DSA SERIES

- Learn Coding

Topic to be Covered today

Binary Search on Answer

LETS START TODAY'S LECTURE

Binary Search on Answer

1283. Find the Smallest Divisor Given a Threshold

Using linear Search

```
class Solution {
public:
    int smallestDivisor(vector<int>& nums, int threshold) {
        int maxi = *max_element(begin(nums), end(nums));

        int n = nums.size();

        for (int d = 1; d <= maxi; d++) {
            int sum = 0;

            for (int i =0; i < n; i++) {
                sum += nums[i] / d;

                if (nums[i] % d != 0) {
                    sum++;
                }
            }
            if (sum <= threshold) {
                return d;
            }
        }

        return -1;
    }
};
```

Using Binary Search

```
class Solution {
public:
    int ComputeSum(vector<int>& nums, int divisor) {
        int sum = 0;
        for (int num : nums) {
            sum += num / divisor;
            if (num % divisor != 0) {
                sum++;
            }
        }
        return sum;
    }
    int smallestDivisor(vector<int>& nums, int threshold) {

        int left = 1;
        int right = *max_element(begin(nums), end(nums));

        int ans = INT_MAX;

        while (left <= right) {
            int mid = left + (right - left) / 2;
            int sum = ComputeSum(nums, mid);
            if (sum <= threshold) {
                ans = mid;
                right = mid - 1;
            } else {
                left = mid + 1;
            }
        }

        return ans;
    }
};
```

1482. Minimum Number of Days to Make m Bouquets

Using linear Search

```
class Solution {
public:
    int Count(vector<int>& bloomDay, int day, int k) {

        int made = 0;
        int cnt = 0;

        for (int i = 0; i < bloomDay.size(); i++) {
            if (bloomDay[i] <= day) {
                cnt++;
            } else {
                cnt = 0;
            }

            if (cnt == k) {
                made++;
                cnt = 0;
            }
        }
        return made;
    }



 int minDays(vector<int>& bloomDay, int m, int k) {
    int right = *max_element(begin(bloomDay), end(bloomDay));
    int left =1;

    int ans =-1;

    while(left<=right){
        int mid = left + (right-left)/2;

        int bouq = Count(bloomDay,mid,k);


};
```

```
  if(bouq>=m){
            ans =mid;
            right = mid-1;
        } else{
            left = mid+1;
        }
    }

    return ans;


}
```

Using Binary Search

```
class Solution {
public:
    int Count(vector<int>& bloomDay, int day, int k) {

        int made = 0;
        int cnt = 0;

        for (int i = 0; i < bloomDay.size(); i++) {
            if (bloomDay[i] <= day) {
                cnt++;
            } else {
                cnt = 0;
            }

            if (cnt == k) {
                made++;
                cnt = 0;
            }
        }
        return made;
    }
int minDays(vector<int>& bloomDay, int m, int k) {
        int maxi = *max_element(begin(bloomDay), end(bloomDay));

        for (int day = 1; day <= maxi; day++) {

            int bouq = Count(bloomDay, day, k);

            if (bouq >= m) {
                return day;
            }
        }
        return -1;
    }
};
```

1870. Minimum Speed to Arrive on Time

Using linear Search

```
class Solution {
public:
    double possible(vector<int>& dist, int speed) {
        double time = 0.0;
        int n = dist.size();

        for (int i = 0; i <= n - 2; i++) {
            double t = (double)(dist[i]) / (double)speed;

            time += ceil(t);
        }
        time += (double)(dist[n - 1]) / (double)speed;

        return time;
    }
    int minSpeedOnTime(vector<int>& dist, double hour) {
        int maxSpeed = 1e7;

        for (int speed = 1; speed <= maxSpeed; speed++) {
            if (possible(dist, speed) <= hour) {
                return speed;
            }
        }

        return -1;
    }
};
```

Using Binary Search

```
class Solution {
public:
    double possible(vector<int>& dist, int speed) {
        double time = 0.0;
        int n = dist.size();

        for (int i = 0; i <= n - 2; i++) {
            double t = (double)(dist[i]) / (double)speed;

            time += ceil(t);
        }
        time += (double)(dist[n - 1]) / (double)speed;

        return time;
    }

    int minSpeedOnTime(vector<int>& dist, double hour) {
        int left = 1;
        int right = 1e7;
        int minSpeed = -1;

        while (left <= right) {
            int mid = left + (right - left) / 2;

            if (possible(dist, mid) <= hour) {
                minSpeed = mid;
                right = mid - 1;
            } else {
                left = mid + 1;
            }
        }

        return minSpeed;
    }
};
```

Additional Questions to Practice

1. Aggressive Cows (GFG)

(Leetcode)

1. Maximum Candies Allocated to K Children
2. Most Profit Assigning Work
3. Maximum Value at a Given Index in a Bounded Array
4. Minimum Time to Repair Cars
5. Heaters
6. Maximum Number of Removable Characters
7. Earliest Second to Mark Indices I

Learn coding

THANK YOU