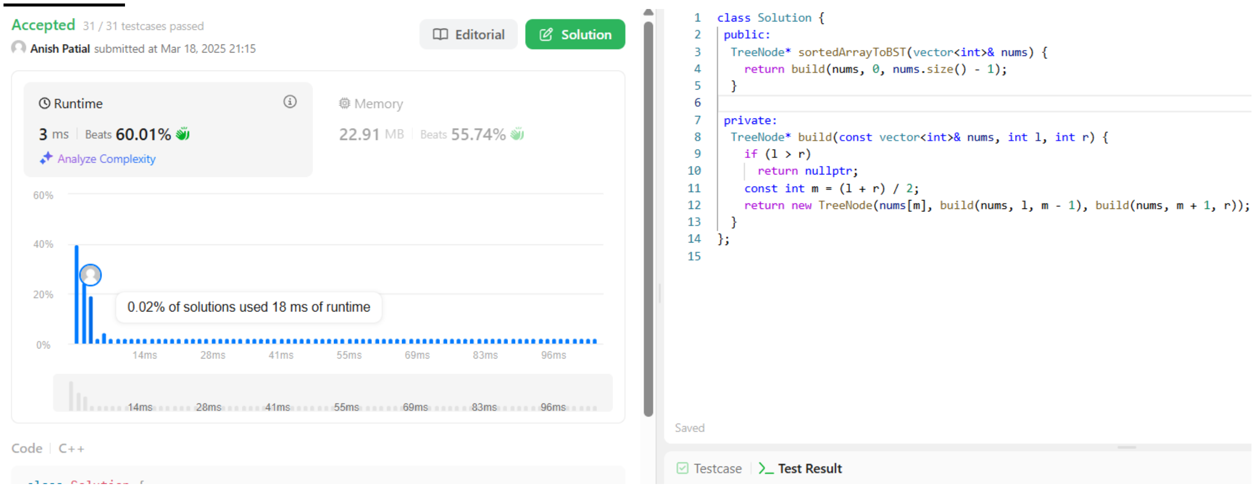
Name: Anish Patial
Uid : 22BCS15029
section: FL_IOT-601/A

1.Convert Sorted Array to Binary Search Tree

```
class Solution {
 public:
  TreeNode* sortedArrayToBST(vector<int>& nums) {
    return build(nums, 0, nums.size() - 1);
  }

 private:
  TreeNode* build(const vector<int>& nums, int l, int r) {
    if (l > r)
      return nullptr;
    const int m = (l + r) / 2;
    return new TreeNode(nums[m], build(nums, l, m - 1), build(nums, m + 1, r));
  }
};
```

RESULT:



