

Name: Krishna Sharma

UID: 22BCS11885

Section: FL_IOT-602 / A

Convert Sorted Array to Binary Search Tree

```
class Solution {
public:
    TreeNode* helper(int left, int right, vector<int>& nums) {
        if(left > right) return nullptr;
        int mid = left+(right-left)/2;
        TreeNode* root = new TreeNode(nums[mid]);
        root->left = helper(left, mid-1, nums);
        root->right = helper(mid+1, right, nums);
        return root;
    }
    TreeNode* sortedArrayToBST(vector<int>& nums) {
        int n = nums.size()-1;
        return helper(0, n, nums);
    }
};
```

The screenshot shows the LeetCode interface for problem 108. On the left, the problem description states: "Given an integer array `nums` where the elements are sorted in **ascending order**, convert it to a **height-balanced** binary search tree." An example is provided with input `nums = [-10, -3, 0, 5, 9]` and output `[0, -3, 9, -10, null, 5]`. A diagram shows a balanced BST with root 0, left child -3, right child 9, and further children -10 and 5. The right panel shows the submission results: "Accepted 31 / 31 testcases passed", "Runtime 9 ms | Beats 8.74%", and "Memory 21.90 MB | Beats 99.65%". A bar chart shows the distribution of runtime results. The code editor shows a C++ solution using a recursive helper function.

108. Convert Sorted Array to Binary Search Tree Solved

Easy Topics Companies

Given an integer array `nums` where the elements are sorted in **ascending order**, convert it to a **height-balanced** binary search tree.

Example 1:

Input: `nums = [-10, -3, 0, 5, 9]`
Output: `[0, -3, 9, -10, null, 5]`
Explanation: `[0, -10, 5, null, -3, null, 9]` is also accepted:

Runtime: 9 ms | Beats: 8.74%
Memory: 21.90 MB | Beats: 99.65%

Code: C++

```
/**
 * Definition for a binary tree node.
 */
```

Number of 1 Bits

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

The screenshot shows the LeetCode interface for the problem '191. Number of 1 Bits'. The problem is marked as 'Solved' and 'Easy'. The description states: '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*).' Three examples are provided: Example 1 (Input: n = 11, Output: 3), Example 2 (Input: n = 128, Output: 1), and Example 3 (Input: n = 2147483645). The right panel shows the submission status as 'Accepted' with 596/596 testcases passed. Performance metrics are displayed: Runtime 0 ms (Beats 100.00%) and Memory 7.30 MB (Beats 100.00%). A bar chart shows the runtime performance relative to other submissions. The code editor shows the C++ solution for the hammingWeight function.

Sort an Array

```
class Solution {
public:
    void merge(vector<int>& A, int start, int mid, int end, vector<int>& buff) {
        int left=start, right=mid+1;
        int s=end-start+1;
        for(int i=0; i<s; i++) {
            int i0=start+i;
            if(left>mid){
                buff[i0]=A[right];
                right++;
            } else if (right>end){
```

```

        buff[i0]=A[left];
        left++;
    } else if (A[left]<A[right]) {
        buff[i0]=A[left];
        left++;
    } else {
        buff[i0]=A[right];
        right++;
    }
}
for(int i=start; i<start+s; i++) A[i]=buff[i];
}

void mergeSort(vector<int>& A, int start, int end, vector<int>& buff ) {
    if(end<=start) return;
    int mid=start+(end-start)/2;
    mergeSort(A, start, mid, buff);
    mergeSort(A, mid+1, end, buff);
    merge(A, start, mid, end, buff);
}

vector<int> sortArray(vector<int>& nums) {
    vector<int> buff(nums.size());
    mergeSort(nums, 0, nums.size()-1 ,buff);
    return nums;
}
};

