Búsqueda Binaria

Leopoldo Taravilse[1] Juan Cruz Piñero[2]

[1]Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires
[2]Facultad de Informática
Universidad Nacional del Comahue

Training Camp 2017

Contenidos

Contenidos

Buscando en listas ordenadas

- Uno de los problemas más comunes que existen es el de querer buscar si un número aparece o no en una lista ordenada.

Buscando en listas ordenadas

- Uno de los problemas más comunes que existen es el de querer buscar si un número aparece o no en una lista ordenada.
- Es trivial ver que se puede resolver el problema en $O(n)$ donde n es la cantidad de elementos de la lista, simplemente buscando uno por uno en orden.

Buscando en listas ordenadas

- Uno de los problemas más comunes que existen es el de querer buscar si un número aparece o no en una lista ordenada.
- Es trivial ver que se puede resolver el problema en $O(n)$ donde n es la cantidad de elementos de la lista, simplemente buscando uno por uno en orden.
- Hay una forma de aprovechar el hecho de que el arreglo está ordenado. Esta forma de buscar eficientemente se conoce como búsqueda binaria.

Ejemplo de Búsqueda Binaria

Supongamos que queremos buscar si el número 22 está en el siguiente arreglo:

```
1 2 4 6 8 14 15 16 21 22 31 35 44 45 56 87 89 95 99 100 103 112 128
```

Ejemplo de Búsqueda Binaria

Supongamos que queremos buscar si el número 22 está en el siguiente arreglo:

```
1 2 4 6 8 14 15 16 21 22 31 35 44 45 56 87 89 95 99 100 103 112 128
```

Podemos preguntarnos ¿Es el primer elemento del arreglo el 22? No, ¿Es el segundo elemento del arreglo el 22? Tampoco... y así hasta que nos preguntamos ¿Es el décimo elemento del arreglo el 22? Sí!

Ejemplo de Búsqueda Binaria

Con búsqueda binaria podemos buscar así

1 2 4 6 8 14 15 16 21 22 31 **35** 44 45 56 87 89 95 99 100 103 112 128 .

1 2 4 6 8 **14** 15 16 21 22 31 **35** 44 45 56 87 89 95 99 100 103 112 128

1 2 4 6 8 **14** 15 16 **21** 22 31 **35** 44 45 56 87 89 95 99 100 103 112 128

1 2 4 6 8 14 15 16 **21** **22** 31 **35** 44 45 56 87 89 95 99 100 103 112 128

Ejemplo de Búsqueda Binaria

Con búsqueda binaria podemos buscar así

1 2 4 6 8 14 15 16 21 22 31 **35** 44 45 56 87 89 95 99 100 103 112 128 .

1 2 4 6 8 **14** 15 16 21 22 31 **35** 44 45 56 87 89 95 99 100 103 112 128

1 2 4 6 8 **14** 15 16 **21** 22 31 **35** 44 45 56 87 89 95 99 100 103 112 128

1 2 4 6 8 14 15 16 **21** **22** 31 **35** 44 45 56 87 89 95 99 100 103 112 128

Ejemplo de Búsqueda Binaria

Con búsqueda binaria podemos buscar así

1 2 4 6 8 14 15 16 21 22 31 **35** 44 45 56 87 89 95 99 100 103 112 128 .

1 2 4 6 8 **14** 15 16 21 22 31 **35** 44 45 56 87 89 95 99 100 103 112 128

1 2 4 6 8 **14** 15 16 **21** 22 31 **35** 44 45 56 87 89 95 99 100 103 112 128

1 2 4 6 8 14 15 16 **21** **22** 31 **35** 44 45 56 87 89 95 99 100 103 112 128

Ejemplo de Búsqueda Binaria

Con búsqueda binaria podemos buscar así

1 2 4 6 8 14 15 16 21 22 31 **35** 44 45 56 87 89 95 99 100 103 112 128 .

1 2 4 6 8 **14** 15 16 21 22 31 **35** 44 45 56 87 89 95 99 100 103 112 128

1 2 4 6 8 **14** 15 16 **21** 22 31 **35** 44 45 56 87 89 95 99 100 103 112 128

1 2 4 6 8 14 15 16 **21** **22** 31 **35** 44 45 56 87 89 95 99 100 103 112 128

Ejemplo de Búsqueda Binaria

Con búsqueda binaria podemos buscar así

1 2 4 6 8 14 15 16 21 22 31 **35** 44 45 56 87 89 95 99 100 103 112 128 .