Accepted 31 / 31 testcases passed

Anish Patial submitted at Mar 18, 2025 21:15

Runtime

3 ms | Beats 60.01%

Memory

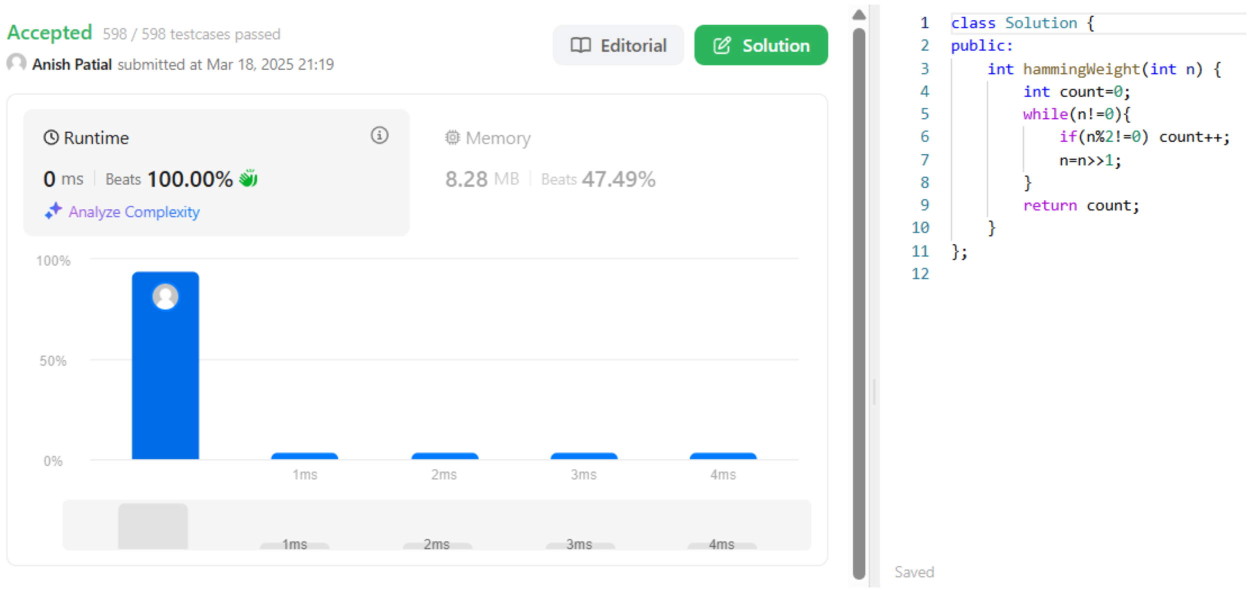
22.91 MB | Beats 55.74%

0.02% of solutions used 18 ms of runtime

```
class Solution {
 public:
  TreeNode* sortedArrayToBST(vector<int>& nums) {
    return build(nums, 0, nums.size() - 1);
  }

 private:
  TreeNode* build(const vector<int>& nums, int l, int r) {
    if (l > r)
      return nullptr;
    const int m = (l + r) / 2;
    return new TreeNode(nums[m], build(nums, l, m - 1), build(nums, m + 1, r));
  }
};
```

2. Number of 1 Bits

```
class Solution {
public:
   int hammingWeight(int n) {
        int count=0;
        while(n!=0){
           if(n%2!=0) count++;
           n=n>>1;
        }
        return count;
     }
};
```



Accepted 598 / 598 testcases passed

Anish Patial submitted at Mar 18, 2025 21:19

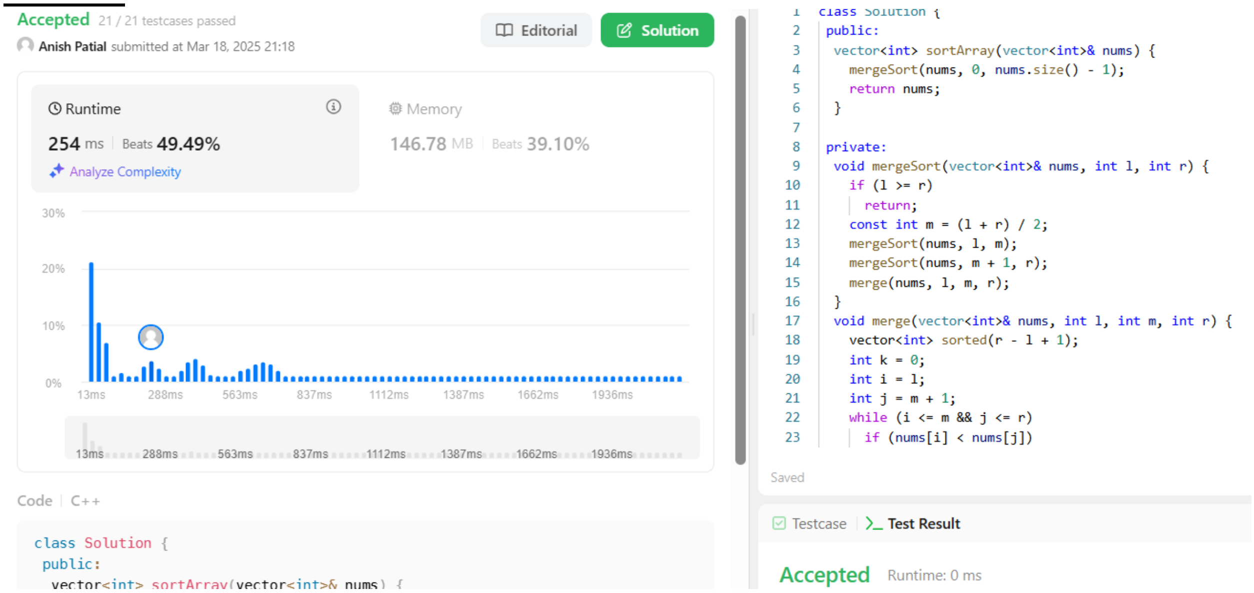
```
class Solution {
public:
    int hammingWeight(int n) {
        int count=0;
        while(n!=0){
            if(n%2!=0) count++;
            n=n>>1;
        }
        return count;
    }
};
```

