Binary search
Lesson 5

DO YOU REALLY UNDERSTAND BINARY SEARCH ?

Your seniors from the Programming Club made a lot of effort in teaching you binary search. They are looking forward to seeing a lot of accepted solutions on this problem. They might even treat everyone who solves this one.

You are given a set of parallel lines having slope -1 on the 2D coordinate. None of the parallel lines are overlapping. All the lines have a positive integral y intercept (y intercept is the y coordinate of the point where x coordinate is 0).

There are Q queries. In each of the queries you are given a point P(x,y). For each point, calculate and print the CPI of the point. CPI of a point is defined as the number of lines that lie between origin and the given point.

It is assured that the point does not lie on any of the lines.

INPUT :

N = 2

A[] = 5 3

Q = 2

x1 = 1 , y1 = 1

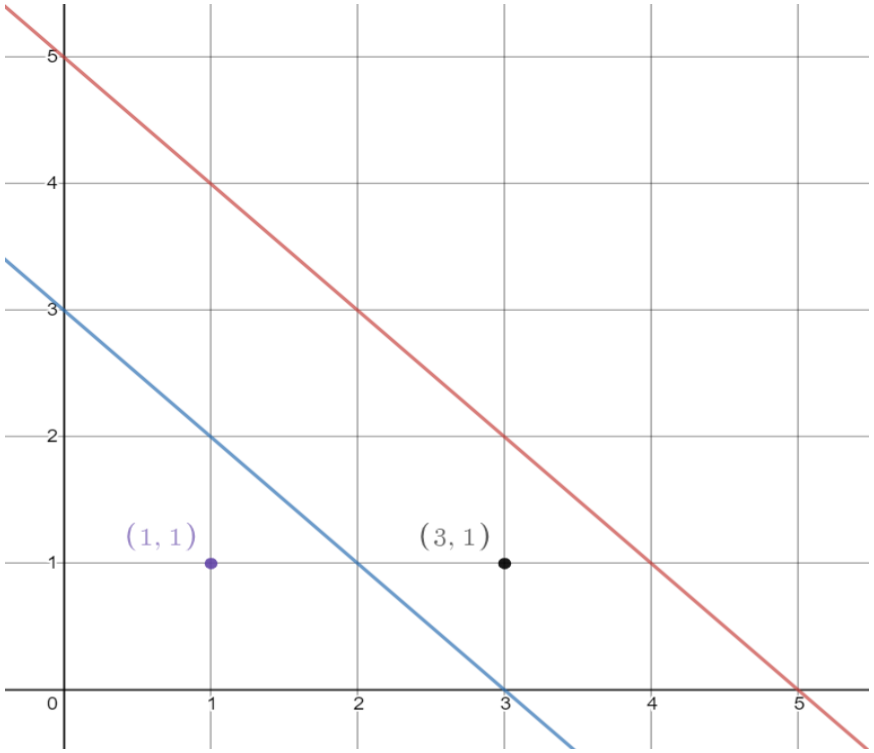
x2 = 3 , y2 = 1

OUTPUT :

0

1

EXPLANATION :



ALGORITHM :

Using Binary search Check the position of point by putting it in the equation x+y=c .

1. If x1 + y1 > c line lies below the point .
2. If x1 + y1 < c line lies above the point .
3. Otherwise the point is on the line.

Now count the no. of points that lie above the line and print the same.

TIME COMPLEXITY : O(logn)

CODE :

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

int binary_search(int arr[], int l, int h, int x)
{
  while (l <= h) {
     int mid = (l + h) / 2;
     if (arr[mid] <= x)
        l = mid + 1;
        else
        h = mid - 1;
    }
    return h+1;
  }

 int main() {
     int n,q;
     cin>>n;
     int A[n];
     for(int i=0;i<n;i++)
     cin>>A[i];
     cin>>q;
```

