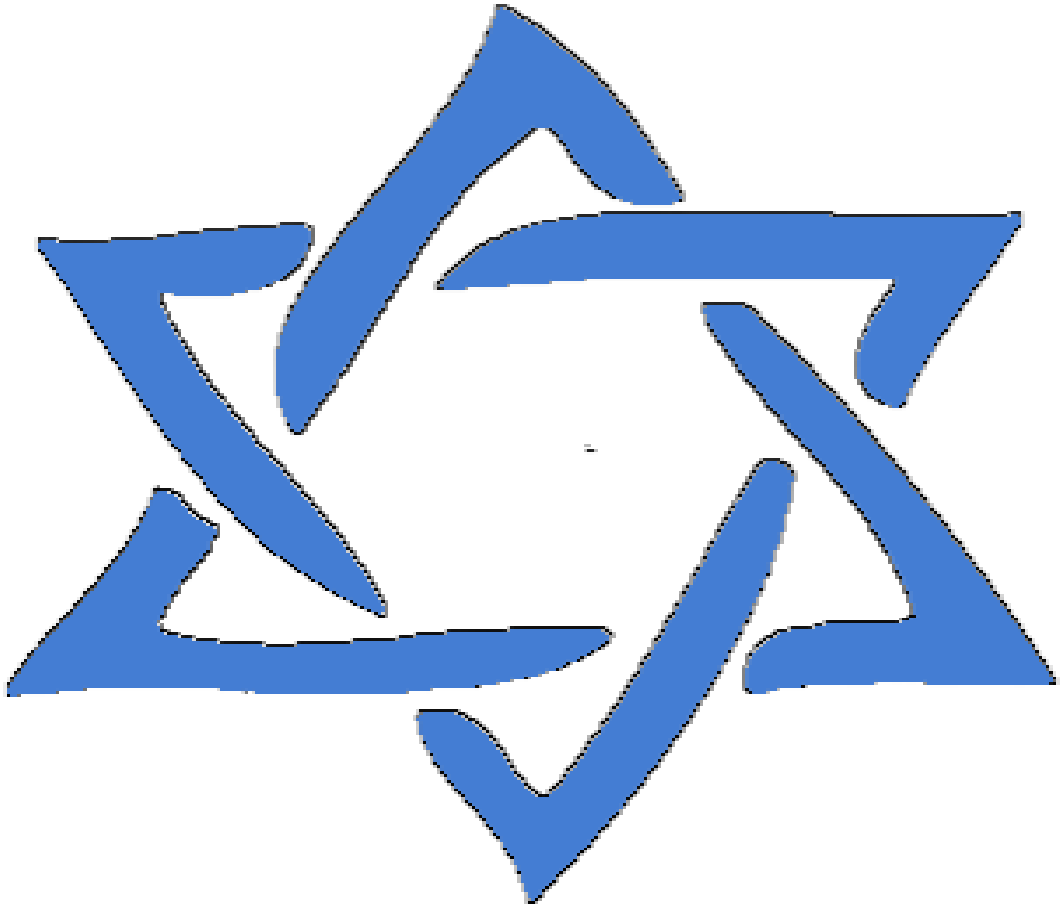
Binary Search
Lesson 7



THE OLD MONK

Big Chandan is a dire lover of Biryani, especially Old Monk's Biryani. Today, he went over to have some of it. To his surprise, the waiter turns out be to be a coding geek and refuses to serves him unless Chandu solves his two- arrays problem, stated as:

Given two non-increasing arrays of integers A,B i.e A[i] >= A[i+1] and B[i] >= B[i+1] and for all i, 0 ≤ i < n-1.

The monkiness of two numbers is given by: M (A[i],B[j]) = j - i , if j >=i and B[j] >= A[i], or 0 otherwise.

Find the monkiness of the two arrays, that is given by: M (A,B)= max (M(A[i],B[j])) for 0≤ i, j< n-1.

INPUT

T = 2

N = 9

A[] = { 7, 7, 3, 3, 3, 2, 2, 2, 1 }

B[] = { 8, 8, 7, 7, 5, 5, 4, 3, 2 }

N = 6

A[] = { 6, 5, 4, 4, 4, 4 }

B[] = { 2, 2, 2, 2, 2, 2 }

OUTPUT

5

0

EXPLANATION:

In the first case, we can see that 3 in the second array is the number which is equal to the 3 in the first array, and the difference between their positions is 5. So, the answer is 5.

