Advanced Programming LAB II

ASSIGNMENT - 7

Submitted by,

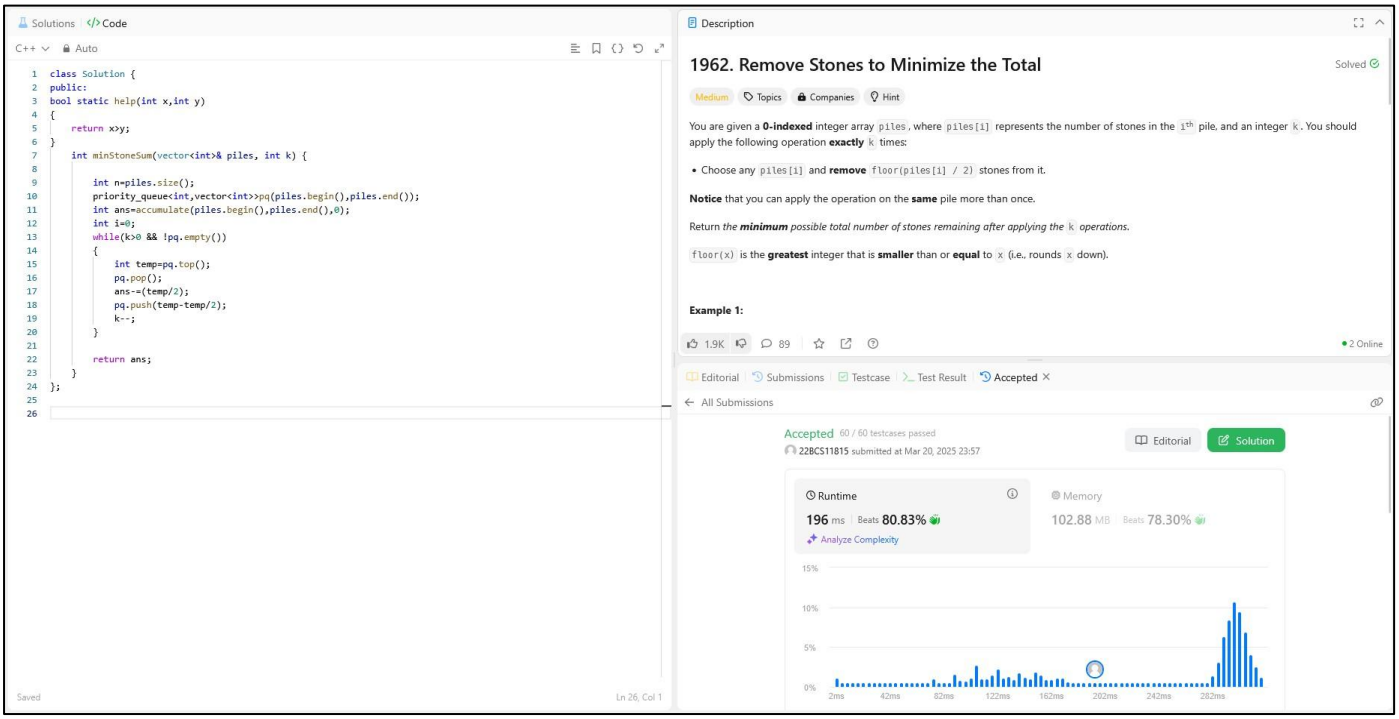
Anwar | 22BCS13961

22BCS_ IOT-637 (A)