3. [Sort an Array]

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

 private:
  void mergeSort(vector<int>& nums, int l, int r) {
    if (l >= r)
      return;
    const int m = (l + r) / 2;
```

```
   mergeSort(nums, l, m);
   mergeSort(nums, m + 1, r);
   merge(nums, l, m, r);
  }
  void merge(vector<int>& nums, int l, int m, int r) {
   vector<int> sorted(r - l + 1);
   int k = 0;
   int i = l;
   int j = m + 1;
   while (i <= m && j <= r)
    if (nums[i] < nums[j])
     sorted[k++] = nums[i++];
    else
     sorted[k++] = nums[j++];
   while (i <= m)
    sorted[k++] = nums[i++];
   while (j <= r)
    sorted[k++] = nums[j++];
   copy(sorted.begin(), sorted.end(), nums.begin() + l);
  }
};
```

RESULT:



Accepted 21 / 21 testcases passed

Anish Patial submitted at Mar 18, 2025 21:18

254 ms | Beats 49.49%

146.78 MB | Beats 39.10%

```
class Solution {
 public:
  vector<int> sortArray(vector<int>& nums) {
```

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

 private:
  void mergeSort(vector<int>& nums, int l, int r) {
    if (l >= r)
      return;
    const int m = (l + r) / 2;
    mergeSort(nums, l, m);
    mergeSort(nums, m + 1, r);
    merge(nums, l, m, r);
  }
  void merge(vector<int>& nums, int l, int m, int r) {
    vector<int> sorted(r - l + 1);
    int k = 0;
    int i = l;
    int j = m + 1;
    while (i <= m && j <= r)
      if (nums[i] < nums[j])
```

