Assignment – 6

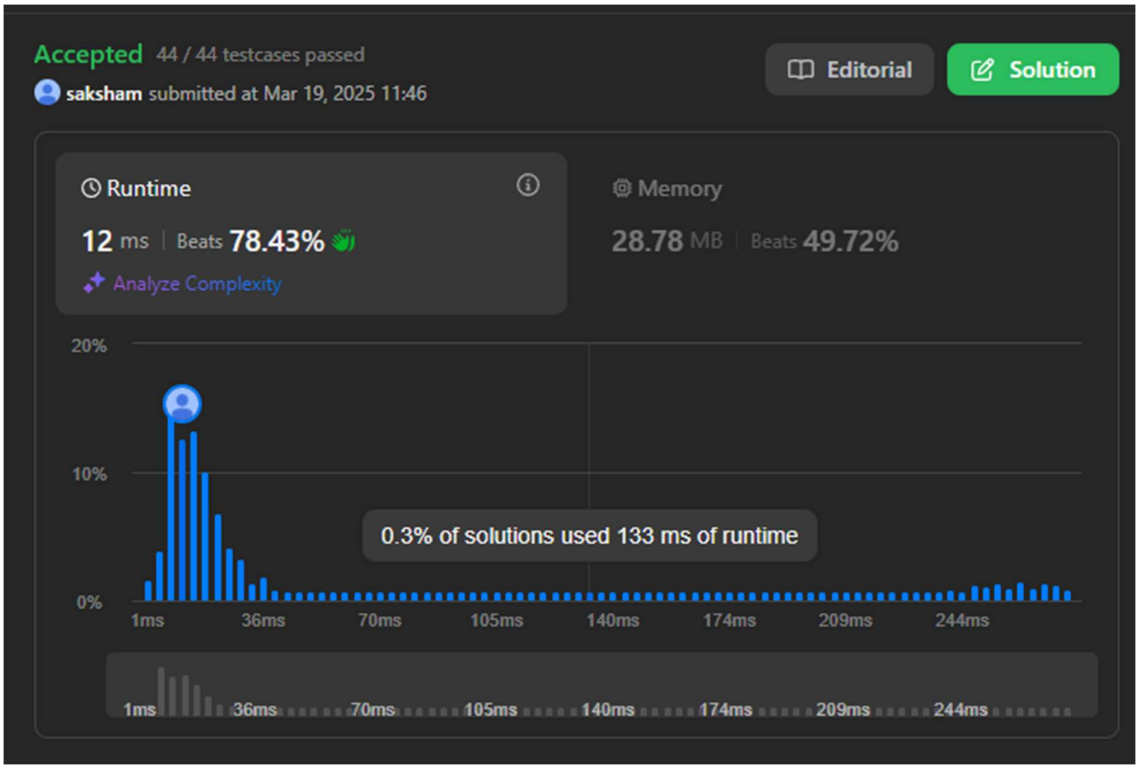
Name – saksham mishra **Uid** – 22bcs15604

1. Skyline problem

https://leetcode.com/problems/the-skyline-problem/

```
class Solution {
public:
  vector<vector<int>> getSkyline(vector<vector<int>>& buildings) {
    vector<vector<int>> ans;
    multiset<int> pq{0};

    vector<pair<int, int>> points;

    for(auto b: buildings){
      points.push_back({b[0], -b[2]});
      points.push_back({b[1], b[2]});
    }

    sort(points.begin(), points.end());

    int ongoingHeight = 0;

    // points.first = x coordinate, points.second = height
    for(int i = 0; i < points.size(); i++){
      int currentPoint = points[i].first;
      int heightAtCurrentPoint = points[i].second;

      if(heightAtCurrentPoint < 0){
        pq.insert(-heightAtCurrentPoint);
      } else {
        pq.erase(pq.find(heightAtCurrentPoint));
      }
```

```
        // after inserting/removing heightAtI, if there's a change
        auto pqTop = *pq.rbegin();
        if(ongoingHeight != pqTop){
          ongoingHeight = pqTop;
          ans.push_back({currentPoint, ongoingHeight});
        }
      }

      return ans;
    }
};
```