```

The screenshot displays a coding platform interface with two main panels. The left panel shows the problem description for '912. Sort an Array', which is a Medium-level problem. It requires sorting an array of integers in ascending order without using built-in functions, with a time complexity of $O(n \log(n))$ and the smallest space complexity possible. Two examples are provided: Example 1 with input [5,2,3,1] and output [1,2,3,5], and Example 2 with input [5,1,1,2,0,0] and output [0,0,1,1,2,5]. The constraints specify that the array length is between 1 and 5×10^4 . The right panel shows the submission result for a user named Krishna Sharma, who submitted the solution on July 25, 2024. The submission is marked as 'Accepted' and passed 21 out of 21 test cases. The runtime is 123 ms, which beats 59.49% of other submissions, and the memory usage is 71.50 MB, which beats 67.93%. A bar chart shows the distribution of runtime times across various test cases, with the highest frequency at 13ms. The code is written in C++ and defines a class Solution with a public method.

912. Sort an Array Solved

Medium Topics Companies

Given an array of integers `nums`, sort the array in ascending order and return it.

You must solve the problem **without using any built-in** functions in $O(n \log(n))$ time complexity and with the smallest space complexity possible.

Example 1:

Input: `nums = [5,2,3,1]`
Output: `[1,2,3,5]`
Explanation: After sorting the array, the positions of some numbers are not changed (for example, 2 and 3), while the positions of other numbers are changed (for example, 1 and 5).

Example 2:

Input: `nums = [5,1,1,2,0,0]`
Output: `[0,0,1,1,2,5]`
Explanation: Note that the values of `nums` are not necessarily unique.

Constraints:

- $1 \leq \text{nums.length} \leq 5 \times 10^4$

6.7K 329 79 Online

Accepted 21 / 21 testcases passed
 Krishna Sharma submitted at Jul 25, 2024 22:38

Runtime: 123 ms Beats 59.49%
 Memory: 71.50 MB Beats 67.93%

Analyze Complexity

Code | C++

```
class Solution {
public:
```

Testcase Test Result

Case 1 Case 2

Source

Maximum Subarray

```
class Solution {
public:
    int maxSubArray(vector<int>& nums) {
        int ans = INT_MIN;
        int currSum = 0;
        for(int i=0;i<nums.size();i++) {
            currSum += nums[i];
            if(currSum > ans) ans = currSum;
            if(currSum < 0) currSum = 0;
        }
        return ans;
    }
};
```

53. Maximum Subarray Solved

Medium Topics Companies

Given an integer array `nums`, find the **subarray** with the largest sum, and return its sum.

Example 1:
Input: `nums = [-2,1,-3,4,-1,2,1,-5,4]`
Output: 6
Explanation: The subarray `[4,-1,2,1]` has the largest sum 6.

Example 2:
Input: `nums = [1]`
Output: 1
Explanation: The subarray `[1]` has the largest sum 1.

Example 3:
Input: `nums = [5,4,-1,7,8]`
Output: 23
Explanation: The subarray `[5,4,-1,7,8]` has the largest sum 23.

Constraints:

- 1 <= nums.length <= 2 * 10⁴
- 10 <= nums[i] <= 10

35.4K 342 473 Online

Accepted 210 / 210 testcases passed
Krishna Sharma submitted at Aug 10, 2024 21:33

Runtime: 72 ms | Beats: 5.38%
Memory: 70.58 MB | Beats: 99.98%

Code C++

```
class Solution {
public:
```

Testcase Test Result

Case 1 Case 2 Case 3 +

</> Source

Beautiful Array

```
class Solution {
public:
    int partition(vector<int> &v, int start, int end, int mask) {
        int j = start;
        for(int i = start; i <= end; i++) {
            if((v[i] & mask) != 0) {
                swap(v[i], v[j]);
                j++;
            }
        }
    }
};
```

```

        return j;
    }
    void sort(vector<int> & v, int start, int end, int mask) {
        if(start >= end) return;
        int mid = partition(v, start, end, mask);
        sort(v, start, mid - 1, mask << 1);
        sort(v, mid, end, mask << 1);
    }
    vector<int> beautifulArray(int N) {
        vector<int> ans;
        for(int i = 0; i < N; i++) ans.push_back(i + 1);
        sort(ans, 0, N - 1, 1);
        return ans;
    }
};

