**1 [Longest Nice Substring](https://leetcode.com/problems/longest-nice-substring/description/)**

class Solution {

public:

string longestNiceSubstring(string s) {

int n = s.size();

int k = -1, mx = 0;

for (int i = 0; i < n; ++i) {

unordered\_set<char> ss;

for (int j = i; j < n; ++j) {

ss.insert(s[j]);

bool ok = true;

for (auto& a : ss) {

char b = a ^ 32;

if (!(ss.count(a) && ss.count(b))) {

ok = false;

break;

}

}

if (ok && mx < j - i + 1) {

mx = j - i + 1;

k = i;

}

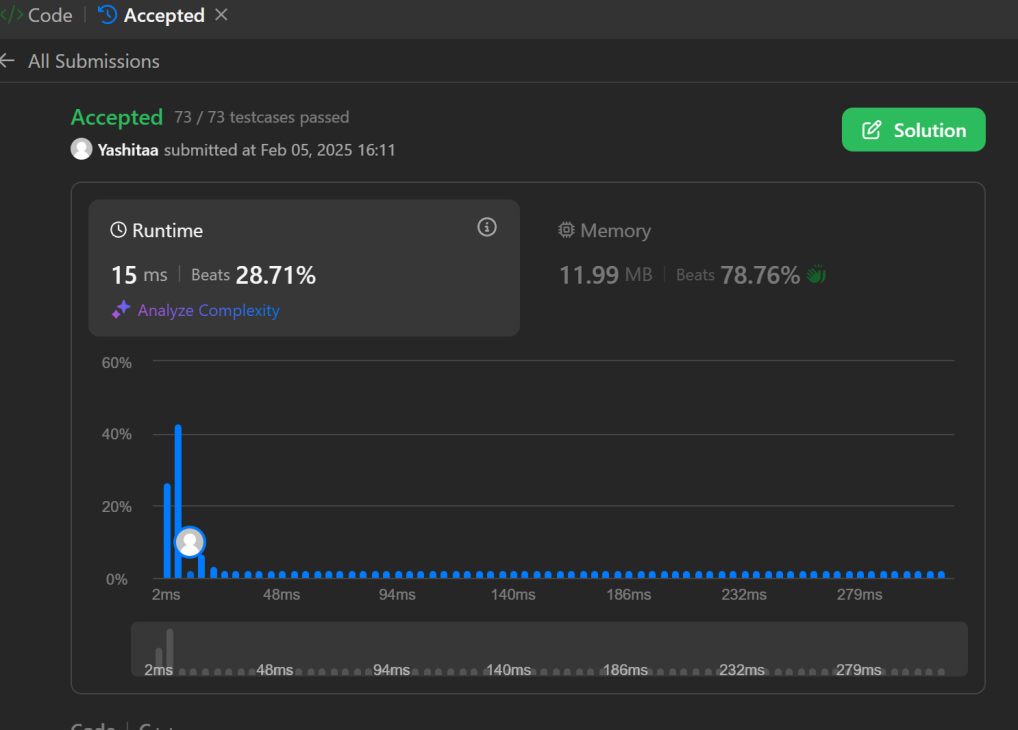
}

}

return k == -1 ? "" : s.substr(k, mx);

}

};



**2. Reverse bits**

class Solution {

public:

uint32\_t reverseBits(uint32\_t n) {

uint32\_t ans = 0;

for (int i = 0; i < 32 && n; ++i) {

ans |= (n & 1) << (31 - i);

