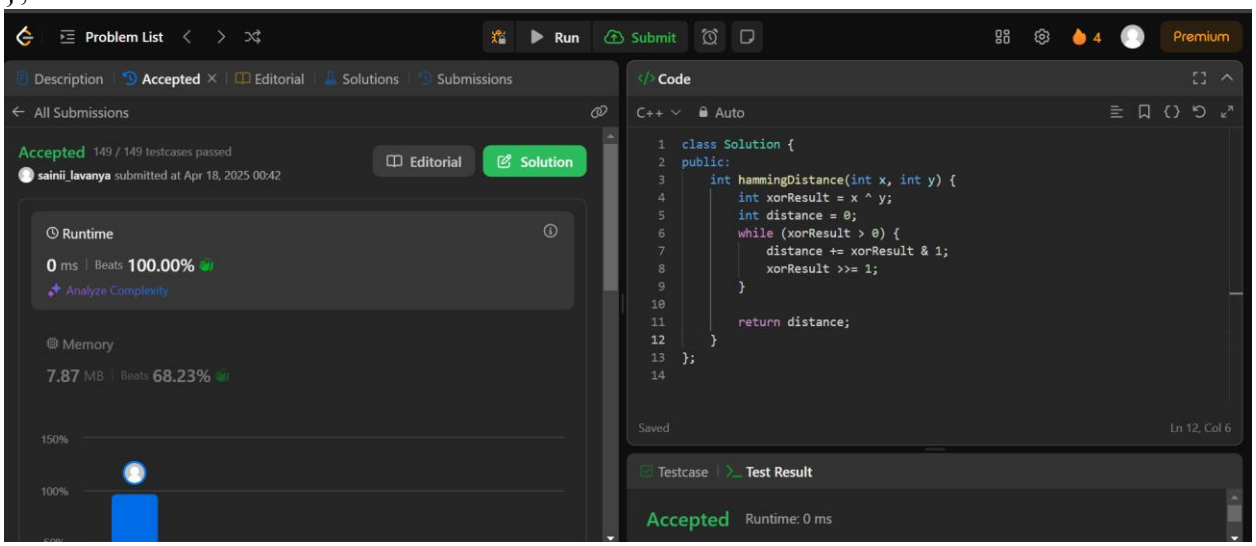
AP LAB

Lavanya Saini – 22BCS15497

22BCS IOT 614-B

1. Hamming Distance

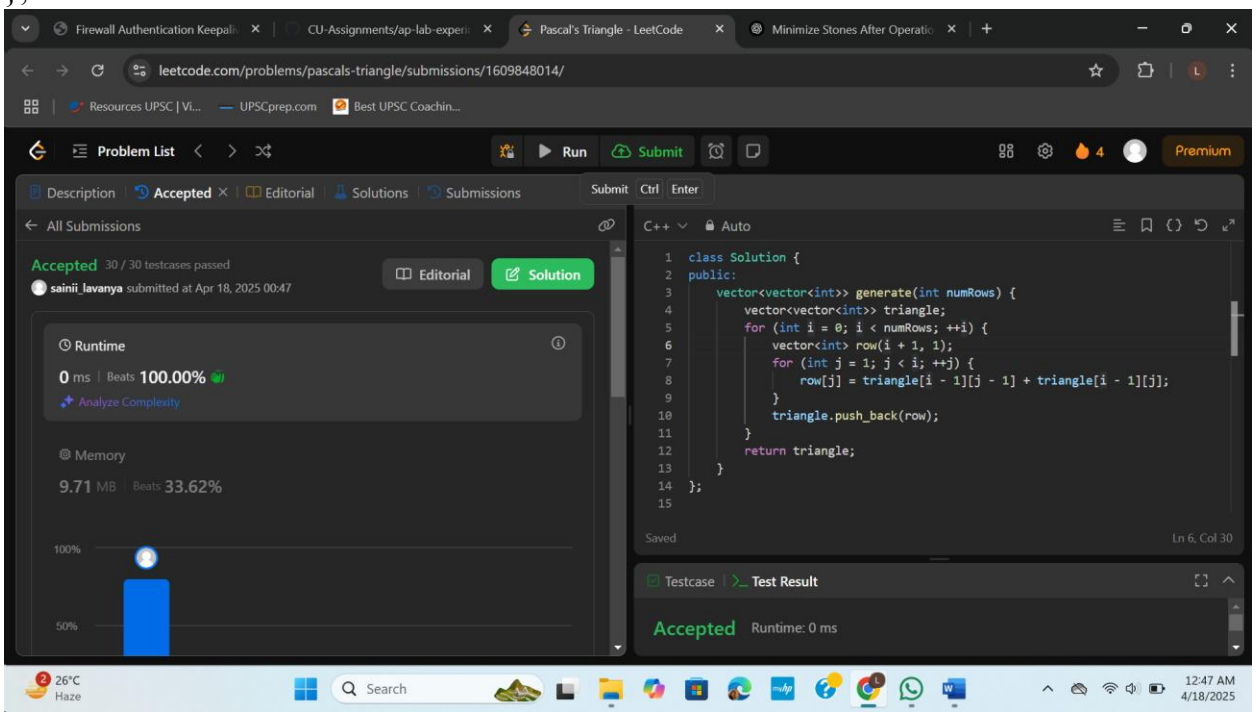
```
class Solution {
public:
    int hammingDistance(int x, int y) {
        int xorResult = x ^ y;
        int distance = 0;
        while (xorResult > 0) {
            distance += xorResult & 1;
            xorResult >>= 1;
        }

        return distance;
    }
};
```