```

The screenshot displays a C++ solution on a coding platform. The left sidebar shows the submission status: 'Accepted' with 38/38 testcases passed, submitted at Mar 18, 2025 20:55. The 'Runtime' section indicates 0 ms and 100.00% efficiency. The 'Memory' section shows 9.43 MB and 93.30% efficiency. A bar chart below these metrics shows the runtime distribution across testcases. The main code editor displays the following C++ code:

```

1  class Solution {
2  public:
3      int partition(vector<int> &v, int start, int end, int mask) {
4          int j = start;
5          for(int i = start; i <= end; i++) {
6              if((v[i] & mask) != 0) {
7                  swap(v[i], v[j]);
8                  j++;
9              }
10         }
11         return j;
12     }
13     void sort(vector<int> &v, int start, int end, int mask) {
14         if(start >= end) return;
15         int mid = partition(v, start, end, mask);
16         sort(v, start, mid - 1, mask << 1);
17         sort(v, mid, end, mask << 1);
18     }
19     vector<int> beautifulArray(int N) {
20         vector<int> ans;
21         for(int i = 0; i < N; i++) ans.push_back(i + 1);
22         sort(ans, 0, N - 1, 1);
23         return ans;
24     }
25 };

```

The bottom of the interface shows a 'Testcase' tab with 'Case 1' and 'Case 2' selected, and a 'Test Result' section.

Super Pow

```

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

```

```

int superPow(int a, vector<int>& b) {
    if(b.empty()) return 1;
    int last_digit = b.back();
    b.pop_back();
    return solve(superPow(a, b), 10) * solve(a, last_digit) % base;
}
};

```

The Skyline Problem

```

class Solution {
public:
    vector<vector<int>> getSkyline(vector<vector<int>>& buildings) {
        multiset<int> height;
        priority_queue<pair<int, int>, vector<pair<int, int>>, greater<pair<int, int>>> pq;
        vector<vector<int>> ans;
        set<int> st;
        for(auto& building : buildings) {
            st.insert(building[0]);
            st.insert(building[1]);
        }
        int j = 0;
        for(auto i: st) {
            while(!pq.empty() && pq.top().first <= i) {
                height.erase(height.find(pq.top().second));
                pq.pop();
            }
            while(j < buildings.size() && buildings[j][0] == i) {
                pq.push({buildings[j][1], buildings[j][2]});
                height.insert(buildings[j][2]);
                j++;
            }
        }
    }
};

```

```

    }
    if(height.empty()) {
        if(ans.empty() || ans.back()[1] != 0)
            ans.push_back({i, 0});
    } else {
        int maxHeight = *height.rbegin();
        if (ans.empty() || ans.back()[1] != maxHeight)
            ans.push_back({i, maxHeight});
    }
}
return ans;
}
};

```

Description

Accepted

Editorial

Solutions

Submit

All Submissions

Accepted 44 / 44 testcases passed

Krishna Sharma submitted at Mar 18, 2025 21:04

Runtime

19 ms | Beats 52.91%

Analyze Complexity

Memory

30.08 MB | Beats 37.98%

Code

C++

Auto

```

1 class Solution {
2 public:
3     vector<vector<int>> getSkyline(vector<vector<int>>& buildings) {
4         multiset<int> height;
5         priority_queue<pair<int, int>, vector<pair<int, int>>, greater<pair<int, int>>> pq;
6         vector<vector<int>> ans;
7         set<int> st;
8         for(auto& building : buildings) {
9             st.insert(building[0]);
10            st.insert(building[1]);
11        }
12        int j = 0;
13        for(auto i: st) {
14            while(!pq.empty() && pq.top().first <= i) {
15                height.erase(height.find(pq.top().second));
16                pq.pop();
17            }
18            while(j < buildings.size() && buildings[j][0] == i) {
19                pq.push({buildings[j][1], buildings[j][2]});

```

Saved

Testcase

Test Result

Source