```
23.     int x[q],y[q],res[q];
24.     for(int i=0;i<q;i++){
25.     cin>>x[i]>>y[i];
26.     res[i]=0;
27.     }
28.     sort(A,A+n);
29.     for(int i=0;i<q;i++){
30.             cout<<binary_search(A,0,n-1,x[i]+y[i])<<endl;
31.     }
32. }
```

Ques link : Do you really understand binary search ?

BABLU KI BABLI

Our dear friend Bablu loves Babli. To impress Babli he needs to collect chocolates for her. He collects *Ai* chocolates for *N* number of days. Babli now needs the chocolates.

Babli asks Bablu *Q* queries, in each query she asks him the number of days he takes to collect *X* number of chocolates.

INPUT :

N = 7, Q = 3

A[N] = 2, 3, 6, 1, 4, 9,1

T[Q] = 5, 6, 10

OUTPUT :

2

3

3

ALGORITHM :

First Find out the array with Cumulative sum of original array elements.

Naive approach :

Iterate linearly and find out the index where cumulative sum is greater or equal to the query.

Binary Search :

Use the Iterative or Recursive approach of Binary search to find out the index where cumulative sum is greater or equal to the query.

TIME COMPLEXITY : *O(logn)*

CODE :

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

int ceilBinarySearch(int n, long long int cumulative[], long long int
key)
{
   int start = 0, end = n-1;
   int res = -1;
   while(start<=end)
   {
       int mid = start + (end-start)/2;
       if (cumulative[mid]==key)
       return (mid+1);
       else if (cumulative[mid]<key)
       start = mid + 1;
```

```
16.      else if (cumulative[mid]>key)
17.      {
18.          res = mid;
19.          end = mid - 1;
20.          }
21.  }
22.      return (res+1);
23.    }
24.
25.    int main(){
26.      ios_base::sync_with_stdio(0);
27.      cin.tie(0);
28.      cout.tie(0);
29.      int n;
30.      cin >> n;
31.  int a[n];
32.        int t;
33.      cin >> t;
34.      long long int cumulative[n] = {0};
35.      cin >> a[0];
36.      cumulative [0] = a[0];
37.      for (int i=1; i<n; i++)
38.      {
39.          cin >> a[i];
40.          cumulative[i] = a[i] + cumulative[i-1];
41.  }
42.      for (int i=0; i<t; i++)
```

```
43.     {
44.         long long int q;
45.         cin >> q;
46.         cout << ceilBinarySearch(n, cumulative, q) << endl;
47.     }
48.   }
```

Ques link : Bablu ki babli

ASSIGNMENTS

Mike is very frustrated, due to the college Assignments and asks his N friends to help him to complete M Assignments.

But each of his friends takes a different amount of time to complete any assignment, 1st friend takes a0 minutes, 2nd friend takes a1 minutes, and so on.

At a time, any of Mike's friends can do only one assignment. He may choose to do the next assignment only after he completes the current assignment.

Find the minimum time taken to complete all the M assignments.

Note: Mike is not feeling well so he does not do any assignments, All the work is done by his friend. It is NOT necessary that M is less than N. Also, it is possible for any of Mike's friends to simultaneously complete different assignments.

INPUT :

N = 3 , M = 11

A[] = 1 , 2 , 3

OUTPUT :

ALGORITHM :

Do Binary search on elements starting from low = 1 to high = 1e5 and find no. of assignment possible in time t = mid ,by dividing it with the time taken by friend in making 1 assignment.

TIME COMPLEXITY : O(NlogN)

CODE :

```
1. #include <bits/stdc++.h>
2. using namespace std;
3.
4. long long solve(long long arr[],long long N,long long key){
5.
6.      long long sum=0;
7.      for(long long i=0;i<N;i++){
8.          sum+=key/arr[i];
9.      }
10.       return sum;
11.}
12.
13.  int main(){
14.
15.  long long N,M,lo,hi,mid,ans=0;
16.
17.  cin>>N>>M;
```

```
18.
19.   long long A[N];
20.
21.   for(long long i=0;i<N;i++){
22.       cin>>A[i];
23.   }
24.
25.   lo=1;
26.   hi=1e5;
27.
28.   while(lo<=hi){
29.       mid=(lo+hi)/2;
30.       if(solve(A,N,mid)>=M){
31.           ans=mid;
32.           hi=mid-1;
33.       }
34.       else{
35.           lo=mid+1;
36.       }
37.   }
38.   cout<<ans<<endl;
39.   return 0;
40.   }
```

Ques link : Assignments

STONE 'S LOVE

Senorita likes stones very much. As she is fond of collecting beautiful stones, everyday she finds some of the stones beautiful and collects it in her bottle.

You are given the number of stones, she collects each day for N numbers of days, starting from the very first day. Now you are given Q queries, in each query her friend shambhavi asks you the number of days she has taken to collect M number of stones. Please note that each query contains the different number of M.

INPUT :

N = 5 , M = 4

Arr[] = 1 2 3 4 5

3 8 10 14

OUTPUT :

2

4

4

5

ALGORITHM :

First Find out the array with Cumulative sum of original array elements.

Naive approach :

Iterate linearly and find out the index where cumulative sum is greater or equal to the query.

Binary Search :

Use Iterative or Recursive approach of Binary search to find out the index where cumulative sum is greater or equal to the query.

TIME COMPLEXITY : *O(logn)*

CODE :

```
#include<bits/stdc++.h>
using namespace std;
#define endl "\n"
#define ll long long int
int main(){
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  cout.tie(NULL);
   ll n,q;cin>>n>>q;
          ll sum[n+1];
  sum[0]=0;
```

```
    for(ll i=1;i<=n;i++){
        ll a;
        cin>>a;
        sum[i]=a+sum[i-1];
    }
    while(q--){
        ll a,ans=n;cin>>a;
        ll lo=1,hi=n;
        while(lo<=hi){
            ll mid = lo+(hi-lo)/2;
            if(a<=sum[mid]){
                hi=mid-1;
                ans=mid;
            }
            else lo=mid+1;
        }
        cout<<ans<<endl;
    }
    return 0;
}
```

Ques link : stone's love

BIKE RACING

Geek is organising a bike race with N bikers. The initial speed of the ith biker is denoted by H_i Km/hr and the acceleration of ith biker as A_i Km/Hr2. A biker whose speed is 'L' or more, is considered be a fast biker. The total speed on the track for every hour is calculated by adding the speed of each fast biker in that hour. When the total speed on the track is 'M' kilometers per hour or more, the safety alarm turns on.