2. Pascal's Triangle

```
class Solution {
public:
    vector<vector<int>> generate(int numRows) {
```

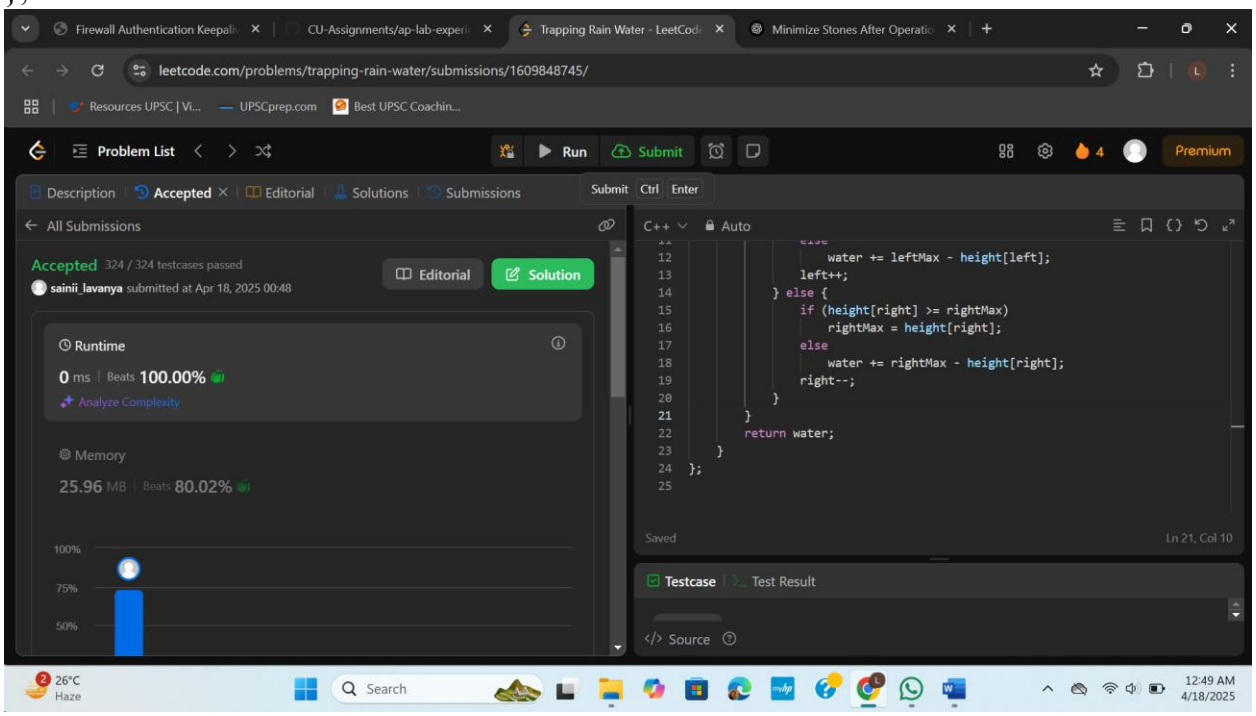
```
        vector<vector<int>> triangle;
        for (int i = 0; i < numRows; ++i) {
            vector<int> row(i + 1, 1);
            for (int j = 1; j < i; ++j) {
                row[j] = triangle[i - 1][j - 1] + triangle[i - 1][j];
            }
            triangle.push_back(row);
        }
        return triangle;
    }
};
```