In the second test case, no number in the second array is greater or equal to elements in the first array. So, the answer is 0.

ALGORITHM:

Basic approach

For an **A[i]**, you have to find an element **B[j]** which is greater than **A[i]** and **(j >= i)** . There will be n no. of pairs satisfying this condition. We have to find the maximum difference of i and j.

Better Approach

To do this, for every **A[i]** , traverse through **B[]** to find the last number greater than or equal to **A[i]** , find **(j - i)** and keep updating the answer to return the maximum difference.

CODE:

```
#include<bits/stdc++.h>
using namespace std;

#define ll long long
#define endl "\n"

int main()
{
    ios_base::sync_with_stdio(false);
    cin.tie(NULL);
    cout.tie(NULL);

    int t; cin>>t;
    while(t--)
    {
        ll n; cin>>n;
        ll a[n], b[n];
        for(int i = 0; i<n; i++)
            cin>>a[i];
        for(int i = 0; i<n; i++)
            cin>>b[i];

```

```
23.         ll ans = 0;
24.         for(int i = 0; i<n; i++)
25.         {
26.             ll low = i, high = n-1, pos = 0;
27.             while(low <= high)
28.             {
29.                 ll mid =  low + (high - low)/2;
30.                 if(b[mid] >= a[i])
31.                 {
32.                     pos = mid;
33.                     low = mid + 1;
34.                 }
35.                 else
36.                     high = mid - 1;
37.             }
38.             ans = max(ans, pos - i);
39.         }
40.         cout<<ans<<endl;
41.     }
42.}
```

Que link: [The Old Monk]

WHEN WILL SHE TALK?

You know a shop where you can buy chocolates for your friend. The shop has limited availability of chocolates. On day i, there are only Ai chocolates available in the shop. Your friend is angry with you and she asks you to buy her X chocolate otherwise she won't talk to you. Also due to COVID 19 pandemic, the government has imposed lockdown from Lth day to Rth day which means the shop will be closed from L to R (both inclusive). Now you want to find out the no of days she won't talk to you. If you fail to buy her X chocolate on or before nth day, she will never talk to you and print -1 in this case.

Note: She will start talking to you the day after you fulfill her demand.

INPUT

N = 5

A[] = { 4, 3, 6, 2, 8 }

Q = 3

L = 2, R= 3, X = 3

L = 2, R= 3, X = 8

L = 2, R= 3, X = 15

OUTPUT

1

5

-1

EXPLANATION:

In the first query, l = 2 and r = 3 and x = 3. So you will buy 3 chocolate on 1st day. In this way you will have total 3 chocolate on 1st day. So, answer is 1

In the second query, l = 2, r = 3 and x = 8. So you will buy 4 chocolate on 1st day, can't buy on 2nd and 3rd day, 2 chocolate on 4th day and 2 chocolate on 5th day. In this way you will have total 8 chocolate on 5th day. So answer is 5.

In the third query, l = 2, r = 3 and x = 15. So you will buy 4 chocolate on 1st day, can't buy on 2nd and 3rd day, 2 chocolate on 4th day and 8 chocolate on 5th day but still total chocolate you bought is 14, so the answer is -1.

ALGORITHM:

Simple approach

Linearly traverse through A[] and keep adding the chocolates, if the day lies between lockdown i.e. (i >= L && i <= R) don't add. Return the day when first time the sum exceeds X. If the array is completely traversed, return -1.

Better approach

Firstly, create a Sumarray, that stores the sum of elements upto that position. Find the position of the minimum element greater than equal to X. If it is less than L, (i.e. before lockdown) , simply print the position.

If it is greater than L, we know he can not buy chocolates during the lockdown. We will find the number of chocolates that he could have bought between L and R and then simply find an element (X + non-bought chocolates) in the array after R.

Eg. arr[] = { 4, 3, 6, 2, 8 } Sumarray[] = { 4, 7, 13, 15, 23 }

L = 2, R = 3 Here, he can not buy chocolates on 2nd and 3rd day

So on 4th and 5th day he has 6 and 14 chocolates and not 15 and 23 respectively. Instead of altering the array, we will simply find (X+ 9) chocolates in the [R+1 …. N] array.

