# NAME- Neeraj Kumar| UID- 22BCS14147 | SECTION- 601/A

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

class Solution { public:

string longestNiceSubstring(string s) { if(s.size()<2) return "";

unordered\_set<char>uset; for(int i=0;i<s.size();i++){

uset.insert(s[i]);

}

for(int i=0;i<s.size();i++){

if(uset.count(tolower(s[i]))==true && uset.count(toupper(s[i]))==true) continue;

string prev=longestNiceSubstring(s.substr(0,i)); string next=longestNiceSubstring(s.substr(i+1));

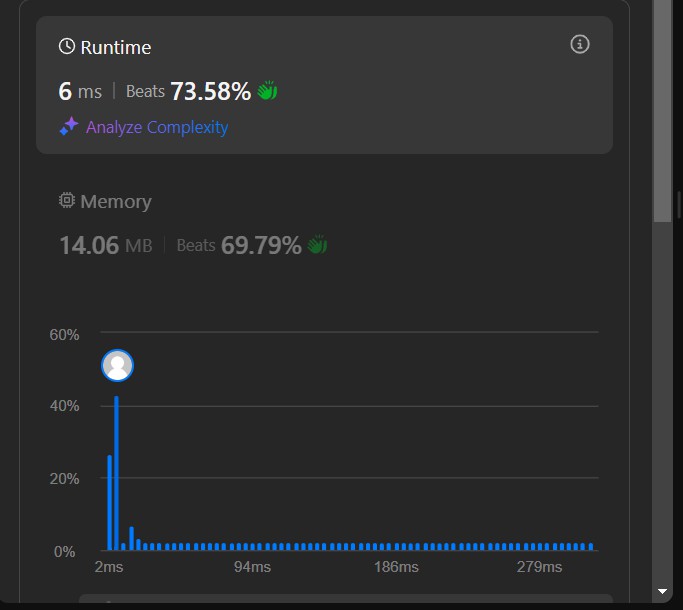
return prev.size()>=next.size()?prev:next;

}

return s;

}

};



# 2. Reverse bits

class Solution { public:

uint32\_t reverseBits(uint32\_t n) { uint32\_t result=0;

for(int i = 0;i < 32; i++){ result <<=1;

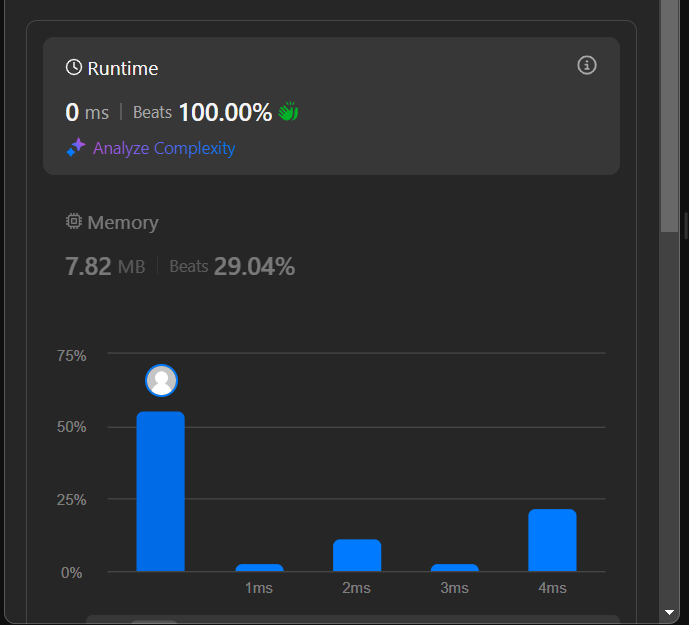
if(n&1) result++; n >>=1;

}

return result;

}

};



# 3 Number of 1 bit

class Solution { public:

int hammingWeight(int n) { int count=0;

while(n){

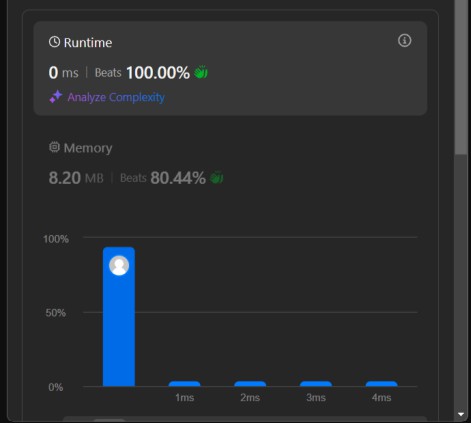
if(n & 1) count++; n >>= 1;

}

return count;

}

};



# Maximum Subarray

class Solution { public:

int maxSubArray(vector<int>& nums) { int current\_sum = nums[0];

int max\_sum = nums[0];

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

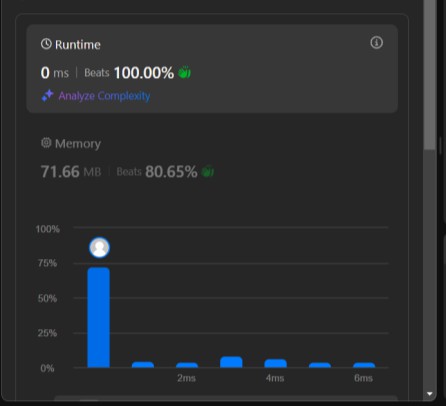
current\_sum = std::max(nums[i], current\_sum + nums[i]); max\_sum = std::max(max\_sum, current\_sum);

}

return max\_sum;

}

};



# Search a 2D Matrix

class Solution { public:

bool searchMatrix(vector<vector<int>>& matrix, int target) { if (matrix.empty() || matrix[0].empty()) {

return false;

}

int rows = matrix.size(); int cols = matrix[0].size(); int row = 0;

int col = cols - 1;

while (row < rows && col >= 0) { if (matrix[row][col] == target) {

return true;

} else if (matrix[row][col] > target) {

--col;

} else {

++row;

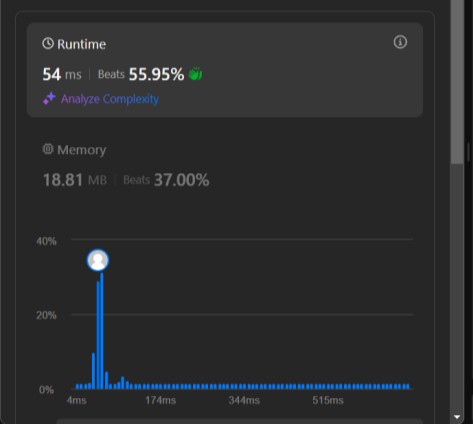
}

}

return false;

}

};



# Search Pow

class Solution { public:

int modPow(int a, int b, int mod) { int result = 1;

a %= mod; while (b > 0) {

if (b % 2 == 1) {

result = (result \* a) % mod;

}

a = (a \* a) % mod;

b /= 2;

}

return result;

}

int superPow(int a, vector<int>& b) { const int mod = 1337;

int result = 1;

for (int digit : b) {

result = modPow(result, 10, mod) \* modPow(a, digit, mod) % mod;

}

return result;

}

};

