DSA SERIES

- Learn Coding

Topic to be Covered today

Binary Search on Answer

LETS START TODAY'S LECTURE

Binary Search on Answer

69. Sqrt(x)

```
class Solution {
public:
    int mySqrt(int x) {
        if(x == 1) return x;

        int ans = 0;

        long long left = 1;
        long long right = x/2;

        while(left <= right){

            long long mid = left + (right-left)/2;

            long long squared = mid * mid;

            if(squared == x){
                return mid;
            }
```

```
   else if(squared < x){
                 ans = mid;
                 left = mid+1;
            } else{
                 right = mid-1;
            }
        }

        return ans;
    }
};
```

1011. Capacity To Ship Packages Within D Days

```
class Solution {
public:
    int CountDays(vector<int>& weights, int capacity) {
        int day = 1, load = 0;
        int n = weights.size();

        for (int i = 0; i < n; i++) {
            if (load + weights[i] > capacity) {
                day = day + 1;
                load = weights[i];
            } else {
                load += weights[i];
            }
        }

        return day;
    }
    int shipWithinDays(vector<int>& weights, int days) {
        int left = 0;
        int right = 0;
int Max = INT_MIN;
int sum = 0;

    for (int num : weights) {
    sum += num;

    if (Max < num) {
        Max = num;
    }
}

left = Max;
right = sum;

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

    int day = CountDays(weights, mid);

    if (day <= days) {
        right = mid - 1;
    }
```

```
            else {
                left = mid + 1;
            }
        }

        return left;
    }
};
```

875. Koko Eating Bananas

```
class Solution {
public:

    long long CountHours(vector<int> &piles,int bph){
        long long hour = 0;

        int n = piles.size();

        for(int i =0;i<n;i++){
            hour += piles[i]/bph;

            if(piles[i] % bph !=0){
                hour++;
            }
        }
        return hour;
    }
    int minEatingSpeed(vector<int>& piles, int h) {

        long left = 1;
        long long right = *max_element(begin(piles),end(piles));

        long long ans = INT_MAX;

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

            long long hour = CountHours(piles,mid);

            if(hour <= h){
                ans = mid;
                right = mid - 1;
            } else{
                left = mid + 1;
            }
        }

        return ans;
    }
};
```

Learn coding

THANK YOU