3. Trapping Rain Water

```
class Solution {
public:
    int trap(vector<int>& height) {
        int left = 0, right = height.size() - 1;
        int leftMax = 0, rightMax = 0;
        int water = 0;
        while (left < right) {
            if (height[left] < height[right]) {
                if (height[left] >= leftMax)
                    leftMax = height[left];
                else
                    water += leftMax - height[left];
```

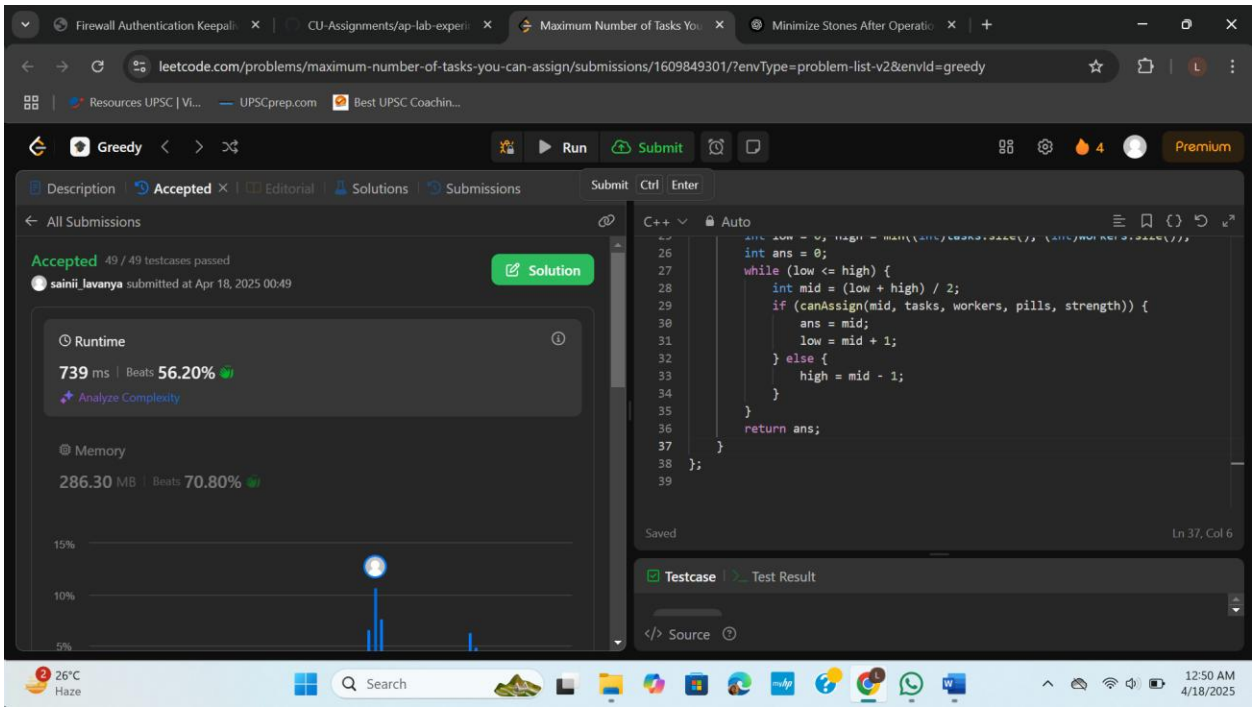
```
            left++;
        } else {
            if (height[right] >= rightMax)
                rightMax = height[right];
            else
                water += rightMax - height[right];
            right--;
        }
    }
    return water;
  }
};
```



