Lecture-15

Book Allocation Problem || Aggressive Cows Problem || Binary Search Advanced Problems

Q1. **Allocate Books**
https://www.codingninjas.com/studio/problems/allocate-books_1090540

```
bool isPossible(vector<int>& arr, int n, int m, int mid){
int studentCount = 1;
int pageSum = 0;
for(int i=0;i<n;i++){
if(pageSum + arr[i] <= mid){
pageSum += arr[i];
}
else{
studentCount++;
if(studentCount>m || arr[i]>mid){
return false;
}
pageSum = arr[i];
}
}
return true;
}
int findPages(vector<int>& arr, int n, int m) {
// Write your code here.
int s=0;
int sum = 0;
for(int i=0;i<n;i++){
sum = sum+arr[i];
}
int e=sum;
int ans=-1;
int mid = s+(e-s)/2;
while(s<=e && n>=m){
if(isPossible(arr, n, m, mid)){
ans = mid;
e = mid-1;
```

```
}else{
s = mid+1;
}
mid = s+(e-s)/2;
}
return ans;
}
```

Q2. **Painter's Partition Problem**
https://www.codingninjas.com/studio/problems/painter's-partition-problem_1089557?source=youtube&campaign=love_babbar_codestudio2

```
bool isPossible(vector<int> &boards, int k, int mid){
int painterCount = 1;
int paintSum = 0;
for(int i=0;i<boards.size();i++){
if(paintSum+boards[i]<=mid){
paintSum+=boards[i];
}
else{
painterCount++;
if(painterCount>k || boards[i]>mid){
return false;
}
paintSum = boards[i];
}
}
return true;
}
int findLargestMinDistance(vector<int> &boards, int k)
{
// Write your code here.
int s = 0;
int sum=0;
for(int i=0;i<boards.size();i++){
sum+=boards[i];
}
int e = sum;
int ans = -1;
int mid = s+(e-s)/2;
while(s<=e){
if(isPossible(boards, k, mid)){
ans=mid;
```

```
e=mid-1;
}else{
s=mid+1;
}
mid = s+(e-s)/2;
}
return ans;
}
```

Q3. **Aggressive Cows**
https://www.codingninjas.com/studio/problems/aggressive-cows_1082559?source=youtube&campaign=love_babbar_codestudio2

```
bool isPossible(vector<int> &stalls, int k, int mid){
int cowCount = 1;
int lastPos = stalls[0];
for(int i=0;i<stalls.size();i++){
if(stalls[i]-lastPos >=mid){
cowCount++;
if(cowCount==k){
return true;
}
lastPos = stalls[i];
}
}
return false;
}
int aggressiveCows(vector<int> &stalls, int k)
{
// Write your code here.
sort(stalls.begin(), stalls.end());
int s = 0;
int maxi = -1;
for(int i=0;i<stalls.size();i++){
maxi = max(maxi, stalls[i]);
}
int e = maxi;
int ans = -1;
int mid = s + (e-s)/2;
while(s<=e){
if(isPossible(stalls, k, mid)){
ans = mid;
s = mid+1;
}else{
```

```
e = mid-1;
}
mid = s + (e-s)/2;
}
return ans;
}
```

Q4. HW
https://www.spoj.com/problems/EKO/
Lumberjack Mirko needs to chop down **M** metres of wood. It is an easy job for him since he has a nifty new woodcutting machine that can take down forests like wildfire. However, Mirko is only allowed to cut a single row of trees.

Mirko‟s machine works as follows: Mirko sets a height parameter **H** (in metres), and the machine raises a giant sawblade to that height and cuts off all tree parts higher than **H** (of course, trees not higher than **H** meters remain intact). Mirko then takes the parts that were cut off. For example, if the tree row contains trees with heights of 20, 15, 10, and 17 metres, and Mirko raises his sawblade to 15 metres, the remaining tree heights after cutting will be 15, 15, 10, and 15 metres, respectively, while Mirko will take 5 metres off the first tree and 2 metres off the fourth tree (7 metres of wood in total).

Mirko is **ecologically** minded, so he doesn‟t want to cut off more wood than necessary. That‟s why he wants to set his sawblade as high as possible. Help Mirko find the **maximum integer height** of the sawblade that still allows him to cut off **at least M** metres of wood.

```
#include<iostream>
using namespace std;
bool isPossible(int arr[], int n, int k, int mid){
int total = 0;
for(int i=0;i<n;i++){
if(mid<arr[i]){
total += arr[i] - mid;
}
}
return total >= k; // Return true if enough wood can be collected
}
```

```
int toCut(int arr[], int n, int k){
int s = 0;
int maxi = 0;
for(int i=0;i<n;i++){
maxi = max(maxi, arr[i]);
}
int e=maxi;
int ans=-1;
while(s<=e){
int mid = s + (e-s)/2; // Calculate mid inside the loop
if(isPossible(arr, n, k, mid)){
ans=mid;
s = mid+1;
}else{
e=mid-1;
}
}
return ans;
}
int main(){
int N = 5; // Number of trees
int M = 20; // Required wood amount
int arr[] = {4,40,42,26,46}; // Heights of trees
cout<<"Maximum height to cut is: "<<toCut(arr, N, M);
return 0;
}
```