Apuntes OIFem II Nivel 1

Búsqueda binaria

Problema base: Tenemos un vector ordenado de menor a mayor y queremos comprobar si un número está incluido o no y en caso de que esté incluido saber el valor. **Algoritmo intuitivo, pero ineficiente:**: La primera idea, que se nos viene a la cabeza es pasar el vector elemento a elemento y buscar el número que queremos encontrar.

Algoritmo optimo: BINARY SEARCH

Tenemos el vector lista de tamaño n y vamos a usar dos variables de ayuda low y hi para encontrar al número x. Low guardará en todo momento el menor valor posible no explorado y hi el mayor valor posible no explorado. Es decir al principio low=0 y hi=n-1, ya que no hemos visto nigún valor todavía. Por último tenemos una variable mitad m= low+(hi-low)/2 , siempre será la mitad entre low y hi.

1. Miramos si lista[m]==x entonces devolvemos m y hemos terminado.

2. Si no es el caso miramos si lista[m] es menor que x, esto significa que todos los valores de 0 a m son menores que x, ya que la lista está ordenada, y por tanto no nos interesan. Podemos por tanto cambiar low=m+1.

3. Si no es el caso lista[m] es mayor que x, esto significa que todos los valores de m a n son mayores que x, ya que la lista está ordenada, y por tanto no nos interesan. Podemos por tanto cambiar hi=m-1y así descartamos ese tramo.

4. Si low es mayor que hi, hemos "descartado" todos los valores, es decir x no está en la lisa y devolvemos -1

Como vamos descartando nos "ahorramos" comprobar muchos números y acaba siendo una técnica mucha más eficiente que la lineal. **Link a la animación**

Código

```
int busquedabinaria(vector<int>&lista,int x){
     int n=lista.size();
     int low=0,hi=n-1,m;
     while(low<=hi){//si low>hi, es que ya hemos descartado todos los valores
        m=low+(hi-low)/2;
        if(lista[m]==x)return m;
        else if(lista[m]<x)low=m+1;
        else hi=m-1
     }
     return -1;
```

```
}
```

Método de bisección

Muchos problemas nos van a pedir el mínimo o el máximo a partir del que se cumple una condición. Es decir estamos buscando un número x y sabemos que x tiene que estar entre un low y un hi. Ejemplo para el mínimo: Vamos ha hacer una búsqueda binaria personalizada, para cada mid miraremos si se cumple la condición y si se cumple descartamos todos los valores por encima y nos guardamos ese como el mínimo hasta el momento, en caso contrario todos los menores.

Ejemplo de problema:278. First Bad Version

Usted es director de una empresa, que diseña productos, y actualmente dirige un equipo para desarrollar un nuevo producto. Por desgracia, la última versión de su producto no pasa el control de calidad. Dado que cada versión se desarrolla sobre la base de la versión anterior, todas las versiones posteriores a una versión mala son también malas.

Supongamos que tienes n versiones [1, 2, ..., n] y quieres averiguar cuál es la primera mala, que hace que todas las siguientes sean malas.

Se le da una API bool isBadVersion(version) que devuelve si la versión es mala. Implementa una función para encontrar la primera versión mala. Debes minimizar el número de llamadas a la API.

Link al problema

Código:

```
int firstBadVersion(int n) {
      int low=1,hi=n,r=1,m;
        while(low<=hi){
            m=low+(hi-low)/2;
            if(isBadVersion(m)){//aquí nos dan la función ya hecha, pero
             ↪  normalemente tendremos que programarla nosotros
                r=m;
                hi=m-1;
            }
            else{
                low=m+1;
            }
        }
        return r;
    }
```

Suma de prefijos

En muchas ocasiones esta nueva técnica nos va a resultar muy útil para resolver problemas de bisección.

Problema:

Tenemos un vector y queremos saber para cada uno de los valores del vector cuanto suman los todos los índices que van antes de él y el. Podríamos calcularlo para cada uno, pero se nos quedaría en una solución cuadrática, es decir muy ineficiente. En cambio podemos incluir un nuevo vector llamado suma de prefijos, acortado como ps. Recorremos el vector y decimos ps[i]=ps[i-1]+lista[i]. De esta manera en cada posición del vector ps tendremos la solución a nuestro problema inicial.

Con este vector también podemos ver cuanto vale la suma desde el indice a al indice b: suma=ps[b]-ps[a-1]

Código- ps:

```cpp
#include <cmath>
#include <cstdio>
#include <vector>
#include <iostream>
#include <algorithm>
#include <utility>
using namespace std;
int main(){
  int n;
  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];// el primer elmento es igual, no podemos hacer
  ↪  ps[0]=ps[-1]+lista[0],ya que nos daría error
  for(int i=1;i<n;i++)ps[i]=ps[i-1]+lista[i];
  for(int i=0;i<n;i++)cout<<ps[i]<<" ";
  int a,b; // intervalo del que queremos saber la suma
  cin>>a>>b;
  if(a==0)cout<<ps[b];
  else cout<<ps[b]-ps[a-1];
  return 0;
}
```

Problema: Bisección y ps: Informados de las medidas de higiene

Al final de cada uno de los N largos días de la contingencia, Susana Distancia anota minuciosamente a cuántas nuevas personas ha informado acerca de las medidas de higiene necesarias. Esto lo hace por si alguien le pregunta ¿cuál fue el primer día en el que hubo un total de al menos T personas informadas? lo cual, por cierto, le preguntarán M veces. Ayuda a Susana Distancia a prepararse para esas preguntas.

Solución:

Creamos un ps con el número de personas que informa cada día y luego hacemos binary search sobre él.

Link al problema

Código

```cpp
#include <cmath>
#include <cstdio>
```

```cpp
#include <vector>
#include <iostream>
#include <algorithm>
#include <utility>
using namespace std;
int buscar(vector<int>&ps,int T){
  int low=0,hi=ps.size()-1,r=-1,mid;
  while(low<=hi){
    mid=low+(hi-low)/2;
    if(ps[mid]>=T){
      hi=mid-1;
      r=mid;
    }
    else low=mid+1;
  }
  return r+1;
}
int main(){
  int N,M;
  cin>>N>>M;
  vector<int>personas(N);
  vector<int>ps(N,0);
  for(int i=0;i<N;i++)cin>>personas[i];
  ps[0]=personas[0];
  for(int i=1;i<N;i++)ps[i]=ps[i-1]+personas[i];
  while(M--){
    int T;
    cin>>T;
    cout<<buscar(ps,T)<<" ";
  }
  return 0;
}
```