Find the minimum number of hours after which the safety alarm will start.

INPUT:

N = 3, M = 400, L = 120

H = {20, 50, 20}

A = {20, 70, 90}

OUTPUT: 3

EXPLANATION:

Speeds of all the Bikers at ith hour

Biker1= [20 40 60 80 100]

Biker2= [50 120 190 260 330]

Biker3= [20 110 200 290 380]

Initial Speed on track = 0

because none of the biker's speed is fast enough.

Speed on track after 1st Hour= 120

Speed on track after 2nd Hour= 190+200=390

Speed on track after 3rd Hour= 260+290=550

Alarm will start at 3rd Hour.

ALGORITHM :

Do Binary search on time starting from low = 0th hour to high = 1e10th hour and calculate the sum of speed of each biker if exceeding M for mid=(low+high)/2 th hour. Compare if sum of speed is greater than M search in left side otherwise in right side.

TIME COMPLEXITY : O(N*log(max(L,M)))

CODE :

```
1. class Solution{
2. public:
3. long calc(long x,long N,long L,long H[],long A[])
4. {
5. long total_speed=0;
```

```
6. for(long i=0;i<N;i++) {
7.    long spp=x*A[i]+H[i];
8.    if(spp>=L)
9. {
10.    total_speed+=spp;
11.}
12.    }
13.    return total_speed;
14.    }
15.       long buzzTime(long N, long M, long L, long H[], long A[])
16.       {
17.           // code here
18.          long l=0;
19.    long r=1e10;
20.    long ans=0;
21.    while(l<=r)
22.    {
23.    long mid=l+(r-l)/2;
24.    long x=calc(mid,N,L,H,A);
25.    if(x>=M)
26.    {
27.    ans=mid;
28.    r=mid-1;
29.    }
30.    else
```

```
31.    {
32.    l=mid+1;
33.    }
34.    }
35.    return ans;
36.    }
37.    };
```

Ques link : [Bike Racing]

HIGHEST AVERAGE

You are given an array A of length N. You have to choose a subset S from given array A, such that the average of S is less than K. You need to print the maximum possible length of S.

INPUT:

N = 5

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

Q = 5

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

OUTPUT:

0

2

4

5

5

EXPLANATION:

In the first query, there is no possible subset such that its average is less than 1.

In the second query, you can select the subset {1,2}.

In the third query, you can select the subset {1,2,3,4}.

In the fourth and fifth query, you can select the complete array {1,2,3,4,5}.

ALGORITHM :

Firstly ,we make sum array and average array that stores sum and average of all previous elements in the array. Then take a variable l=1 and h=n .

Start a loop and it goes till l is less than high, every time calculates the mid, if

Average of mid is less than the given no. then go into right half and store the mid into a variable.. Else go into left half , Finally return the answer.

TIME COMPLEXITY : O(N)

CODE :

```
1. #include<bits/stdc++.h>
2. #define ll long long int
3. #define endl "\n"
4. using namespace std;
5.
6. int main()
7. {
8.  ios_base::sync_with_stdio(false);
9.  cin.tie(NULL);
10.     ll n;
11. cin>>n;
```

```
12.    ll a[n+1];
13.    for(int i=1;i<=n;i++)
14.    cin>>a[i];
15.    sort(a+1,a+n+1);
16.    ll sum[n+1];
17.    sum[1]=a[1];
18.     for(int i=2;i<=n;i++)
19.       sum[i]=sum[i-1]+a[i];
20.    ll prefix[n+1];
21.    for(int i=1;i<=n;i++)
22.       prefix[i]=sum[i]/i;
23.    ll q;
24.    cin>>q;
25.    while(q--)
26.    {
27.          ll k;
28.          cin>>k;
29.
30.          if(k<=a[1])
31.          cout<<"0"<<endl;
32.          else
33.          {
34.             ll l=1,h=n,id=-1;
35.           while(l<=h)
36.           {
```

```
37.             ll mid=(l+h)/2;
38.             if(prefix[mid]<k)
39.             {
40.                 l=mid+1;
41.                 id=mid;
42.             }
43.             else
44.             {
45.                 h=mid-1;
46.             }
47.         }
48.         cout<<id<<endl;
49.       }
50.    }
51.  }
```

QUES LINK: [Highest Average]

PRACTICE QUESTIONS

1. Coins
2. highest average <Nissan>
3. Alice Bob Carrey (ABC)
4. c impresses everyone