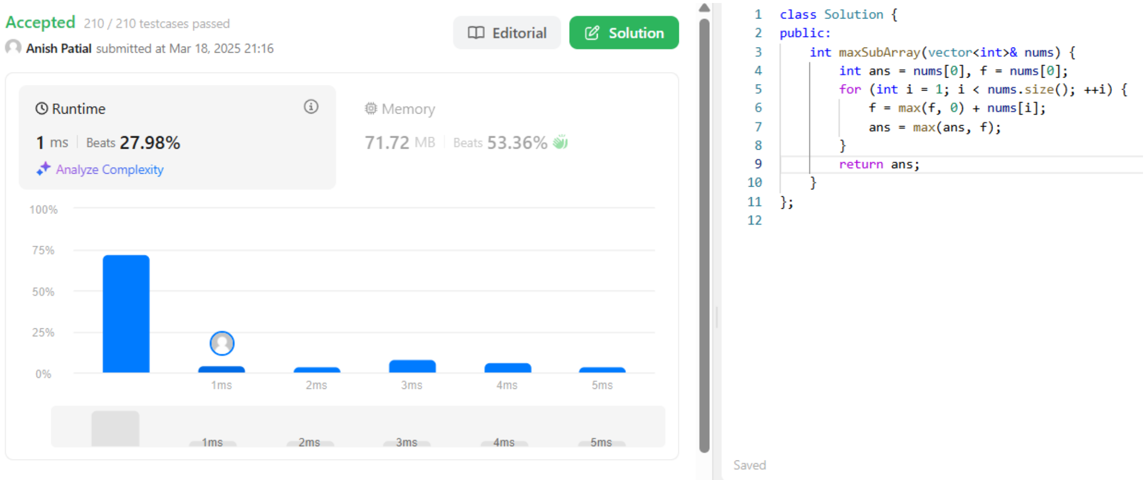
Saved

Accepted Runtime: 0 ms

4. 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;
    }
};
```

RESULT:



Accepted 210 / 210 testcases passed

Anish Patial submitted at Mar 18, 2025 21:16

Runtime: 1 ms | Beats 27.98%

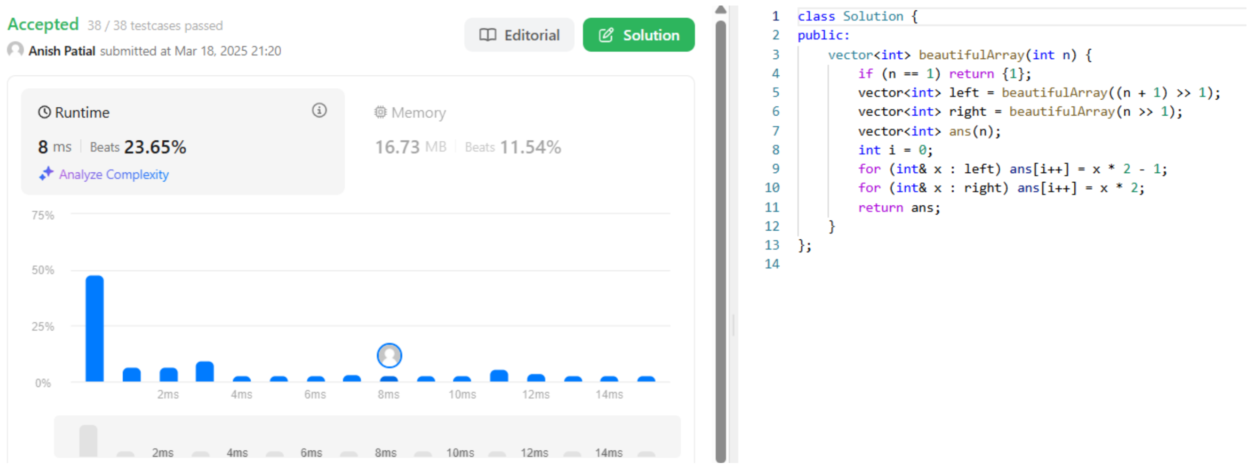
Memory: 71.72 MB | Beats 53.36%

```
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. Beautiful Array

```
class Solution {
public:
    vector<int> beautifulArray(int n) {
        if (n == 1) return {1};
        vector<int> left = beautifulArray((n + 1) >> 1);
        vector<int> right = beautifulArray(n >> 1);
        vector<int> ans(n);
        int i = 0;
        for (int& x : left) ans[i++] = x * 2 - 1;
        for (int& x : right) ans[i++] = x * 2;
        return ans;
    }
};
```

RESULT:



Accepted 38 / 38 testcases passed

Anish Patial submitted at Mar 18, 2025 21:20

Runtime

8 ms | Beats 23.65%

Memory

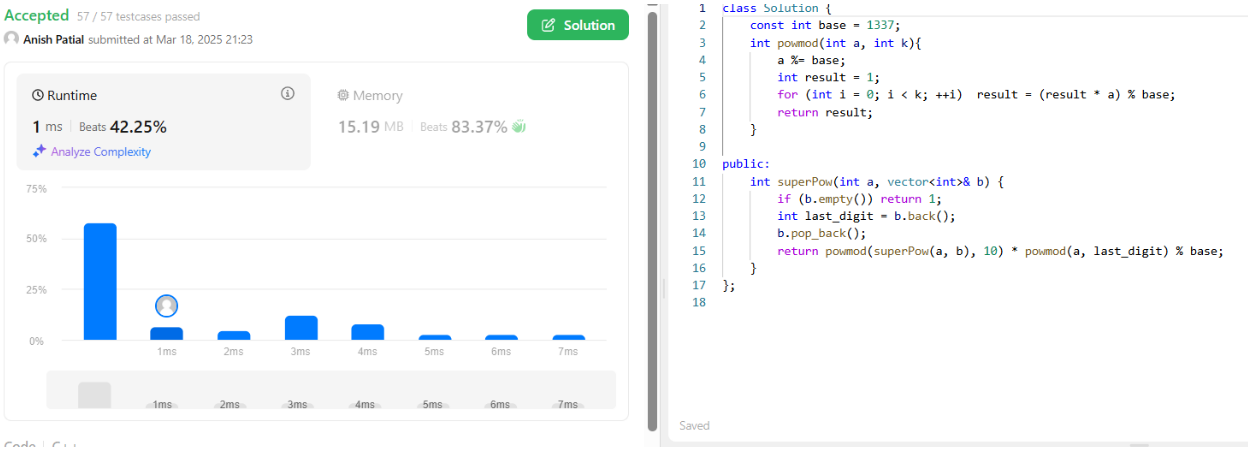
16.73 MB | Beats 11.54%

```
class Solution {
public:
    vector<int> beautifulArray(int n) {
        if (n == 1) return {1};
        vector<int> left = beautifulArray((n + 1) >> 1);
        vector<int> right = beautifulArray(n >> 1);
        vector<int> ans(n);
        int i = 0;
        for (int& x : left) ans[i++] = x * 2 - 1;
        for (int& x : right) ans[i++] = x * 2;
        return ans;
    }
};
```

