Soluciones OIFem II Nivel 1

Entrenamiento 5

Watermelon

Teoría

- vectores

Solución

Tenemos un contador, que inicializamos en cero. Pasamos por el vector de los puntos de los participantes y si son mayor o iguales que el del participante k y mayores que cero sumamos 1 al contador. Al estar ordenados de mayor a menor, en cuanto algún número sea menor que el número de puntos del participante k dejamos de buscar.

Link al problema

Código

```cpp
#include <vector>
#include <iostream>
using namespace std;

int main(){
  int n,k;
  cin>>n>>k;
  k--;
  vector<int>score(n);
  for(int i=0;i<n;i++)cin>>score[i];
  int contador=0;
  for(int i=0;i<n;i++){
  if(score[i]>=score[k]){
    if(score[i]>0)contador++;
  }
  else break;
  }
  cout<<contador;
  return 0;
}
```

Binary Search-Basic

Teoría

- búsqueda binaria

Solución

El código como implementado en clase.

Link al problema

Código

```cpp
#include <iostream>
#include <utility>
#include <algorithm>
#include <vector>
using namespace std;
int busquedabinaria(vector<int>&lista,int key){
  int low=0,hi=lista.size(),medio;
  while(low<=hi){
    medio=low+(hi-low)/2;
    if(lista[medio]==key){
      return medio;
    }
    else if(lista[medio]<key){
      low=medio+1;
    }
    else{
      hi=medio-1;
    }
  }
  return -1;
}
int main ()
{ int N,key;
  cin>>N;
  vector<int>lista(N);
  for(int i=0;i<N;i++)cin>>lista[i];
  cin>>key;
  sort(lista.begin(),lista.end());
  cout<<busquedabinaria(lista,key);
  return 0;
}
```

CSUMQ - Cumulative Sum Query

Teoría

- suma de prefijos

Solución

Suma de prefijos como la de clase, segunda parte del código

Link al problema

Código

```cpp
#include <vector>
#include <iostream>
using namespace std;

int main(){
  int N,Q;
  cin>>N;
  vector<int>lista(N);
  for(int i=0;i<N;i++)cin>>lista[i];
  vector<int>ps(N,0);
  ps[0]=lista[0];
  for(int i=1;i<N;i++)ps[i]=ps[i-1]+lista[i];
  cin>>Q;
  while(Q--){
    int a,b;
    cin>>a>>b;
    if(a==0)cout<<ps[b]<<"\n";
    else cout<<ps[b]-ps[a-1]<<"\n";
  }
  return 0;
}
```

Alicia y las llaves doradas de las puertas

Teoría

- búsqueda binaria

Solución

Para cada valor de llave hacemos una busqueda binaria sobre la lista inicial ordenada. Devolvemos la posición+1 si encontramos el valor correspondiente en la lista y 0 en caso contrario.

Link al problema

Código

```cpp
#include <iostream>
#include <utility>
#include <algorithm>
#include <vector>
using namespace std;
int busquedabinaria(vector<int>&lista,int valor){
  int low=0,hi=lista.size(),medio;
```

```
  while(low<=hi){
    medio=low+(hi-low)/2;
    if(lista[medio]==valor){
      return medio+1;
    }
    else if(lista[medio]<valor){
      low=medio+1;
    }
    else{
      hi=medio-1;
    }
  }
  return 0;
}
int main ()
{ int n,k,l;
  cin>>n;
  vector<int>lista(n);
  for(int i=0;i<n;i++)cin>>lista[i];
  sort(lista.begin(),lista.end());
  cin>>k;
  for(int i=0;i<k;i++){
  cin>>l;
  cout<<busquedabinaria(lista,l)<<" ";
  }
  return 0;
}
```

Un problema facil 2

Teoría

- búsqueda binaria

Solución

Ordenamos la lista y tenemos un contador,que inicializamos en 1. Este contador nos dirá cuantas parejas tenemos de tal manera que su diferencia sea k. Pasamos la lista y en cada iteración haremos una búsqueda binaria en la que comprobarmenos si para el elemento lista[i] hay otro elemento de tal manera, que la diferencia sea k. Si es así sumaremos uno, si no es así sumaremos cero.