1 2 4 6 8 **14** 15 16 21 22 31 **35** 44 45 56 87 89 95 99 100 103 112 128

1 2 4 6 8 **14** 15 16 **21** 22 31 **35** 44 45 56 87 89 95 99 100 103 112 128

1 2 4 6 8 14 15 16 **21** **22** 31 **35** 44 45 56 87 89 95 99 100 103 112 128

Notemos que en 4 pasos pudimos encontrar el 22 cuando antes nos llevaba 10 pasos.

Código de la búsqueda binaria

```
busqueda binaria(min,max,v)
    while(max-min > 1)
        med = (max+min)/2
        if(f(med,v))
            max = med
        else
            min = med
    return min
```

Código de la búsqueda binaria

```
busqueda binaria(min,max,v)
    while(max-min > 1)
        med = (max+min)/2
        if(f(med,v))
            max = med
        else
            min = med
    return min
```

f es la función que evaluamos y comparamos con *v* para saber si es mayor o menor igual.
El hecho de retornar min depende del caso, a veces queremos retornar max.
Nuestras hipótesis serán que *f* es creciente en $[min, max)$ y que *v* está ensanguchado entre $f(min)$ y $f(max)$.

Ejemplo de Búsqueda Binaria

Buscar un número que no está es mucho más costoso con búsqueda lineal, sin embargo con búsqueda binaria es casi tan rápido como buscar un número que si está. Busquemos por ejemplo el 23.

1 2 4 6 8 14 15 16 21 22 31 **35** 44 45 56 87 89 95 99 100 103 112 128 .

1 2 4 6 8 **14** 15 16 21 22 31 **35** 44 45 56 87 89 95 99 100 103 112 128

1 2 4 6 8 **14** 15 16 **21** 22 31 **35** 44 45 56 87 89 95 99 100 103 112 128

1 2 4 6 8 14 15 16 **21** **22** 31 **35** 44 45 56 87 89 95 99 100 103 112 128

1 2 4 6 8 14 15 16 21 **22** **31** **35** 44 45 56 87 89 95 99 100 103 112 128

1 2 4 6 8 14 15 16 21 **22** **31** 35 44 45 56 87 89 95 99 100 103 112 128

Ejemplo de Búsqueda Binaria

Buscar un número que no está es mucho más costoso con búsqueda lineal, sin embargo con búsqueda binaria es casi tan rápido como buscar un número que si está. Busquemos por ejemplo el 23.

1 2 4 6 8 14 15 16 21 22 31 **35** 44 45 56 87 89 95 99 100 103 112 128 .

1 2 4 6 8 **14** 15 16 21 22 31 **35** 44 45 56 87 89 95 99 100 103 112 128

1 2 4 6 8 **14** 15 16 **21** 22 31 **35** 44 45 56 87 89 95 99 100 103 112 128

1 2 4 6 8 14 15 16 **21** **22** 31 **35** 44 45 56 87 89 95 99 100 103 112 128

1 2 4 6 8 14 15 16 21 **22** **31** **35** 44 45 56 87 89 95 99 100 103 112 128

1 2 4 6 8 14 15 16 21 **22** **31** 35 44 45 56 87 89 95 99 100 103 112 128

Ejemplo de Búsqueda Binaria

Buscar un número que no está es mucho más costoso con búsqueda lineal, sin embargo con búsqueda binaria es casi tan rápido como buscar un número que si está. Busquemos por ejemplo el 23.

1 2 4 6 8 14 15 16 21 22 31 **35** 44 45 56 87 89 95 99 100 103 112 128 .

1 2 4 6 8 **14** 15 16 21 22 31 **35** 44 45 56 87 89 95 99 100 103 112 128

1 2 4 6 8 **14** 15 16 **21** 22 31 **35** 44 45 56 87 89 95 99 100 103 112 128

1 2 4 6 8 14 15 16 **21** **22** 31 **35** 44 45 56 87 89 95 99 100 103 112 128

1 2 4 6 8 14 15 16 21 **22** **31** **35** 44 45 56 87 89 95 99 100 103 112 128

1 2 4 6 8 14 15 16 21 **22** **31** 35 44 45 56 87 89 95 99 100 103 112 128

Ejemplo de Búsqueda Binaria

Buscar un número que no está es mucho más costoso con búsqueda lineal, sin embargo con búsqueda binaria es casi tan rápido como buscar un número que si está. Busquemos por ejemplo el 23.

