Assignment-4

Advanced Programming Lab

SARTHAK KUMAR-22BCS12027

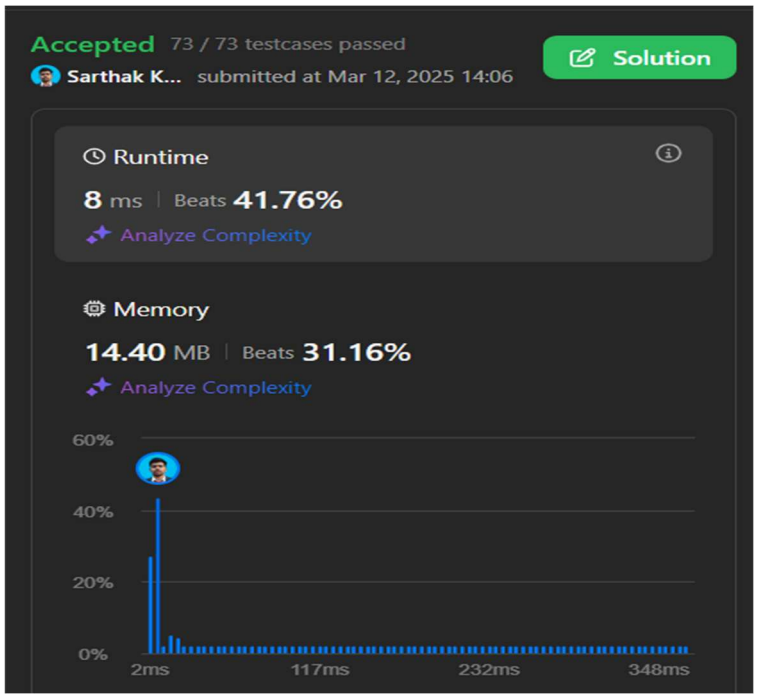
Question 1. A string s is **nice** if, for every letter of the alphabet that s contains, it appears **both** in uppercase and lowercase. For example, "abABB" is nice because 'A' and 'a' appear, and 'B' and 'b' appear. However, "abA" is not because 'b' appears, but 'B' does not. (LONGEST NICE SUBSTRING)

Solution 1. Code Snippet:

C++ Auto

```cpp
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;
    }
};
```

OUTPUT:



Question 2. Reverse bits of a given 32 bits unsigned integer.(REVERSE BITS)

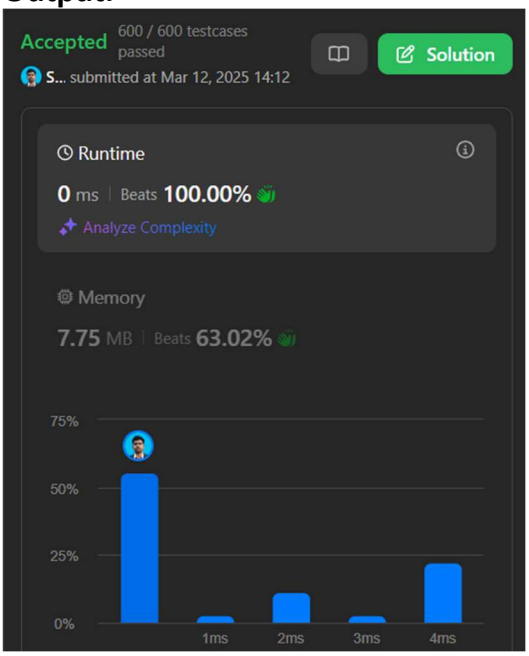
Solution 2.