4. Maximum Number of Tasks You Can Assign

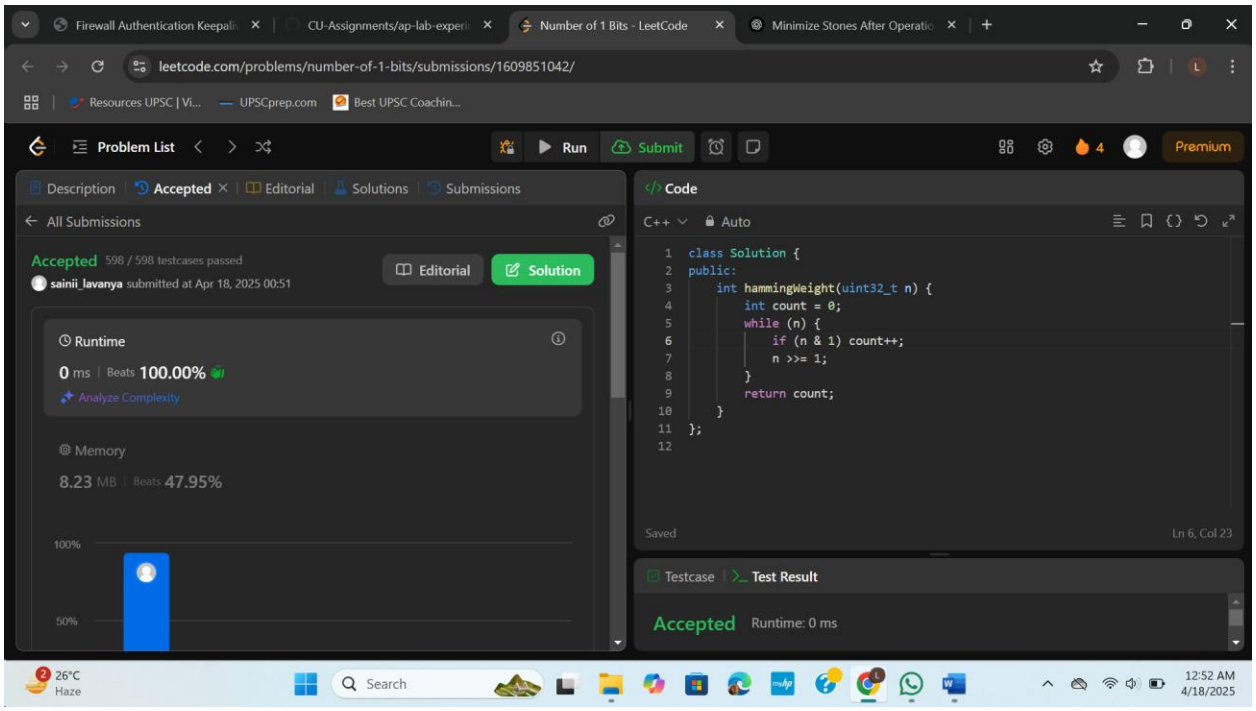
```
class Solution {
public:
    bool canAssign(int k, vector<int>& tasks, vector<int>& workers, int pills, int strength) {
        multiset<int> wk(workers.end() - k, workers.end());
        int remainingPills = pills;
        for (int i = k - 1; i >= 0; i--) {
            int t = tasks[i];
            auto it = wk.lower_bound(t);
            if (it != wk.end()) {
                wk.erase(it);
            } else {
```

```
            if (remainingPills == 0) return false;
            auto it2 = wk.lower_bound(t - strength);
            if (it2 == wk.end()) return false;
            wk.erase(it2);
            remainingPills--;
        }
    }
    return true;
}

int maxTaskAssign(vector<int>& tasks, vector<int>& workers, int pills, int strength) {
    sort(tasks.begin(), tasks.end());
    sort(workers.begin(), workers.end());
    int low = 0, high = min((int)tasks.size(), (int)workers.size());
    int ans = 0;
    while (low <= high) {
        int mid = (low + high) / 2;
        if (canAssign(mid, tasks, workers, pills, strength)) {
            ans = mid;
            low = mid + 1;
        } else {
            high = mid - 1;
        }
    }
    return ans;
}
};
```