CODE:

```
1. #include<bits/stdc++.h>
2. using namespace std;
3.
4. #define ll long long
5. #define endl "\n"
6.
7. ll BS(ll arr[], ll low, ll high, ll ele)
8. {
9.   ll ans = high + 1;
10.    while(low <= high)
11.    {
12.        ll mid = low + (high - low)/2;
13.        if(arr[mid] >= ele)
14.        {
15.            ans = mid;
16.            high = mid - 1;
17.        }
18.        else
19.            low = mid + 1;
20.    }
21.    return ans;
22.  }
```

```
23.
24.  int main()
25.  {
26.    ios_base::sync_with_stdio(false);
27.    cin.tie(NULL);
28.    cout.tie(NULL);
29.
30.    ll N;  cin>>N;
31.    ll arr[N];
32.    ll sum =0;
33.    ll sumarr[N];
34.    for(ll i = 0; i<N; i++)
35.    {
36.        cin>>arr[i];
37.        sum += arr[i];
38.        sumarr[i] = sum;
39.    }
40.
41.    ll Q; cin>>Q;
42.    while(Q--)
43.    {
44.        ll l,r, x;
45.        cin>>l>>r>>x;
```

```
46.         if(x == 0)
47.         {
48.             cout<<0<<endl;
49.             continue;
50.         }
51.
52.         ll ind = BS(sumarr, 0, N-1, x);
53.  //ll ind = lower_bound(sumarr, sumarr + N, x) - sumarr;
54.
55.         if(ind < l - 1)
56.         {
57.             cout<<ind + 1<<endl;
58.             continue;
59.         }
60.
61.         ll remove = l>1 ? sumarr[r-1] - sumarr[l-2] :
   sumarr[r-1];
62.
63.         ll indN = BS(sumarr, r, N-1, x + remove);
64.  //ll indN = lower_bound(sumarr+r, sumarr+N, x+remove)
   - sumarr;
65.
66.         if(indN < N)
```

```
67.             cout<<indN + 1<<endl;
68.         else
69.             cout<<-1<<endl;
70.     }
71. }
```

Que link: [When will she talk]

A PLANE JOURNEY

A flight company has to schedule a journey of groups of people from the same source to the same destination. Here, represents the number of people in each group. All groups are present at the source. The flight company has planes which represent the capacity of each plane.

You are required to send all groups to destination with the following conditions:

Each plane can travel from the Source to Destination with only one group at a time such that the capacity of a plane is enough to accommodate all people in that group.

All people belonging to the same group travel together.

Every plane can make multiple journeys between source and destination.

It costs 1 unit of time to travel between source to destination and vice versa.

Note: Multiple planes can fly together and also it is not necessary for planes to end their journey at the source.

Determine the minimum time required to send all groups from the source to the destination.

INPUT

N = 4, M = 3

A[] = { 2, 1, 2, 6 }

B[] = { 5, 5, 6 }

OUTPUT

3

Explanation:

In T = 3.

Every plane can make 2 trips.

Plane with capacity 6 , send one group with capacity 6 and another with capacity 2. Other two planes can send each group in single trip

It is not possible to send groups in time unit less than 3

ALGORITHM:

Sort arrays A[] and B[].

Create a vector to store the number of rounds of each plane.

For every group of passengers in A[], find the least capacity plane possible (T) and then increment the round of that plane.

In the next iteration, reduce the search space such that low = T + 1 (T calculated in above step).

If at any point, no such plane exists in the search space, then increase the search space to complete the array (such that low = 0 and high = M-1) for the next round .

At the end find the maximum number of rounds taken by the planes, and return (2 * rounds - 1) unit time.

CODE:

```
#include<bits/stdc++.h>
using namespace std;
#define ll long long
#define endl "\n"

ll BS(ll arr[], ll low, ll high, ll ele)
{
    ll ans = -1;
    while(low<= high)
    {
        ll mid = low + (high - low)/2;
        if(arr[mid] >= ele)
        {
            ans = mid;
            high = mid - 1;
        }
        else
            low = mid + 1;
    }
    return ans;
```

```
21. }
22.
23. int main()
24. {
25.     ios_base::sync_with_stdio(false);
26.     cin.tie(NULL);
27.     cout.tie(NULL);
28.
29.     ll n,m;
30.     cin>>n>>m;
31.     ll a[n], b[m];
32.     for(int i = 0; i<n; i++)
33.         cin>>a[i];
34.     for(int i = 0; i<m; i++)
35.         cin>>b[i];
36.
37.     sort(a, a+n);
38.     sort(b, b+m);
39.
40.     int flag = 0;
41.     vector<ll> answer (m, 0);
42.     ll low = 0,high = m-1;
43.     ll i = 0;
```

```
44.     while(i<n)
45.     {
46.         ll ans = BS(b, low, high, a[i]);
47.         if(ans == -1)
48.         {
49.             if(low == 0 && high == m-1)
50.             {
51.                 cout<<-1<<endl;
52.                 flag++;
53.                 break;
54.             }
55.             else
56.             {
57.                 low = 0;
58.                 high = m-1;
59.                 continue;
60.             }
61.         }
62.         answer[ans] += 1;
63.         low = ans + 1;
64.         i++;
65.     }
66.
```

```
67.     if(flag == 0)
68.     {
69.         sort(answer.begin(), answer.end());
70.         cout<<2 * answer.back() - 1<<endl;
71.     }
72. }
```