Code Snippet:

C++ Auto

```cpp
class Solution {
public:
  uint32_t reverseBits(uint32_t n) {
    uint32_t ans = 0;

    for (int i = 0; i < 32; ++i)
      if (n >> i & 1)
        ans |= 1 << 31 - i;

    return ans;
  }
};
```

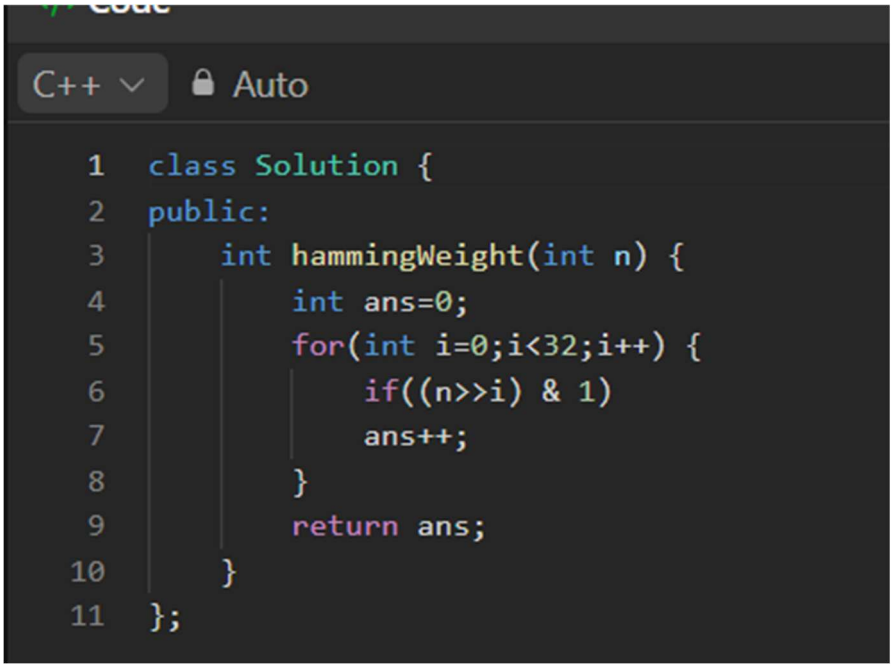
Output:



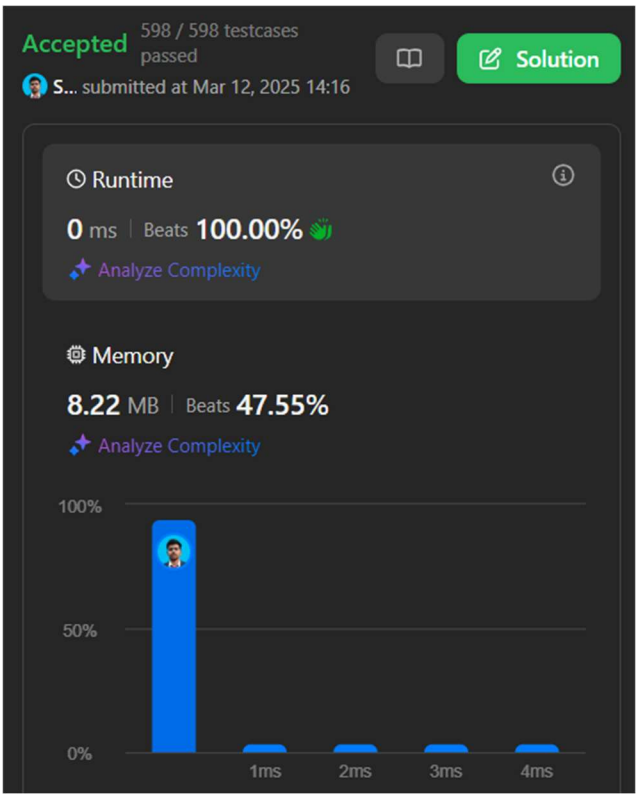
Question 3. Given a positive integer n, write a function that returns the number of set bits in its binary representation (also known as the Hamming weight).(NUMBER OF 1BITs)

Solution 3.