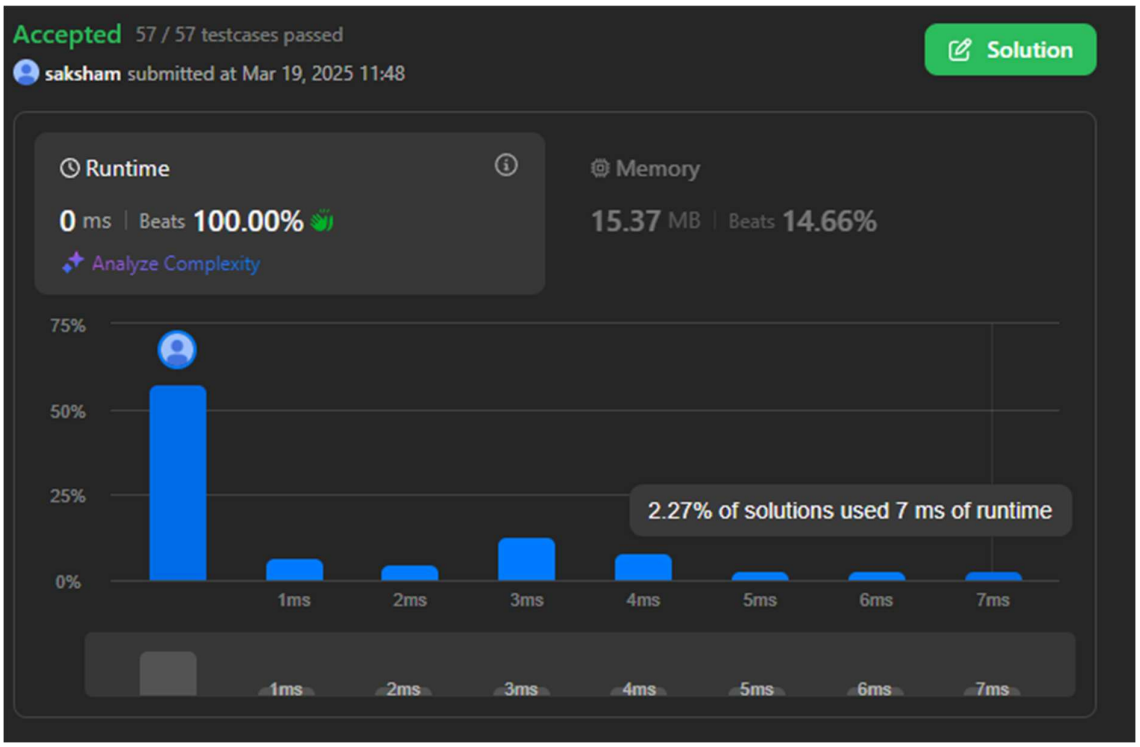


2. Super pow

https://leetcode.com/problems/super-pow/description/

```
class Solution {
  const int base = 1337;
  int powmod(int a, int k) //a^k mod 1337 where 0 <= k <= 10
  {
```

```
        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;
    }
};
```