5. Number of 1 Bits

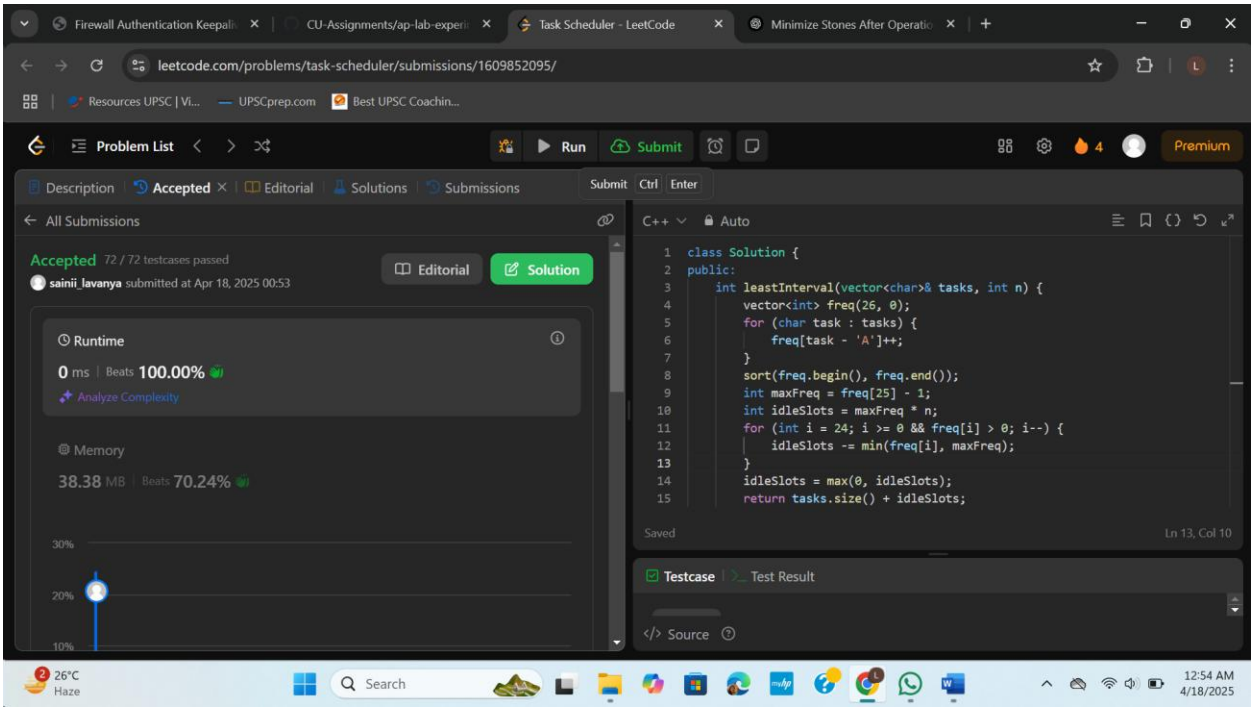
```
class Solution {
public:
    int hammingWeight(uint32_t n) {
        int count = 0;
        while (n) {
            if (n & 1) count++;
            n >>= 1;
        }
        return count;
    }
};
```



