AP Assignment 4

Name: Sneha Thapa

UID: 22BCS15832

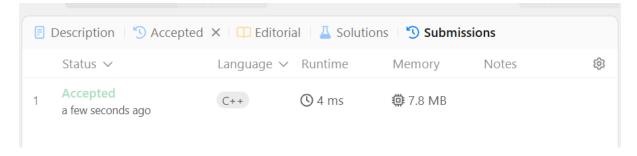
Section: 608-B

1) Longest Nice Substring

```
class Solution {
 public:
      bool isNice(string s) {
          unordered_set<char> st(s.begin(), s.end());
          for (char c : s) {
              if (st.count(toupper(c)) == 0 || st.count(tolower(c)) == 0) {
                   return false;
              }
          }
          return true;
      }
      string longestNiceSubstring(string s) {
          int n = s.size();
          string longest = "";
          for (int i = 0; i < n; i++) {
              for (int j = i; j < n; j++) {</pre>
                   string sub = s.substr(i, j - i + 1);
                   if (isNice(sub) && sub.size() > longest.size()) {
                       longest = sub;
                   }
              }
          }
          return longest;
      }
 };
🗉 Description | 🛄 Editorial | 🚣 Solutions | 🧐 Submissions
    Status V
                          Language ∨ Runtime
                                                   Memory
                                                                 Notes
                                                                                 (ģ)
    Accepted
                                                    ⊕ 128.5 MB
                                      ③ 381 ms
                          C++
                                                                 + Notes
    Mar 06, 2025
    Accepted
                          C++
                                      (S) 377 ms
                                                   128.5 MB
    Feb 27, 2025
```

2) Reverse Bits

```
class Solution {
public:
    uint32_t reverseBits(uint32_t n) {
        uint32_t result = 0;
        for (int i = 0; i < 32; i++) {
            result = (result << 1) | (n & 1);
            n >>= 1;
        }
        return result;
    }
};
```

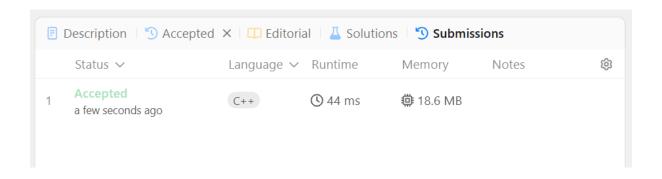


3) Number of 1 Bits

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

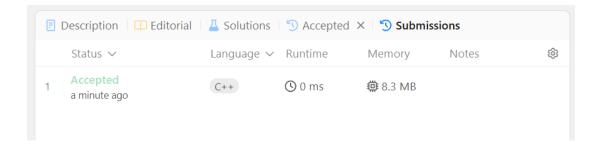
4) Search a 2D Matrix II

```
class Solution {
public:
  bool searchMatrix(vector<vector<int>>& matrix, int target) {
    int r=0,c=matrix[0].size()-1;
    while (r<matrix.size() && c>=0){
        if (matrix[r][c]==target){return true;}
        else if (matrix[r][c]<target){r++;}
        else {c--;}
    }
    return false;
}</pre>
```



5) Super Pow

```
class Solution {
public:
  int getSum(int a, int b) {
      vector<int>out1;
     if(a < 0 \&\& b == 0) return a;
     if(b<0 && a== 0) return b;
     if( a < 0 \&\& b > 0)
     for(int i = 0;i < b;i++) out1.push back(1);
     for(int i = 0; i < abs(a); i++) out1.erase(out1.begin() + i);
     }
     else if (a > 0 \&\& b < 0)
        for(int i = 0; i < a; i++) out1.push back(1);
        for(int i = 0;i < abs(b);i++) out1.erase(out1.begin() + i);
     }
     else if (a <0 && b <0){
        for(int i = 0; i < abs(a); i++) out1.push back(1);
        for(int i = 0;i < abs(b);i++) out1.push back(1);
       int c = out1.size();
       return -c;
     }
     else{
        for(int i = 0;i < a;i++) out1.push back(1);
        for(int i = 0;i < b;i++) out1.push back(1);
     }
     return out1.size();
};
```

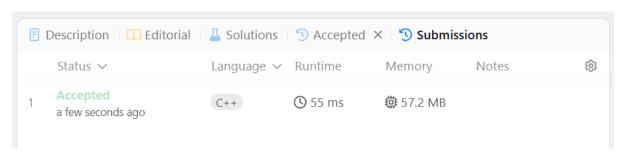


6) Beautiful Array

```
class Solution {
public:
    vector<int> beautifulArray(int n) {
        if (n==1) return {1};
        vector<int> arr = beautifulArray(n-1);
        vector<int> res;
        for (auto i: arr)
            if (2*i - 1 <= n)
                res.push_back(2*i-1);

        for (auto i: arr)
            if (2*i <= n)
                res.push_back(2*i);

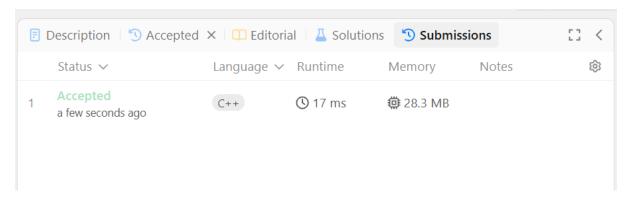
        return res;
        }
};</pre>
```