1 2 4 6 8 14 15 16 21 22 31 **35** 44 45 56 87 89 95 99 100 103 112 128 .

1 2 4 6 8 **14** 15 16 21 22 31 **35** 44 45 56 87 89 95 99 100 103 112 128

1 2 4 6 8 **14** 15 16 **21** 22 31 **35** 44 45 56 87 89 95 99 100 103 112 128

1 2 4 6 8 14 15 16 **21** **22** 31 **35** 44 45 56 87 89 95 99 100 103 112 128

1 2 4 6 8 14 15 16 21 **22** **31** **35** 44 45 56 87 89 95 99 100 103 112 128

1 2 4 6 8 14 15 16 21 **22** **31** 35 44 45 56 87 89 95 99 100 103 112 128

Ejemplo de Búsqueda Binaria

Buscar un número que no está es mucho más costoso con búsqueda lineal, sin embargo con búsqueda binaria es casi tan rápido como buscar un número que si está. Busquemos por ejemplo el 23.

1 2 4 6 8 14 15 16 21 22 31 **35** 44 45 56 87 89 95 99 100 103 112 128 .

1 2 4 6 8 **14** 15 16 21 22 31 **35** 44 45 56 87 89 95 99 100 103 112 128

1 2 4 6 8 **14** 15 16 **21** 22 31 **35** 44 45 56 87 89 95 99 100 103 112 128

1 2 4 6 8 14 15 16 **21** **22** 31 **35** 44 45 56 87 89 95 99 100 103 112 128

1 2 4 6 8 14 15 16 21 **22** **31** **35** 44 45 56 87 89 95 99 100 103 112 128

1 2 4 6 8 14 15 16 21 **22** **31** 35 44 45 56 87 89 95 99 100 103 112 128

Ejemplo de Búsqueda Binaria

Buscar un número que no está es mucho más costoso con búsqueda lineal, sin embargo con búsqueda binaria es casi tan rápido como buscar un número que si está. Busquemos por ejemplo el 23.

1 2 4 6 8 14 15 16 21 22 31 **35** 44 45 56 87 89 95 99 100 103 112 128 .

1 2 4 6 8 **14** 15 16 21 22 31 **35** 44 45 56 87 89 95 99 100 103 112 128

1 2 4 6 8 **14** 15 16 **21** 22 31 **35** 44 45 56 87 89 95 99 100 103 112 128

1 2 4 6 8 14 15 16 **21** **22** 31 **35** 44 45 56 87 89 95 99 100 103 112 128

1 2 4 6 8 14 15 16 21 **22** **31** **35** 44 45 56 87 89 95 99 100 103 112 128

1 2 4 6 8 14 15 16 21 **22** **31** 35 44 45 56 87 89 95 99 100 103 112 128

Complejidad de la búsqueda binaria

La complejidad de la búsqueda binaria es $O(\log n)$ donde n es el tamño del arreglo. Esto es fácil de probar ya que en cada paso se reduce el espacio de búsqueda a la mitad.

Contenidos

Funciones monótonas

Cuando tenemos una función f que cumple

$$x > y \Rightarrow f(x) \geq f(y)$$

decimos que f es monótona creciente. Si en cambio f cumple

$$x > y \Rightarrow f(x) \leq f(y)$$

decimos que f es monótona decreciente. Si f es monótona creciente o monótona decreciente decimos que f es monótona.
Cuando una función es monótona creciente (decreciente) podemos aplicar búsqueda binaria para buscar el menor valor de x tal que $f(x) \geq y$ ($f(x) \leq y$) para un y dado.