Que link: [A plane journey]

MANGO THIEF

In a village there is a merchant who owns many mango trees (M) and each mango tree has a boundary which has some height (H) . There are some children (N) in the village who want to steal mangoes but for stealing they have to clear the boundary.

Now , you will be given the jumping power of all the children and the height of each boundary with the number of mangoes on the tree.

You have to calculate how many mangoes the children are going to steal.

INPUT

N = 5 , M = 4

Children[] = { 1, 4, 2, 3, 5}

h = 0, m = 5

h = 8, m = 20

h = 3, m = 3

h = 2, m = 9

OUTPUT

5 17 14 17 17

EXPLANATION

for 1 he can only access boundary with 0 height - 5

for 4 he can access boundary with 0,2,3 height - 5+3+9 = 17

for 2 he can access boundary with 0,2 height - 5+9 = 14

for 3 he can access boundary with 0,2,3 height - 5+3+9 = 17
for 5 he can access boundary with 0,2,3 height - 5+3+9 = 17

ALGORITHM:

Sort the trees according to the height of the trees.
Find the sum array for the mangoes.
For every element in the children array,

- In the height array find the maximum element less than or equal to their power of jumping.
- Now sum all the mangoes upto that index and print it.

CODE:

```
#include<bits/stdc++.h>
using namespace std;

#define ll long long
#define endl "\n"

ll BS(ll arr[], ll n, ll ele)
{
    ll low = 0, high = n-1, ans = -1;
    while(low <= high)
    {
        ll mid = low + (high - low)/2;
        if(arr[mid] <= ele)
```

```
14.         {
15.             ans = mid;
16.             low = mid + 1;
17.         }
18.         else
19.             high = mid - 1;
20.     }
21.     return ans;
22. }
23.
24. void sortTogether(ll arr1[], ll arr2[], ll n)
25. {
26.     pair<ll,ll> arr[n];
27.     for(int i = 0; i<n; i++)
28.     {
29.         arr[i].first = arr1[i];
30.         arr[i].second = arr2[i];
31.     }
32.
33.     sort(arr, arr+n);
34.
35.     for(int i = 0; i<n; i++)
36.     {
```

```
37.         arr1[i] = arr[i].first;
38.         arr2[i] = arr[i].second;
39.     }
40. }
41.
42. int main()
43. {
44.     ios_base::sync_with_stdio(false);
45.     cin.tie(NULL);
46.     cout.tie(NULL);
47.
48.     ll n,m; cin>>n>>m;
49.     ll arr[n];
50.     for(int i = 0; i<n; i++)
51.         cin>>arr[i];
52.
53.
54.     ll size[m], mango[m];
55.     for(int i = 0; i<m; i++)
56.         cin>>size[i]>>mango[i];
57.
58.     sortTogether(size, mango, m);
59.
```

```
60.     ll sum = 0;
61.     for(int i = 0; i< m; i++)
62.     {
63.         sum += mango[i];
64.         mango[i] = sum;
65.     }
66.
67.     for(int i = 0; i<n; i++)
68.     {
69.         ll ind = BS(size, m, arr[i]);
70.    //ll ind = upper_bound(size, size + m, arr[i]) - size -1;
71.
72.         if(ind == -1)
73.             cout<<0<<" ";
74.         else
75.             cout<<mango[ind]<<" ";
76.     }
77. }
```

Que link: [MANGO THIEF]

PRACTICE QUES :

1. Maximize Function!
2. zoro goes sword shopping
3. The furious five
4. Discover the monk
5. Foo and Exams
6. Monk's encounter with polynomial
7. Bishu and Soldiers