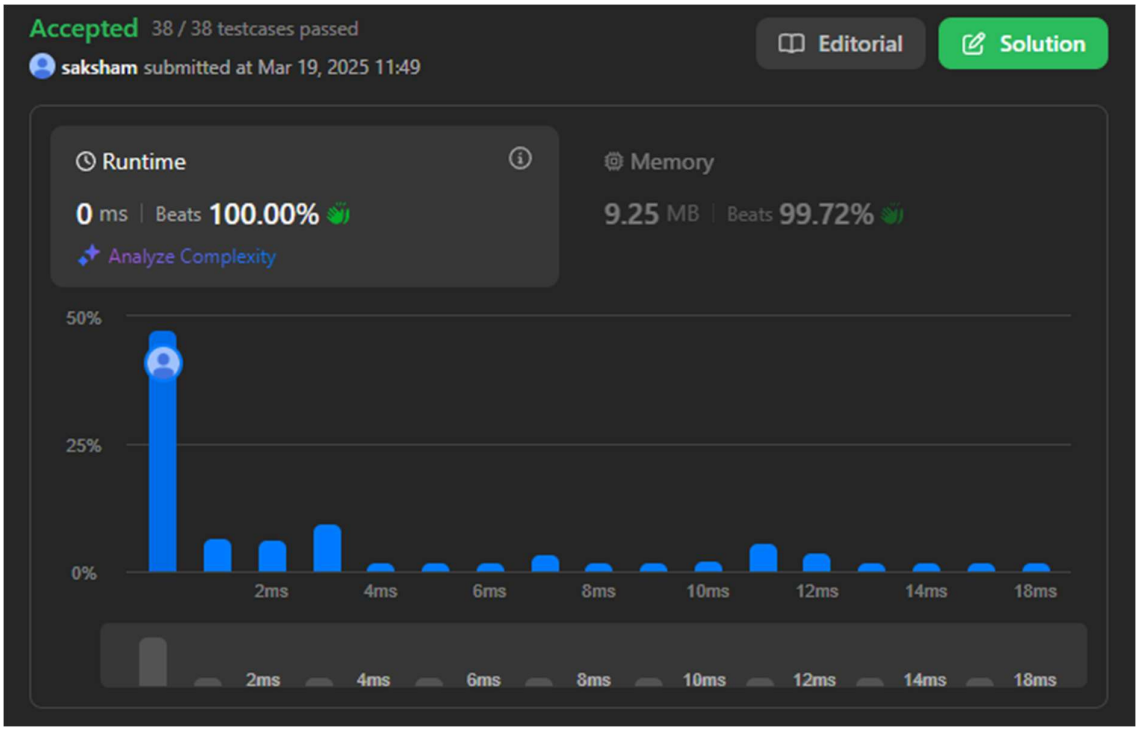


3. Beautiful array

https://leetcode.com/problems/beautiful-array/description/

```
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;
    }
};
```

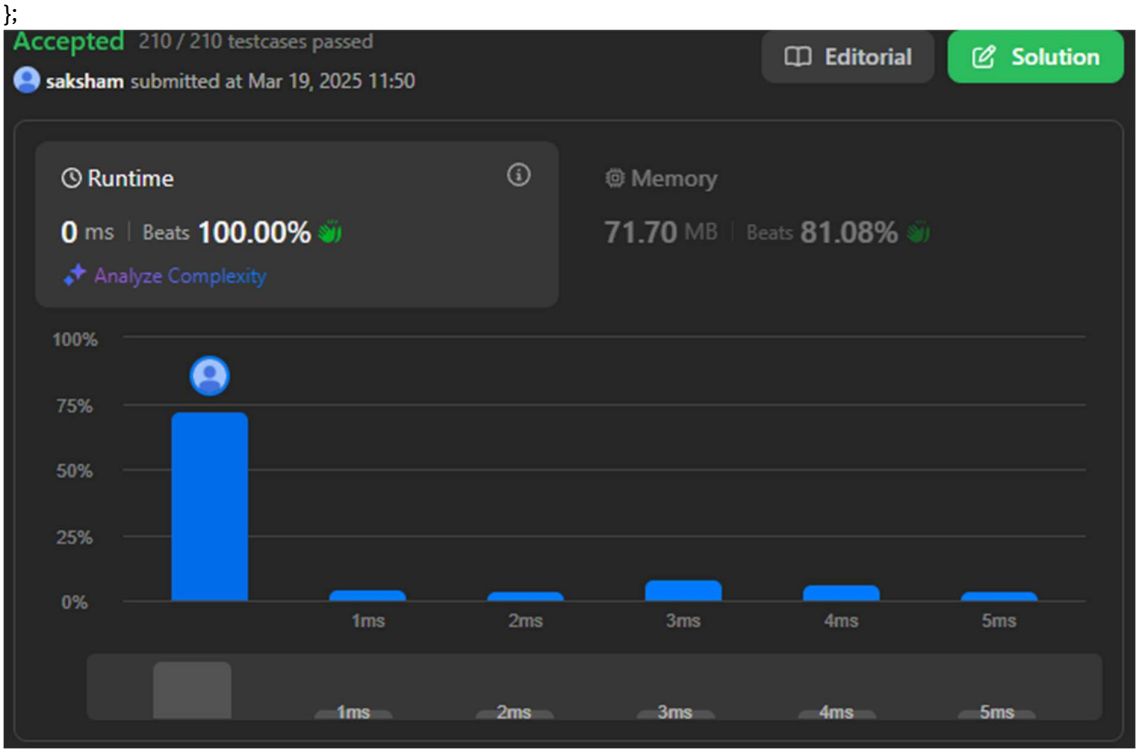


4. Maximum subarray

https://leetcode.com/problems/maximum-subarray/

```
class Solution {
public:
  int maxSubArray(vector<int>& nums) {
    int res = nums[0];
    int total = 0;

    for (int n : nums) {
      if (total < 0) {
        total = 0;
      }

      total += n;
      res = max(res, total);
    }
```

```
        return res;
    }
};
```