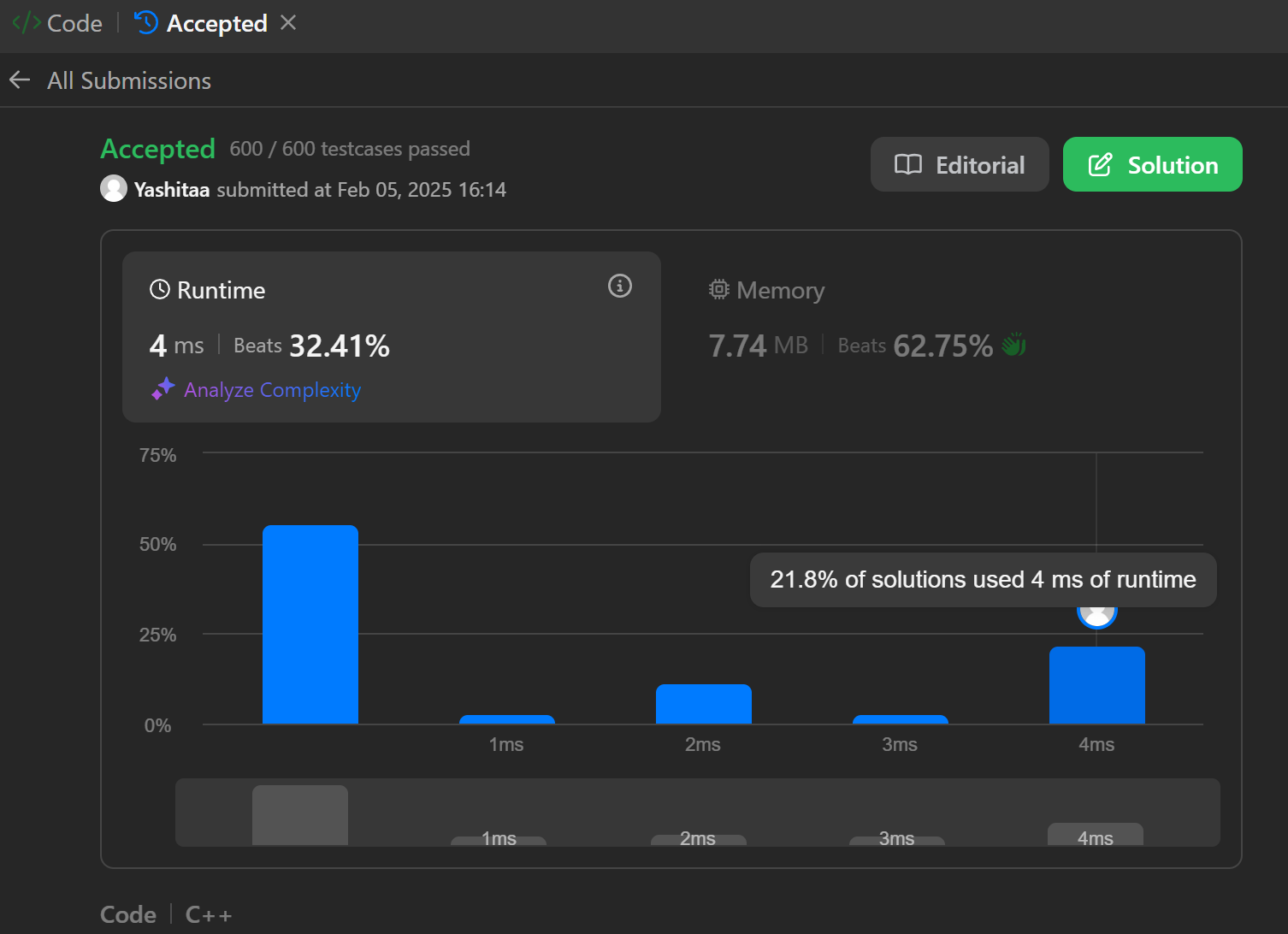
n >>= 1;

}

return ans;

}

};



**3 Number of 1 bit**

class Solution {

public:

int hammingWeight(int n) {

int count=0;

while(n!=0){

if(n%2!=0) count++;