Code Snippet



Output:



Question 4. Given an integer array nums, find the subarray with the largest sum, and return *its sum*.(MAXIMUM SUBARRAY)

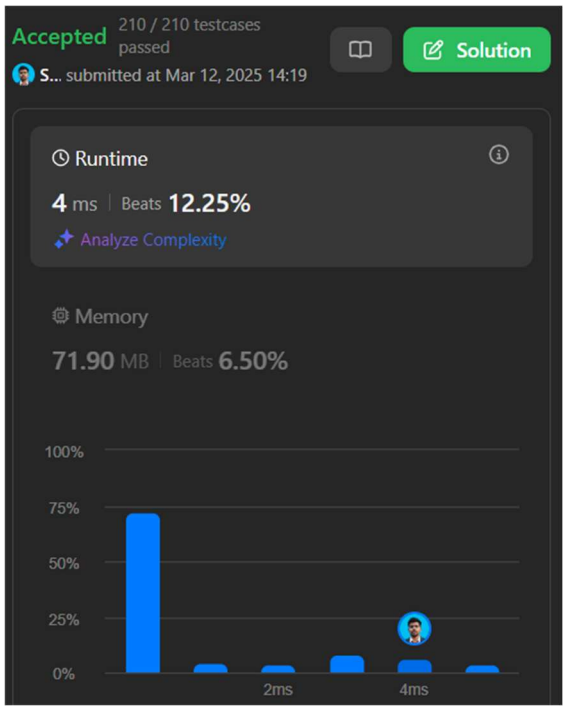
Solution 4.

Code Snippet:

C++ Auto

```cpp
class Solution {
public:
 int maxSubArray(vector<int>& nums) {
   int ans = INT_MIN;
   int sum = 0;

   for (const int num : nums) {
     sum = max(num, sum + num);
     ans = max(ans, sum);
   }

   return ans;
 }
};
```

Output:



Question 5. Write an efficient algorithm that searches for a value target in an m x n integer matrix matrix. .(SEARCH A 2D MATRIX II)

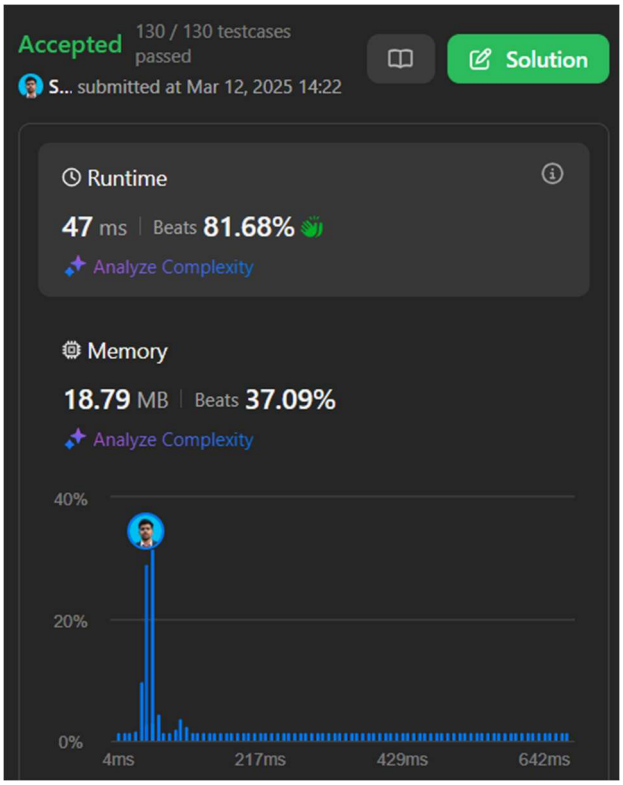
Solution 5:

.

C++ Auto

```cpp
class Solution {
public:
 bool searchMatrix(vector<vector<int>>& matrix, int target) {
   int r = 0;
   int c = matrix[0].size() - 1;

   while (r < matrix.size() && c >= 0) {
     if (matrix[r][c] == target)
       return true;
     if (matrix[r][c] > target)
       --c;
     else
       ++r;
   }

   return false;
 }
};
```

Output:



Question 6. Your task is to calculate a^b mod 1337 where a is a positive integer and b is an extremely large positive integer given in the form of an array.(SUPER POW)