6. Task Scheduler

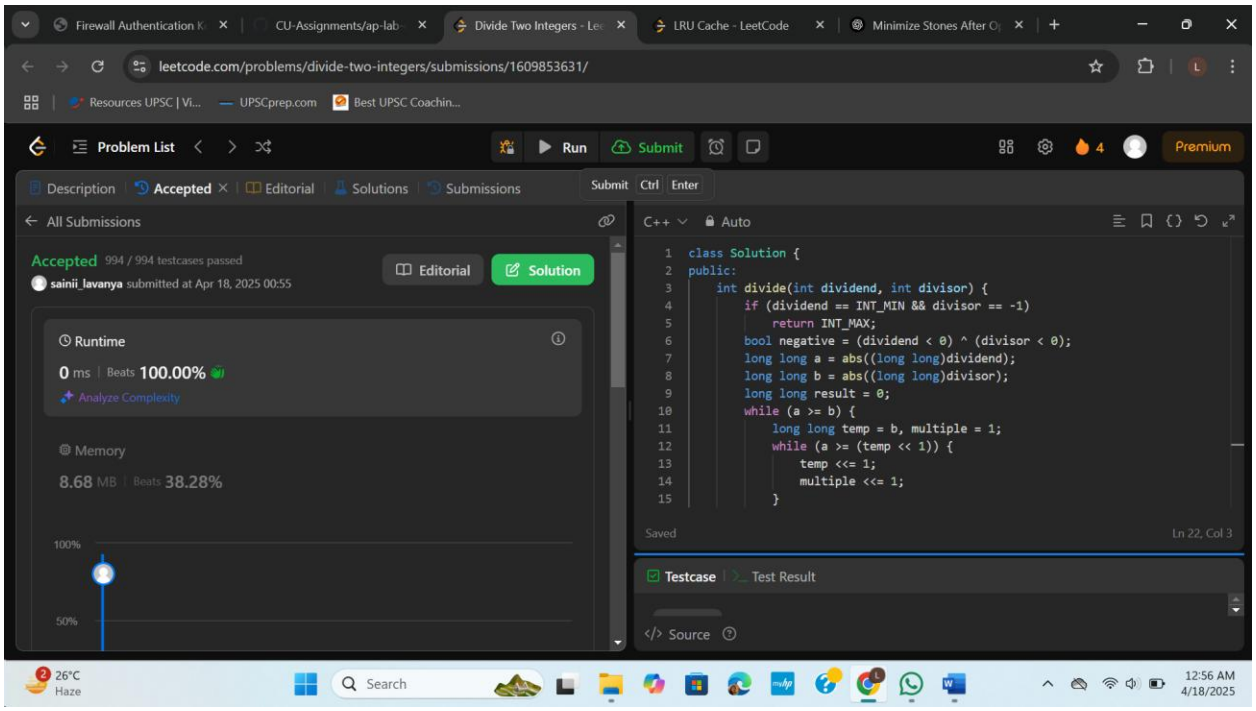
```
class Solution {
public:
    int leastInterval(vector<char>& tasks, int n) {
        vector<int> freq(26, 0);
        for (char task : tasks) {
            freq[task - 'A']++;
        }
        sort(freq.begin(), freq.end());
        int maxFreq = freq[25] - 1;
        int idleSlots = maxFreq * n;
        for (int i = 24; i >= 0 && freq[i] > 0; i--) {
            idleSlots -= min(freq[i], maxFreq);
        }
        idleSlots = max(0, idleSlots);
        return tasks.size() + idleSlots;
    }
```

};



7. Divide Two Integers

```
class Solution {
public:
    int divide(int dividend, int divisor) {
        if (dividend == INT_MIN && divisor == -1)
            return INT_MAX;
        bool negative = (dividend < 0) ^ (divisor < 0);
        long long a = abs((long long)dividend);
        long long b = abs((long long)divisor);
        long long result = 0;
        while (a >= b) {
            long long temp = b, multiple = 1;
            while (a >= (temp << 1)) {
                temp <<= 1;
                multiple <<= 1;
            }
            a -= temp;
            result += multiple;
        }
        result = negative ? -result : result;
        return result;
    }
};
```