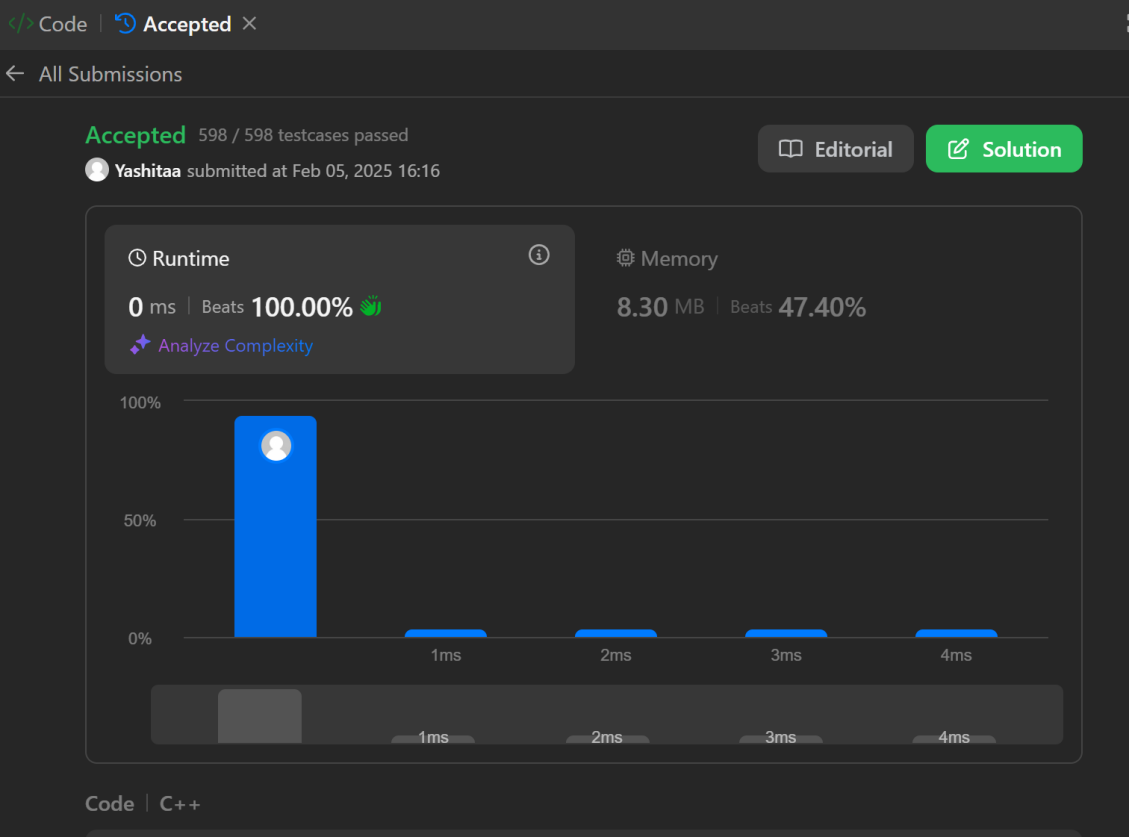
n=n>>1;

}

return count;

}

};



1. **maximum Subarray**

class Solution {

public:

int maxSubArray(vector<int>& nums) {

int ans = nums[0], f = nums[0];

for (int i = 1; i < nums.size(); ++i) {

f = max(f, 0) + nums[i];

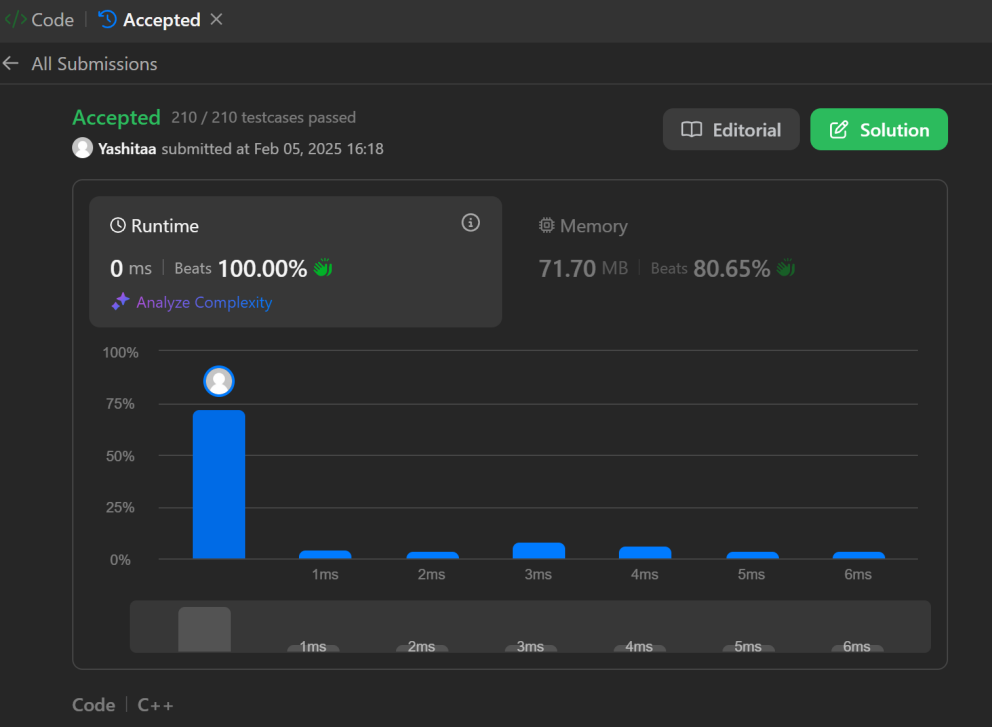
ans = max(ans, f);