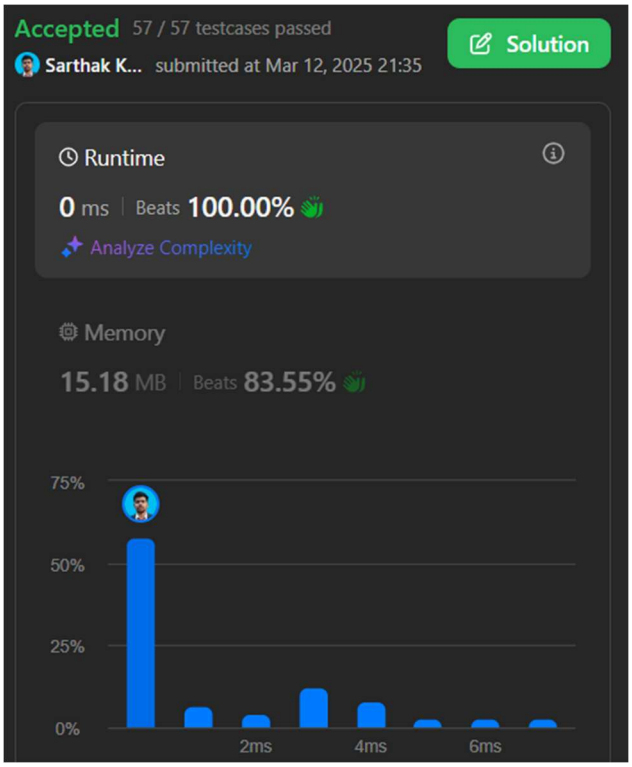
Solution 6.

Code Snippet:

C++ Auto

```cpp
class Solution {
public:
 int superPow(int a, vector<int>& b) {
   int ans = 1;
   a %= kMod;
   for (const int i : b)
     ans = modPow(ans, 10) * modPow(a, i) % kMod;
   return ans;
 }
private:
 static constexpr int kMod = 1337;
 long modPow(long x, long n) {
   if (n == 0)
     return 1;
   if (n % 2 == 1)
     return x * modPow(x % kMod, (n - 1)) % kMod;
   return modPow(x * x % kMod, (n / 2)) % kMod;
 }
};
```

Output:



Question 7.Beautiful Array.

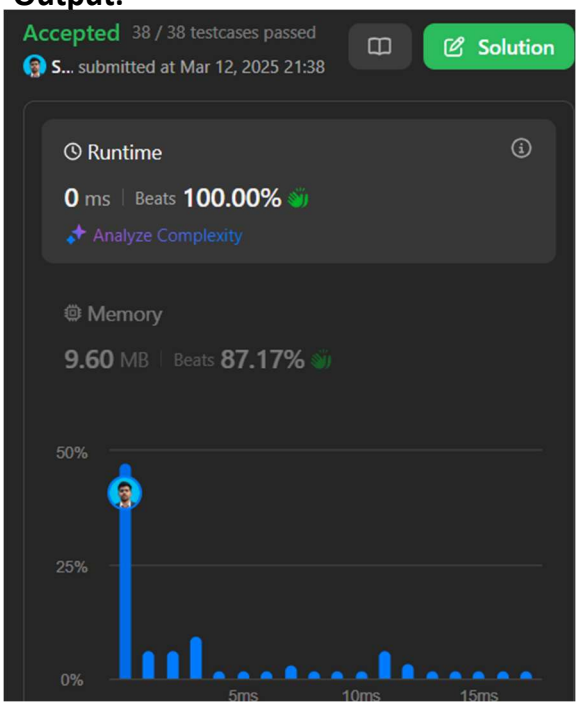
Solution 7.