8) The Skyline Problem

```
class Solution {
public:
  vector<vector<int>>> getSkyline(vector<vector<int>>& buildings) {
  vector<pair<int, pair<int, int>>> events;
  for (auto& b : buildings) {
     events.push back({b[0], {-b[2], b[1]}});
    events.push back({b[1], {b[2], 0}});
  }
  sort(events.begin(), events.end());
  multiset<int> heights = \{0\};
  vector<vector<int>> result;
  int prevMaxHeight = 0;
  for (auto& e : events) {
     int x = e.first, h = e.second.first, r = e.second.second;
    if (h < 0) {
       heights.insert(-h);
     } else {
       heights.erase(heights.find(h));
     }
     int currMaxHeight = *heights.rbegin();
     if (currMaxHeight != prevMaxHeight) {
       result.push back({x, currMaxHeight});
       prevMaxHeight = currMaxHeight;
     }
  return result;
```

};



9) Reverse Pairs

```
class Solution {
public:
  int reversePairs(vector<int>& nums) {
     if (left \geq= right) return 0;
     int mid = left + (right - left) / 2;
     int count = mergeSortAndCount(nums, left, mid) + mergeSortAndCount(nums,
mid + 1, right);
     int j = mid + 1;
     for (int i = left; i \le mid; i++) {
       while (j \le right \&\& nums[i] > 2LL * nums[j]) j++;
       count += (j - (mid + 1));
     }
     vector<int> temp;
     int i = left, k = mid + 1;
     while (i \le mid \&\& k \le right) {
       if (nums[i] <= nums[k]) temp.push back(nums[i++]);
       else temp.push back(nums[k++]);
     }
     while (i \le mid) temp.push back(nums[i++]);
```

```
while (k \le right) temp.push back(nums[k++]);
     for (int i = left; i \le right; i++) {
       nums[i] = temp[i - left];
     }
     return count;
  int reversePairs(vector<int>& nums) {
     return mergeSortAndCount(nums, 0, nums.size() - 1);
  }
};

■ Description | □ Editorial | △ Solutions | ⑤ Accepted × ⑤ Submissions

                             Language ∨ Runtime
     Status ~
                                                       Memory
                                                                     Notes
                                                                                     (6)
     Accepted
                             C++
                                         (§ 580 ms
                                                       @ 243.5 MB
```

10) Longest Increasing Subsequence II

a few seconds ago

```
class Solution {
public:
    int st[(int)1e6 + 1];
    vector<int> A;
    void build(int tidx, int lo, int hi) {
        if(lo == hi) {
            st[tidx] = A[lo];
            return;
        }
        int mid = (lo + hi) / 2;
        build(2 * tidx, lo, mid);
        build(2 * tidx + 1, mid + 1, hi);
        st[tidx] = max(st[2 * tidx], st[2 * tidx + 1]);
```

```
}
int query(int tidx, int l, int r, int ql, int qr) {
  if(r < 1) {
     return 0;
  }
  if(1 == q1 \&\& r == qr) {
     return st[tidx];
  }
  int mid = (1 + r) / 2;
  if(qr \leq mid) \{
     return query(2 * tidx, l, mid, ql, qr);
  } else if(ql > mid) {
     return query(2 * tidx + 1, mid + 1, r, ql, qr);
  }
  int leftMax = query(2 * tidx, l, mid, ql, mid);
  int rightMax = query(2 * tidx + 1, mid + 1, r, mid + 1, qr);
  return max(leftMax, rightMax);
}
void update(int tidx, int l, int r, int aidx, int val) {
  if(l == r) {
     st[tidx] = val;
     return;
  }
  int mid = (1 + r) / 2;
  if(aidx <= mid) {</pre>
     update(2 * tidx, l, mid, aidx, val);
  } else {
```

```
update(2 * tidx + 1, mid + 1, r, aidx, val);
}
st[tidx] = max(st[2 * tidx], st[2 * tidx + 1]);
}
int lengthOfLIS(vector<int>& nums, int k) {
    A = nums;
    int n = A.size();
    for(int i = 0; i < n; i++) {
        int mxLen = query(1, 0, (int)1e5, max(0, nums[i] - k), nums[i] - 1);
        update(1, 0, (int)1e5, nums[i], mxLen + 1);
}
return st[1];
}
};</pre>
```