Link al problema

Código

```
#include <iostream>
#include <utility>
#include <algorithm>
#include <vector>
using namespace std;
int busquedabinaria(vector<int>&lista,int valor){
  int low=0,hi=lista.size(),medio;
  while(low<=hi){
    medio=low+(hi-low)/2;
    if(lista[medio]==valor){
      return 1;//la diferencia es 1, así que hemos encontado una pareja para la
      ↪  que se cumple
    }
    else if(lista[medio]<valor){
      low=medio+1;
    }
    else{
      hi=medio-1;
    }
  }
  return 0;
}
int main ()
{ int n,k;
  cin>>n>>k;
  vector<int>lista(n);
  for(int i=0;i<n;i++)cin>>lista[i];
  sort(lista.begin(),lista.end());
  int contador=0;
  for(int i=0;i<n;i++)
  contador+=busquedabinaria(lista,k+lista[i]);
  cout<<contador;
  return 0;
}
```

Los libros de Santy 2

Teoría

- búsqueda binaria

Solución

Ordenamos el vector de los libros y hacemos una búsqueda binaria como la que hemos visto en clase.

Link al problema

Código

```cpp
#include <cmath>
#include <cstdio>
#include <vector>
#include <iostream>
#include <algorithm>
#include <utility>
using namespace std;
int buscar(vector<long long int>&libros,long long int B){
  int low=0,hi=libros.size()-1,mid;
  while(low<=hi){
    mid=low+(hi-low)/2;
    if(libros[mid]==B)return mid;
    else if(libros[mid]>B)hi=mid-1;
    else low=mid+1;
  }
}
int main(){
  ios::sync_with_stdio(false);
  cin.tie(NULL);
  int N,Q;
  cin>>N>>Q;
  vector<long long int>libros(N);
  for(int i=0;i<N;i++)cin>>libros[i];
  sort(libros.begin(),libros.end());
  while(Q--){
    long long int B;
    cin>>B;
    cout<<buscar(libros,B)<<" ";
  }
  return 0;
}
```

Chocolates CIIC 2015

Teoría

- Bisección
- suma de prefijos

Solución

Usamos el método de bisección para encontrar el máximo número posible de chocolates que se puede comer al día. Para esto usaremos una función posible: Usamos un vector de sumas de prefijos (un vector, que va sumando todos los valores de c hasta el momento [linea 50]).

Código

```
#include <cmath>
#include <cstdio>
#include <vector>
#include <iostream>
#include <algorithm>
#include <utility>
#include <iomanip>
using namespace std;

bool posible(vector<long long int>&ps,int chocolatesdiarios){
  bool pos=true;
  int n=ps.size();
  long long int contador=chocolatesdiarios;
  for(int i=0;i<n;i++){
    if(ps[i]<contador){
      pos=false;
      break;
    }
    contador+=chocolatesdiarios;
  }
  return pos;
}
int buscar(vector<long long int>&ps){
  int n=ps.size();
  int low=0,hi=ps[n-1]/n,r=-1,mid;
  while(low<=hi){
    mid=low+(hi-low)/2;
    if(posible(ps,mid)){
      r=mid;
      low=mid+1;
    }
    else{
      hi=mid-1;
    }
  }
  return r;
}

```

```
int main(){
  ios::sync_with_stdio(false);
  cin.tie(NULL);
  int n;
  cin>>n;
  vector<int>c(n);
  for(int i=0;i<n;i++){
    cin>>c[i];
  }
  vector<long long int>ps(n);
  ps[0]=c[0];
  for(int i=1;i<n;i++)ps[i]=ps[i-1]+c[i];//suma de prefijos
  cout<<buscar(ps);
  return 0;
}
```

Minimum Number of Days to Make m Bouquets

Teoría

- Bisección

Solución

Usaremos el método de bisección para encontrar el mínimo día para el que es posible formar tal ramo de flores. El método de bisección es calcado al de clase y cambiamos la función de bool posible(). En esta función pasaremos todo el vector, contando primero cuantas flores adyacentes podemos coger y los ramos que podeos formar. Una vez tenemos suficientes flores para un ramo, sumas uno al contador de ramos y reiniciamos el contador de flores adyacentes. Este último también lo volveremos a cero en cuanto no podamos coger la flor. Por último miramos si el contador de ramos es igual o mayor a los ramos que tenemos que formar. Si esto es cierto devolvemos true, si no devolvemos false.