Ejemplo de búsqueda binaria con funciones monótonas

Pedro estudia Ciencias de la Computación y le quedan por cursar n materias. Él sabe que si hace una sóla materia le va a costar 1 unidad de esfuerzo aprobarla, pero si hace una segunda materia le cuesta 2 unidades de esfuerzo para aprobar la segunda materia, y en general, si hace i materias le cuesta i unidades de esfuerzo aprobar la i-ésima materia.
El sabe que hacer un esfuerzo de t unidades o más lo va a estrezar y por lo tanto va a dejar la carrera. Cuántos cuatrimestres necesita para recibirse?

Ejemplo de búsqueda binaria con funciones monótonas

- Lo primero que tenemos que hacer es calcular cuántas materias puede hacer por cuatrimestre. Luego es una división.

Ejemplo de búsqueda binaria con funciones monótonas

- Lo primero que tenemos que hacer es calcular cuántas materias puede hacer por cuatrimestre. Luego es una división.
- Para calcular cuántas materias puede hacer por cuatrimestre, sabemos que ese número x cumple con $\frac{x(x+1)}{2} < t$. Luego hacemos búsqueda binaria para encontrar ese máximo valor posible de x.

Contenidos

Búsqueda binaria continua

- Así como usamos búsqueda binaria cuando tenemos un dominio discreto, también podemos usar búsqueda binaria cuando el dominio es continuo.

Búsqueda binaria continua

- Así como usamos búsqueda binaria cuando tenemos un dominio discreto, también podemos usar búsqueda binaria cuando el dominio es continuo.
- Por ejemplo, tenemos una función f monótona creciente y continua y queremos hallar el mínimo x tal que $f(x) \geq y$.

Búsqueda binaria continua

- Así como usamos búsqueda binaria cuando tenemos un dominio discreto, también podemos usar búsqueda binaria cuando el dominio es continuo.
- Por ejemplo, tenemos una función f monótona creciente y continua y queremos hallar el mínimo x tal que $f(x) \geq y$.
- Como f es continua, existe un x tal que $f(x) = y$, luego queremos encontrar $f^{-1}(y)$.

Búsqueda binaria continua

- Así como usamos búsqueda binaria cuando tenemos un dominio discreto, también podemos usar búsqueda binaria cuando el dominio es continuo.
- Por ejemplo, tenemos una función f monótona creciente y continua y queremos hallar el mínimo x tal que $f(x) \geq y$.
- Como f es continua, existe un x tal que $f(x) = y$, luego queremos encontrar $f^{-1}(y)$.
- Si existen varias preimágenes de y con búsqueda binaria podemos encontrar la más chica (o la más grande) de todas.

Ejemplo de búsqueda binaria para encontrar el inverso de una función

Dado un número x, calcular la raiz cuadrada de x

Ejemplo de búsqueda binaria para encontrar el inverso de una función

En este caso la busqueda binaria, al ser continua, debe terminar cuando los extremos del intervalo son muy parecido y no cuando distan en 1

```
f(int t, int x)
    return t > x*x

busqueda binaria(min,max,x)
    while(max-min > epsilon)
        med = (max+min)/2
        if(f(med,x))
            max = med
        else
            min = med
    return min
```

Cálculo de la raiz cuadrada

- A diferencia del caso de la búsqueda binaria discreta, cuando el dominio de la búsqueda binaria es continuo tenemos que poner un punto de corte arbitrario para la búsqueda binaria.

Cálculo de la raiz cuadrada

- A diferencia del caso de la búsqueda binaria discreta, cuando el dominio de la búsqueda binaria es continuo tenemos que poner un punto de corte arbitrario para la búsqueda binaria.
- En este caso podemos por ejemplo elegir $\epsilon = 10^{-9}$ ya que es un valor tan chico que es casi despreciable.

Cálculo de la raiz cuadrada

- A diferencia del caso de la búsqueda binaria discreta, cuando el dominio de la búsqueda binaria es continuo tenemos que poner un punto de corte arbitrario para la búsqueda binaria.
- En este caso podemos por ejemplo elegir $\epsilon = 10^{-9}$ ya que es un valor tan chico que es casi despreciable.
- Cuando la búsqueda termina, el intervalo $[min, max)$ es tan chico que cualquier valor en el intervalo es la solución con un error a lo sumo muy pequeño. Por esta misma razón comparar por $>$ o por $\geq$ en f es lo mismo.

Contenidos

Búsqueda binaria con predicados booleanos

- Ya vimos como hace búsqueda binaria sobre funciones que tienen una imagen numérica y que son monotonas.