6. Super 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;
    }
};
```

RESULT:



Accepted 57 / 57 testcases passed

Anish Patial submitted at Mar 18, 2025 21:23

Runtime

1 ms | Beats 42.25%

Memory

15.19 MB | Beats 83.37%

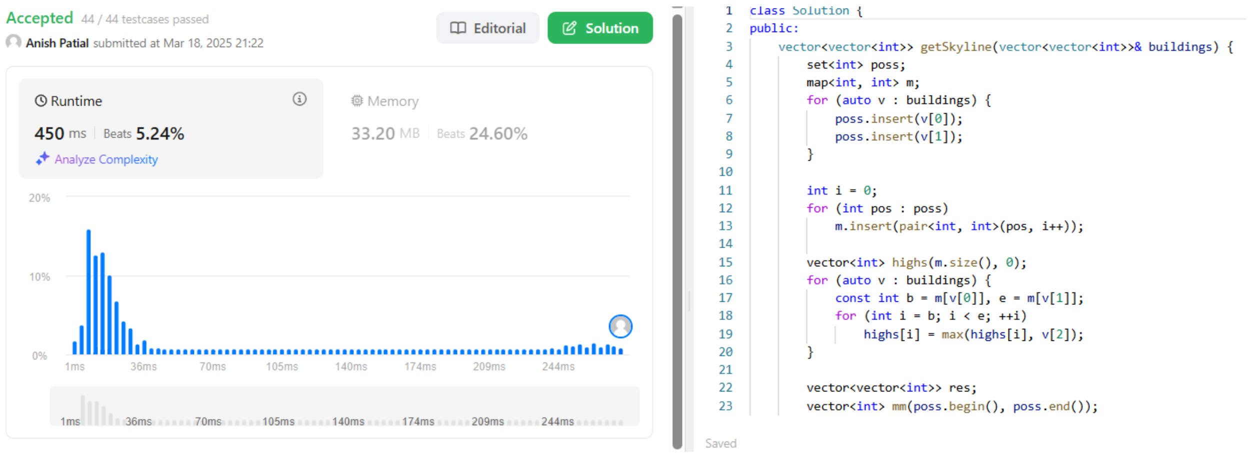
```
class Solution {
    const int base = 1337;
    int powmod(int a, int k){
        a %= base;
        int result = 1;
        for (int i = 0; i < k; ++i)  result = (result * a) % base;
        return result;
    }

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;
    }
};
```

7. The Skyline Problem

```
class Solution {
public:
   vector<vector<int>> getSkyline(vector<vector<int>>& buildings) {
      set<int> poss;
      map<int, int> m;
      for (auto v : buildings) {
         poss.insert(v[0]);
         poss.insert(v[1]);
      }

      int i = 0;
      for (int pos : poss)
         m.insert(pair<int, int>(pos, i++));

      vector<int> highs(m.size(), 0);
      for (auto v : buildings) {
         const int b = m[v[0]], e = m[v[1]];
         for (int i = b; i < e; ++i)
            highs[i] = max(highs[i], v[2]);
      }

      vector<vector<int>> res;
```

```
    vector<int> mm(poss.begin(), poss.end());
    for (int i = 0; i < highs.size(); i++) {
      if (i+1 < highs.size() && highs[i] != highs[i + 1])
        res.push_back({mm[i], highs[i]});
      else {
        const int start = i;
        res.push_back({mm[start], highs[i]});
        while (i+1 < highs.size() && highs[i] == highs[i + 1])
          ++i;
      }
    };
    return res;
  }
};
```

RESULT :



Accepted 44 / 44 testcases passed

Anish Patial submitted at Mar 18, 2025 21:22

Runtime 450 ms | Beats 5.24%

Memory 33.20 MB | Beats 24.60%

```
class Solution {
public:
    vector<vector<int>> getSkyline(vector<vector<int>>& buildings) {
        set<int> poss;
        map<int, int> m;
        for (auto v : buildings) {
            poss.insert(v[0]);
            poss.insert(v[1]);
        }

        int i = 0;
        for (int pos : poss)
            m.insert(pair<int, int>(pos, i++));

        vector<int> highs(m.size(), 0);
        for (auto v : buildings) {
            const int b = m[v[0]], e = m[v[1]];
            for (int i = b; i < e; ++i)
                highs[i] = max(highs[i], v[2]);
        }

        vector<vector<int>> res;
        vector<int> mm(poss.begin(), poss.end());
```