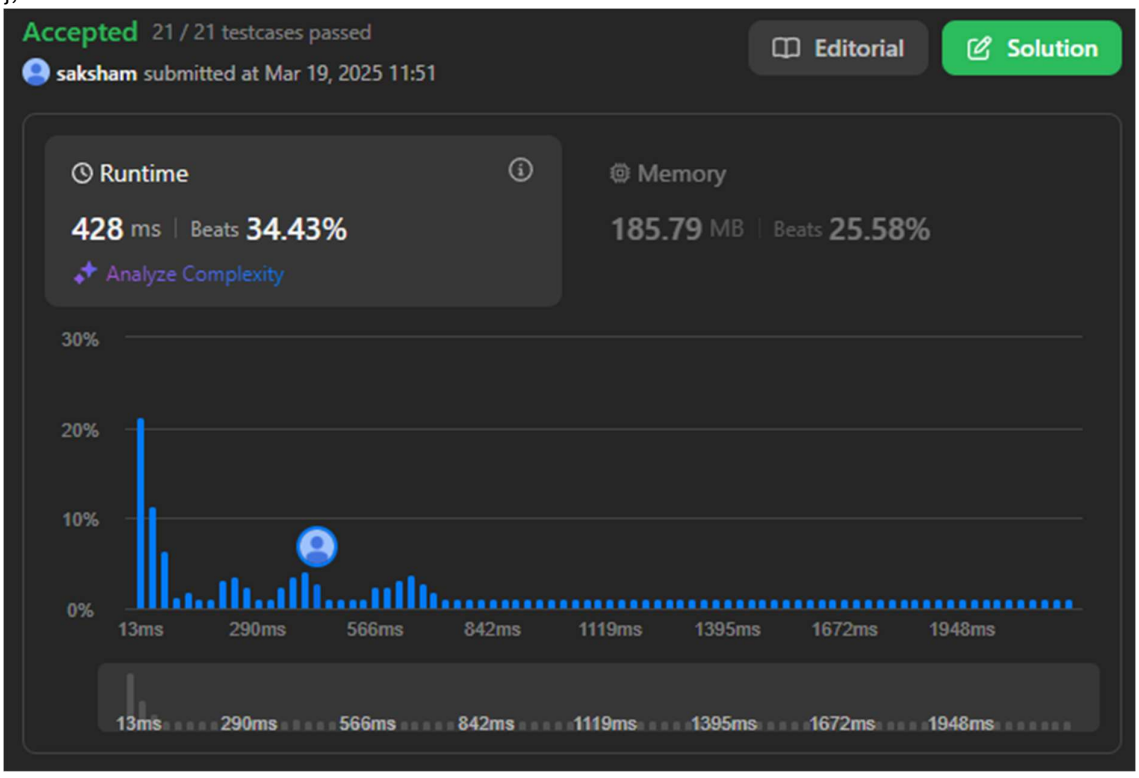


5. Sort an Array

https://leetcode.com/problems/sort-an-array/description/

```
class Solution {
public:
    void merge(vector<int>& nums,int s, int e){
        int m=(s+e)/2;
        vector<int>first(m-s+1),second(e-m);
        for (int i=0;i<first.size();i++){first[i]=nums[s+i];}
        for (int i=0;i<second.size();i++){second[i]=nums[m+1+i];}
        int i1=0,i2=0,maindex=s;
        while (i1<first.size() && i2<second.size()){
            if (first[i1]<second[i2]){nums[maindex++]=first[i1++];}
            else {nums[maindex++]=second[i2++];}
        }
        while (i1<first.size()){nums[maindex++]=first[i1++];}
```

```
    while (i2<second.size()){nums[maindex++]=second[i2++];}
  }
  void mergesort(vector<int>&nums, int s, int e){
    if (s>=e){return ;}
    int m=(s+e)/2;
    mergesort(nums,s,m);
    mergesort(nums,m+1,e);
    merge(nums,s,e);
  }
  vector<int> sortArray(vector<int>& nums) {
    mergesort(nums,0,nums.size()-1);
    return nums;
  }
};
```



6. Convert sorted array to binary search tree

https://leetcode.com/problems/convert-sorted-array-to-binary-search-tree/description/

```
class Solution {
```

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

private:
    TreeNode* helper(vector<int>& nums, int left, int right) {
        if (left > right) return nullptr;
        int mid = left + (right - left) / 2;
        TreeNode* root = new TreeNode(nums[mid]);
        root->left = helper(nums, left, mid - 1);
        root->right = helper(nums, mid + 1, right);
        return root;
    }
};
```