Code Snippet

C++ Auto

```cpp
class Solution {
public:
  vector<int> beautifulArray(int n) {
    vector<int> arr(n);
    iota(arr.begin(), arr.end(), 1);
    divide(arr, 0, n - 1, 1);
    return arr;
  }
private:
  void divide(vector<int>& arr, int l, int r, int mask) {
    if (l >= r)
      return;
    const int m = partition(arr, l, r, mask);
    divide(arr, l, m, mask << 1);
    divide(arr, m + 1, r, mask << 1);
  }
  int partition(vector<int>& arr, int l, int r, int mask) {
    int nextSwapped = l;
    for (int i = l; i <= r; ++i)
      if (arr[i] & mask)
        swap(arr[i], arr[nextSwapped++]);
    return nextSwapped - 1;
  }
};
```

Output:



Question 8. A city's **skyline** is the outer contour of the silhouette formed by all the buildings in that city when viewed from a distance. Given the locations and heights of all the buildings, return *the **skyline** formed by these buildings collectively*..(**The Skyline problem**)

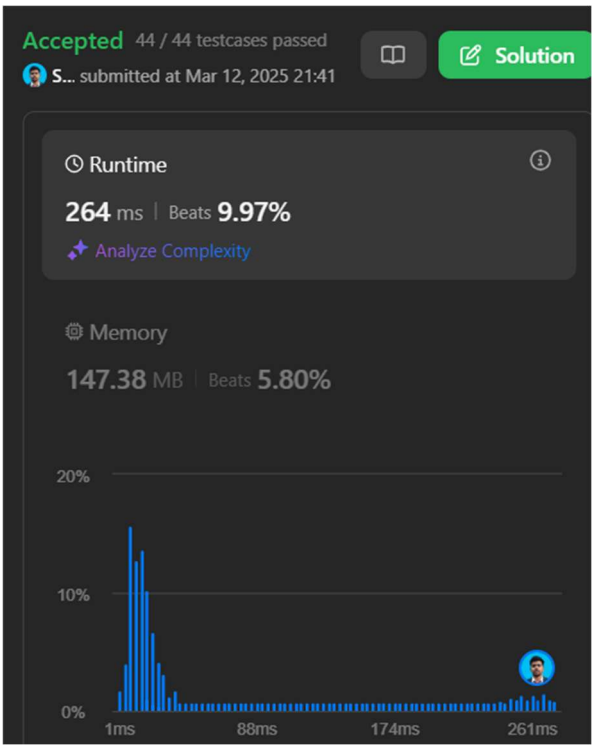
Solution 8.

Code Snippet:

C++ Auto

```cpp
class Solution {
public:
 vector<vector<int>> getSkyline(const vector<vector<int>>& buildings) {
   const int n = buildings.size();
   if (n == 0)
     return {};
   if (n == 1) {
     const int left = buildings[0][0];
     const int right = buildings[0][1];
     const int height = buildings[0][2];
     return {{left, height}, {right, 0}};
   }

   const vector<vector<int>> left =
       getSkyline({buildings.begin(), buildings.begin() + n / 2});
   const vector<vector<int>> right =
       getSkyline({buildings.begin() + n / 2, buildings.end()});
   return merge(left, right);
 }

private:
 vector<vector<int>> merge(const vector<vector<int>>& left,
                           const vector<vector<int>>& right) {
   vector<vector<int>> ans;
   int i = 0;  // left's index
   int j = 0;  // right's index
```

Output:



Question 9. Given an integer array nums, return *the number of* ***reverse pairs*** *in the array*.(**Reverse Pairs***)*

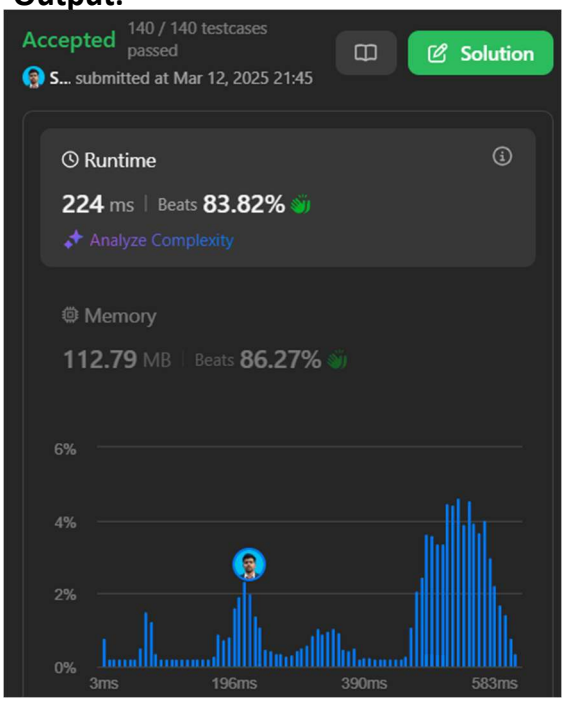
Solution 9.