}

return ans;

}

};



**5. Search a 2D Matrix**

class Solution {

public:

bool searchMatrix(vector<vector<int>>& matrix, int target) {

int i=0; int j=matrix[0].size()-1,a;

while(i<matrix.size() && j>=0){

a=matrix[i][j];

if(a==target) return true;

else if(a>target) j--;

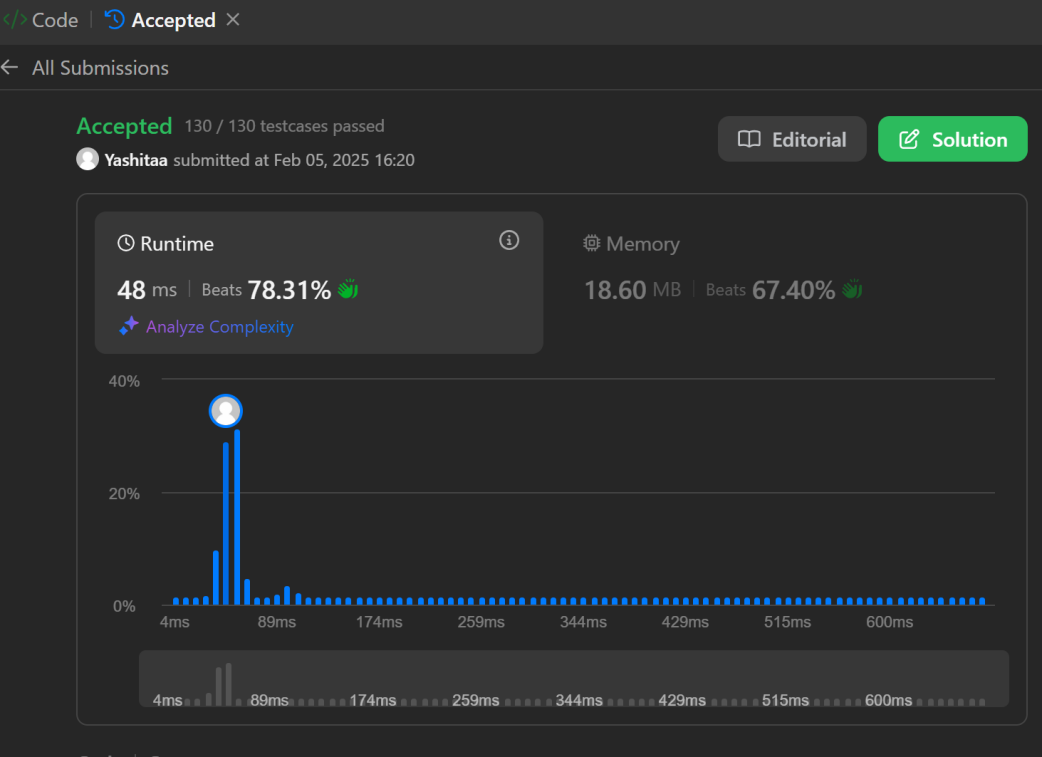
else i++;

}

return false;

}

};



**6. Search Pow**

class Solution {

const int base = 1337;

int powmod(int a, int k){

a %= base;

int result = 1;

for (int i = 0; i < k; ++i) result = (result \* a) % base;

return result;

}

public:

int superPow(int a, vector<int>& b) {

if (b.empty()) return 1;

int last\_digit = b.back();

b.pop\_back();

return powmod(superPow(a, b), 10) \* powmod(a, last\_digit) % base;

}

};