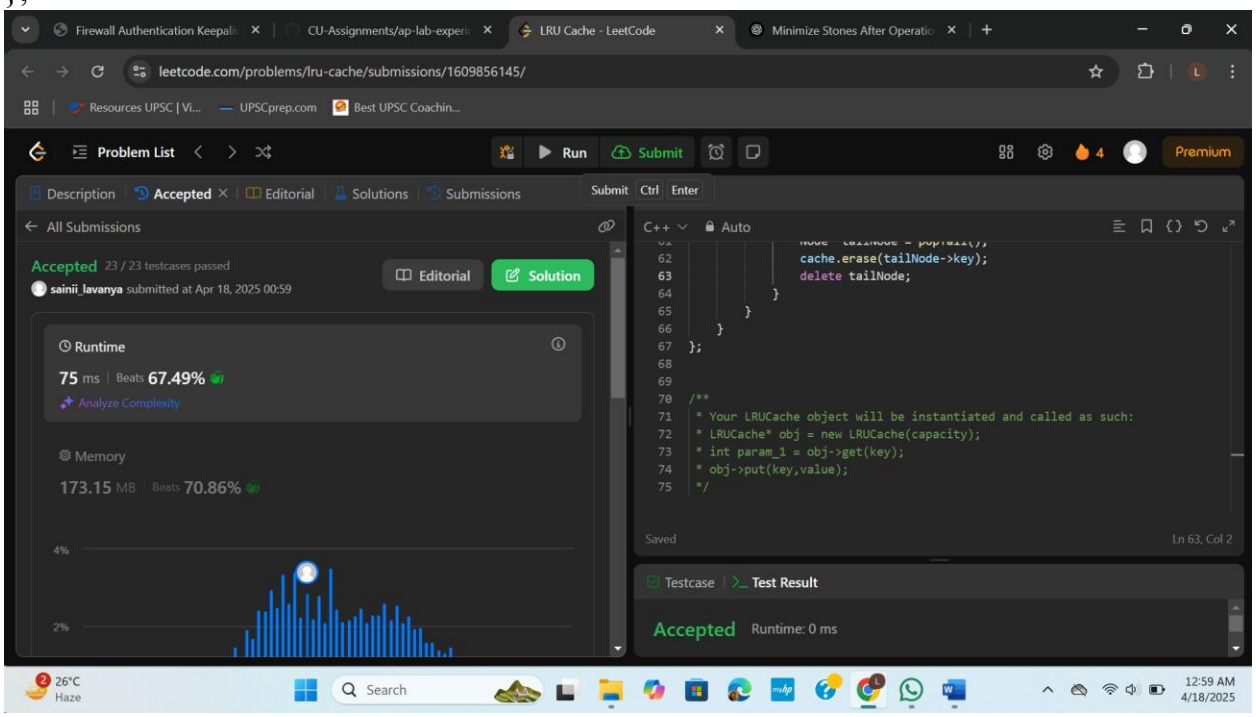


8. LRU Cache

```
class LRUCache {
private:
    struct Node {
        int key, value;
        Node* prev;
        Node* next;
        Node(int k, int v) : key(k), value(v), prev(nullptr), next(nullptr) {}
    };

    unordered_map<int, Node*> cache;
    int capacity;
    Node* head;
    Node* tail;

    void addNode(Node* node) {
        node->next = head->next;
        node->prev = head;
        head->next->prev = node;
        head->next = node;
    }
    void removeNode(Node* node) {
        node->prev->next = node->next;
        node->next->prev = node->prev;
```

```
    }
    void moveToHead(Node* node) {
        removeNode(node);
        addNode(node);
    }
    Node* popTail() {
        Node* node = tail->prev;
        removeNode(node);
        return node;
    }

public:
    LRUCache(int capacity) : capacity(capacity) {
        head = new Node(0, 0);
        tail = new Node(0, 0);
        head->next = tail;
        tail->prev = head;
    }

    int get(int key) {
        if (cache.find(key) == cache.end())
            return -1;
        Node* node = cache[key];
        moveToHead(node);
        return node->value;
    }

    void put(int key, int value) {
        if (cache.find(key) != cache.end()) {
            Node* node = cache[key];
            node->value = value;
            moveToHead(node);
        } else {
            Node* newNode = new Node(key, value);
            cache[key] = newNode;
            addNode(newNode);
            if (cache.size() > capacity) {
                Node* tailNode = popTail();
                cache.erase(tailNode->key);
                delete tailNode;
            }
        }
    }
```

};


Firewall Authentication Keepali
CU-Assignments/ap-lab-experi
LRU Cache - LeetCode
Minimize Stones After Operatio
leetcode.com/problems/lru-cache/submissions/1609856145/
Resources UPSC | Vi...
UPSCprep.com
Best UPSC Coachin...
Problem List
Run
Submit
Premium
Description
Accepted
Editorial
Solutions
Submissions
Submit Ctrl Enter
All Submissions
Accepted 23 / 23 testcases passed
sainii_lavanya submitted at Apr 18, 2025 00:59
Editorial
Solution
Runtime
75 ms | Beats 67.49%
Analyze Complexity
Memory
173.15 MB | Beats 70.86%
4%
2%
C++
Auto
cache.erase(tailNode->key);
delete tailNode;
/**
* Your LRUCache object will be instantiated and called as such:
* LRUCache* obj = new LRUCache(capacity);
* int param_1 = obj->get(key);
* obj->put(key,value);
*/
Saved
Ln 63, Col 2
Testcase
Test Result
Accepted
Runtime: 0 ms
26°C
Haze
Search
12:59 AM
4/18/2025