Code Snippet:

C++ Auto

```cpp
class Solution {
 public:
  int reversePairs(vector<int>& nums) {
    int ans = 0;
    mergeSort(nums, 0, nums.size() - 1, ans);
    return ans;
  }

 private:
  void mergeSort(vector<int>& nums, int l, int r, int& ans) {
    if (l >= r)
      return;

    const int m = (l + r) / 2;
    mergeSort(nums, l, m, ans);
    mergeSort(nums, m + 1, r, ans);
    merge(nums, l, m, r, ans);
  }

  void merge(vector<int>& nums, int l, int m, int r, int& ans) {
    const int lo = m + 1;
    int hi = m + 1;  // the first index s.t. nums[i] <= 2 * nums[hi]

    // For each index i in range [l, m], add hi - lo to `ans`.
    for (int i = l; i <= m; ++i) {
      while (hi <= r && nums[i] > 2L * nums[hi])
```

Output:



Question 10. Longest Increasing Subsequence II

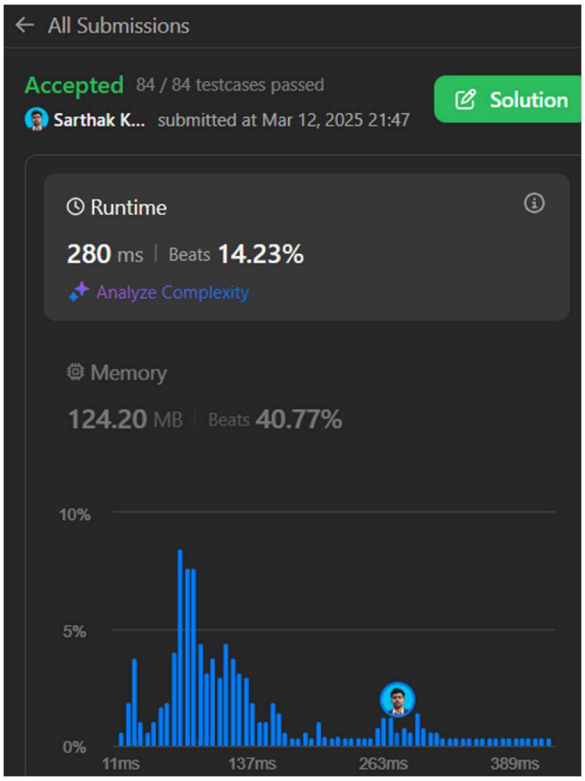
Solution 10.

Code Snippet:

C++ Auto

```cpp
struct SegmentTreeNode {
  int lo;
  int hi;
  int maxLength;
  std::unique_ptr<SegmentTreeNode> left;
  std::unique_ptr<SegmentTreeNode> right;
  // maxLength := the maximum length of LIS ending in [lo..hi]
  SegmentTreeNode(int lo, int hi, int maxLength,
                  std::unique_ptr<SegmentTreeNode> left = nullptr,
                  std::unique_ptr<SegmentTreeNode> right = nullptr)
      : lo(lo),
        hi(hi),
        maxLength(maxLength),
        left(std::move(left)),
        right(std::move(right)) {}
};

class SegmentTree {
 public:
  explicit SegmentTree() : root(make_unique<SegmentTreeNode>(0, 1e5 + 1, 0)) {}

  void updateRange(int i, int j, int maxLength) {
```

Output:


All Submissions
Accepted 84 / 84 testcases passed
Sarthak K... submitted at Mar 12, 2025 21:47
Solution
Runtime
280 ms | Beats 14.23%
Analyze Complexity
Memory
124.20 MB | Beats 40.77%
10%
5%
0%
11ms
137ms
263ms
389ms