Link al problema

Código

```
 bool pos(int x,vector<int>& b,int m,int k){// nuestra función que comprueba si es
  ↪  posible
int n=b.size();
int contador1=0,contador2=0;// contador1: cuenta cuantas flores adyacentes
 ↪  llevamos, contador2:cuenta cuantos ramos llevamos
for(int i=0;i<n;i++){//pasamos el vector
if(contador2==m)break;//ya tenemos lo que necesitamos, así que no tenemos que
 ↪  seguir buscando
if(b[i]<=x){
  contador1++;
  if(contador1==k){
    contador1=0;// lo ponemos a 0 una vez formamos un ramo y sumamos 1 al
     ↪  contador2
    contador2++;
```

```
    }
  }
  else contador1=0;// si todavía no ha florecido la flor "reinciiamos" el contador1
  }
  return contador2>=m;// si tenemos m o más ramos es posible conseguir nuestro
   ↪  objetivo en x días y la función devuleve true, en caso contrario devuelve
   ↪  false
  }
  int minDays(vector<int>& bloomDay, int m, int k) {
          int n=bloomDay.size();
          int maxi=0;
          for(int i=0;i<n;i++){
            maxi=max(maxi,bloomDay[i]);
          }
          if(n<m*k)return -1; //si no hay suficientes flores es imposible
          else if(n==m*k)return maxi;// si hay justo las flores, que piden, entonces
           ↪  devolveremos el último día
          else{
            int low=0,hi=maxi,mid,r=-1;// aquí hacemos la busqueda binaria vista en
             ↪  clase
            while(low<=hi){
              mid=low+(hi-low)/2;
              if(pos(mid,bloomDay,m,k)){
                hi=mid-1;
                r=mid;
              }
              else low=mid+1;
            }
            return r;
          }
      }
```

Baq la calculadora

Teoría

- búsqueda binaria
- suma de prefijos modificada

Solución

Primero meteremos los valores de canicas en un vector y lo ordenamos. Creamos un segundo vector mc, parecido al de suma de prefijos, pero ahora con multiplicaciones en modulo 1e9+7. (Breve explicación de módulos Apuntes Entrenamiento 5 OIFem I págs.1-2). Una vez tenemos eso nos vale buscar con búsqueda binaria el índice x del valor menor/igual al que nos da Anna en el vector de las canicas y devolver mc[x].

Link al problema

Código

```cpp
#include <cmath>
#include <cstdio>
#include <vector>
#include <iostream>
#include <algorithm>
#include <utility>
#include <queue>
#include <map>
#include <iomanip>
#include <stack>
using namespace std;
typedef long long int ll;
int mod=1e9+7;

int buscar(vector<int>&c,int q){
  int low=0,hi=c.size(),m,r=-1;
  while(low<=hi){
    m=low+(hi-low)/2;
    if(c[m]<=q){
      low=m+1;
      r=m;
    }
    else{
      hi=m-1;
    }
  }
  return r;
}

int main(){
  ios::sync_with_stdio(false);
  cin.tie(NULL);
  int t,n,q,a;
  cin>>t;
```

```
35    while(t--){
36      cin>>n;
37      vector<int>canicas(n);
38      for(int i=0;i<n;i++){
39        cin>>canicas[i];//leemos los valores de las canicas
40      }
41      vector<ll>mc(n);
42      sort(canicas.begin(),canicas.end());//ordenamos los valores de las canicas
43      mc[0]=canicas[0]%mod;//mc es como un vector de suma de prefijos, pero con la
        ↪  multiplicación modulo 1e9+7
44      for(int i=1;i<n;i++){
45        mc[i]=mc[i-1]*canicas[i]%mod;
46        mc[i]=mc[i]%mod;
47      }
48
49      cin>>q;
50      for(int i=0;i<q;i++){
51        cin>>a;
52        if(a>=canicas[n-1]){
53          cout<<mc[n-1]<<"\n";
54        }
55        else{
56        int x=buscar(canicas,a);//un binary search como el visto en clase
57        if(x==-1){
58          cout<<1<<"\n";
59        }
60        else{
61           cout<<mc[x]<<"\n";
62        }
63        }
64      }
65
66
67    }
68    return 0;
69  }
```