Búsqueda binaria con predicados booleanos

- Ya vimos como hace búsqueda binaria sobre funciones que tienen una imagen numérica y que son monotonas.
- Cuando la imagen de la función es booleana, por ejemplo, dado un predicado booleano P, encontrar el mínimo x tal que $P(x)$ es verdadero, también podemos definir monotonía y aplicar búsqueda binaria.

Búsqueda binaria con predicados booleanos

- Ya vimos como hace búsqueda binaria sobre funciones que tienen una imagen numérica y que son monotonas.
- Cuando la imagen de la función es booleana, por ejemplo, dado un predicado booleano P, encontrar el mínimo x tal que $P(x)$ es verdadero, también podemos definir monotonía y aplicar búsqueda binaria.
- Cuando un predicado es falso para todos los $x < x_0$ y verdadero para los $x \geq x_0$ decimos que el predicado monótono. Esto equivale con asignarle 0 a falso y 1 a verdadero, en este caso el predicado es monótono creciente.

Búsqueda binaria con predicados booleanos

Podemos ver fácilmente que aplicar búsqueda binaria para encontrar mínimos o máximos es como aplicar búsqueda binaria sobre el predicado $P(x) := f(x) <= y$

Ejemplo de búsqueda binaria con predicados booleanos

Problema

Fito tiene que cruzar un cuarto lleno de dinosaurios desde la esquina superior izquierda hasta la esquina inferior derecha. Son dadas las posiciones de los dinosaurios y las dimensiones del cuarto. Se sabe que si pasa a distancia menor a T de un dinosaurio, este lo ve y se lo come. Cuál es el mínimo T tal que Fito puede lograr su objetivo?

¿Qué pasa si la función no es monótona?

- Si la función f no es monótona no vamos a poder encontrar el mínimo valor que cumple con $f(x) = y$, pero si sabemos que en un rango hay un valor tal que $f(x) = y$ vamos a poder encontrar uno de ellos.
- La idea es que si tenemos un intervalo $[min, max)$ que cumple con $f(min) \leq y < f(max)$ en cada paso reemplazamos min o max por $\frac{min+max}{2}$ de modo tal de que esa desigualdad se siga cumpliendo, y como f es continua usamos el famoso teorema que tanto le gusta a Agustín (Bolzano).

Contenidos

¿Qué es búsqueda ternaria?

Supongamos que queremos buscar un mínimo o un máximo en una función. Si la función f es continua lo que queremos hacer es buscar un x tal que $f'(x) = 0$, y si la derivada es continua podemos hacer búsqueda binaria en la derivada.
A veces calcular la derivada de una función no es fácil, y tal vez la función no está explicita para poder calcularla, o el dominio de la misma no es continuo (por ejemplo si $f : \mathbb{N} \to \mathbb{N}$).
Si queremos buscar por ejemplo un máximo en un intervalo y sabemos que la función es continua y está acotada, nos alcanza con ver que en algún momento crece y después decrece, y por lo tanto en el medio tenemos un máximo.

Aplicando búsqueda ternaria

En este caso asumimos que hay un máximo en el intervalo $[min, max)$ y lo queremos encontrar, y además vamos a asumir que la función en *min* empieza creciendo hasta que en algún momento empieza a decrecer. Para esto alcanza con mantener el invariante de que existe $min < med < max$ tal que $f(med) > f(min)$ y $f(med) > f(max)$.

```
busqueda ternaria(min,max)
    while(max-min > epsilon)
        med1 = (max+2*min)/3
        med2 = (2*max+min)/3
        if(f(med1) > f(med2))
            max = med2
        else
            min = med1
    return min
```

Aplicando búsqueda ternaria

Podemos observar que como f empieza creciendo en min y decreciendo hasta max, si $f(med1) > f(med2)$ quiere decir que antes de $med2$ la función ya empezó a decrecer, entonces sigue valiendo la hipótesis en nuestro rango.

```
busqueda ternaria(min,max)
    while(max-min > epsilon)
        med1 = (max+2*min)/3
        med2 = (2*max+min)/3
        if(f(med1) > f(med2))
            max = med2
        else
            min = med1
    return min
```

En este caso el espacio de búsqueda se multiplica por $\frac{2}{3}$ en cada paso.