DAY 33/180

Q1- Allocate Minimum Number of Pages

```
class Solution
{
    public:
       bool check(int arr[],int N,int M,int mid){
        int sum =0,stu=1;
        for(int i=0;i<N;i++){
            sum += arr[i];
            if(sum>mid){
                stu++;
                sum=arr[i];
            }
            if(stu>M) return 0;
        }
        return 1;
    }
    int findPages(int A[], int N, int M){
        if(N<M) return -1;
        int s=0,e=0,ans=-1;
        for(int i=0;i<N;i++){
            s=max(s,A[i]);
            e+=A[i];
        }
        while(s<=e){
            int mid=(s+e)/2;
            if(check(A,N,M,mid)==1){
                ans=mid;
                e=mid-1;
            }
            else{
                s=mid+1;
            }
        }
        return ans;
    }
};
```

Q2- The Painter's Partition Problem

```
class Solution{
  public:
    bool isPossible(int boards[], long long mid, int k, int n){
        long long sum=0, man=1;
        for(int i=0; i<n; i++){
            if((long long)boards[i]+sum<=mid){
                sum+=boards[i];
            }else{
                man++;
                sum=boards[i];
            }
        }
        if(man<=k) return false;
        return true;
    }
    long long minTime(int boards[], int n, int k){
        long long s=0, e=0;
        for(int i=0; i<n; i++){
            s=max(s,(long long)boards[i]);
            e+=boards[i];
        }
        while(s<=e){
            long long mid=s+(e-s)/2;
            if(isPossible(boards,mid,k,n)) s=mid+1;
            else e=mid-1;
        }
        return s;
    }
};
```

Q3- Capacity to ship Packages within D Days

```cpp
class Solution {
public:
    bool check(int mid,int days,vector<int>&arr){
        int n=arr.size();
        int sum=0,cnt=1;
        for(int i=0;i<n;i++){
            sum+=arr[i];
            if(sum>mid){
                cnt++;
                sum=arr[i];
            }
        }
        return cnt<=days;
    }
    int shipWithinDays(vector<int>& weights, int days) {
        int n=weights.size();
        int s=*max_element(weights.begin(),weights.end());
        int ans=1e9;
        int e=accumulate(weights.begin(),weights.end(),0);
        while(s<=e){
            int mid=(s+e)/2;
            if(check(mid,days,weights)){
                ans=mid;
                e=mid-1;
            }
            else{
                s=mid+1;
            }
        }
        return ans;
    }
};
```

Q4- Koko Eating Bananas.

```cpp
#define ll long long
class Solution {
public:
    bool check(ll mid,vector<int>&piles,ll h){
        int n=piles.size();
        ll time=0;
        for(int i=0;i<n;i++){
            if(piles[i]<mid){
                time++;
            }
            else{
                ll t=piles[i]/mid;
                if(piles[i]%mid) t++;
                time+=t;
            }
        }
        return time<=h;
    }
    int minEatingSpeed(vector<int>& piles, int h) {
        int n=piles.size();
        ll s=1,e=1e9+1;
        int ans=-1;
        while(s<=e){
            int mid=(e+s)>>1;
            if(check(mid,piles,h)){
                ans=mid;
                e=mid-1;
            }
            else{
                s=mid+1;
            }
        }
        return ans;
    }
};
```

Q5- Split Array Largest Sum

```cpp
class Solution {
  public:
    int splitArray(int a[] ,int n, int k) {
        int l=0;
        int r=1e9;
        int ans=1;
        int sum=0;
        while(l<=r){
            int m=(l+r)/2;
            int cnt=0;
            sum=0;
            for(int i=0;i<n;i++){
                if(sum+a[i]>m){
                   cnt++;
                   sum=a[i];
                   if(a[i]>m){
                       cnt=INT_MAX-100;
                       break;
                   }
                }
                else sum+=a[i];
            }
            cnt++;
            if(cnt<=k){
                ans=m;
                r=m-1;
            }
            else l=m+1;
        }
        return ans;
    }
};
```