1662. Remove Stones to Minimize the Total

https://leetcode.com/problems/remove-stones-to-minimize-the-total/description

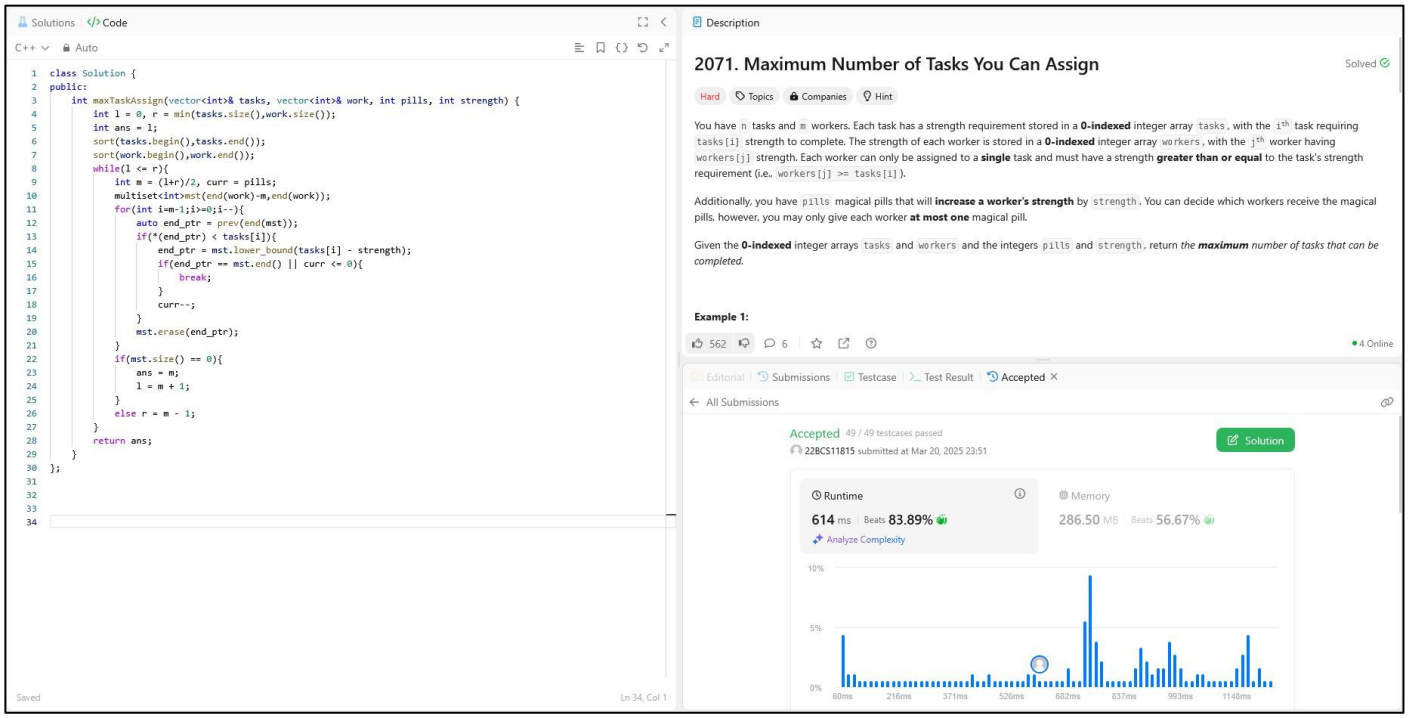
```
class Solution {
public:
bool static help(int x,int y)
{
    return x>y;
}
    int minStoneSum(vector<int>& piles, int k) {

        int n=piles.size();
        priority_queue<int,vector<int>>pq(piles.begin(),piles.end());
        int ans=accumulate(piles.begin(),piles.end(),🅓);
        int i=🅓;
        while(k>🅓 && !pq.empty())
        {
            int temp=pq.top();
            pq.pop();
            ans-=(temp/2);
            pq.push(temp-temp/2);
            k--;
        }

        return ans;
    }
};
```



2071. Maximum Number of Tasks You Can Assign

https://leetcode.com/problems/maximum-number-of-tasks-you-can-assign/description/

```
class Solution {
public:
    int maxTaskAssign(vector<int>& tasks, vector<int>& work, int pills, int strength) {
        int l = 0, r = min(tasks.size(),work.size());
        int ans = l;
        sort(tasks.begin(),tasks.end());
        sort(work.begin(),work.end());
        while(l <= r){
            int m = (l+r)/2, curr = pills;
            multiset<int>mst(end(work)-m,end(work));
            for(int i=m-1;i>=0;i--){
                auto end_ptr = prev(end(mst));
                if(*(end_ptr) < tasks[i]){
                    end_ptr = mst.lower_bound(tasks[i] - strength);
                    if(end_ptr == mst.end() || curr <= 0){
                        break;
                    }
                    curr--;
                }
                mst.erase(end_ptr);
            }
            if(mst.size() == 0){
                ans = m;
                l = m + 1;
            }
            else r = m - 1;
        }
        return ans;
    }
};
```



1827. Minimum Operations to Make the Array Increasing

https://leetcode.com/problems/minimum-operations-to-make-the-array-increasing/description

```
class Solution {
public:
    int maxTaskAssign(vector<int>& tasks, vector<int>& work, int pills, int strength) {
        int l = 🅓, r = min(tasks.size(),work.size());
        int ans = l;
        sort(tasks.begin(),tasks.end());
        sort(work.begin(),work.end());
        while(l <= r){
            int m = (l+r)/2, curr = pills;
            multiset<int>mst(end(work)-m,end(work));
            for(int i=m-1;i>=🅓;i--){
                auto end_ptr = prev(end(mst));
                if(*(end_ptr) < tasks[i]){
                    end_ptr = mst.lower_bound(tasks[i] - strength);
                    if(end_ptr == mst.end() || curr <= 🅓){
                        break;
                    }
                    curr--;
                }
                mst.erase(end_ptr);
            }
            if(mst.size() == 🅓){
                ans = m;
                l = m + 1;
            }
            else r = m - 1;
        }
        return ans;
    }
};
```