

Búsqueda

Clase #21

IIC1103 – Introducción a la Programación

Marcos Sepúlveda (marcos@ing.puc.cl)

Veremos hoy ...

- ▶ Búsqueda en listas
- ▶ Búsqueda en listas de objetos

Búsqueda

- ▶ La búsqueda es una de las operaciones más importantes en el procesamiento de la información, y permite la recuperación de datos previamente almacenados.
- ▶ Veremos 2 métodos:
 - Búsqueda secuencial
 - Búsqueda binaria




Búsqueda secuencial

► Veamos esto con un ejemplo.

- Supongamos que se tiene una lista de 10 elementos enteros:

L	31	6	12	762	45	34	87	56	1	86
posición	0	1	2	3	4	5	6	7	8	9



- Si se quiere buscar el número **87**, entonces hay que recorrer la lista **L** desde la posición 0 hasta la posición donde se encuentre el elemento buscado. Una vez encontrado el elemento, no es necesario seguir visitando los elementos restantes de la lista, por lo que se debe terminar la búsqueda.
- Si no existe, nos daremos cuenta al llegar al final.
- Si el elemento existe, basta con: **L.index(87)**

Búsqueda binaria

- ▶ Requiere que la lista esté **ordenada**.
- ▶ El algoritmo de búsqueda binaria va dividiendo la lista en dos, quedándose con la sub-lista donde podría estar el elemento buscado.
 - Calcula la posición central de la lista inicial
 - Compara el elemento buscado con el elemento que ocupa la posición central. Si es igual entonces devuelve dicha posición. Si no, según sea mayor o menor que el elemento central, se descarta una mitad de la lista y se visita la otra mitad, donde debería estar el elemento buscado.

Búsqueda binaria

► Algoritmo:

1. Se definen los índices inferior y superior.

```
inferior = 0
```

```
superior = len(lista) - 1
```

2. Se calcula el centro de la lista como:

```
centro = (inferior + superior) / 2
```

3. Verificamos si la lista en la posición **centro** es igual al dato que buscamos. Si es igual, significa que encontramos el dato y retornamos **centro**.

4. Si es diferente, verificamos si la lista en la posición **centro** es menor o mayor al dato que queremos buscar.

- Si es menor actualizamos **inferior**:

```
inferior = centro + 1
```

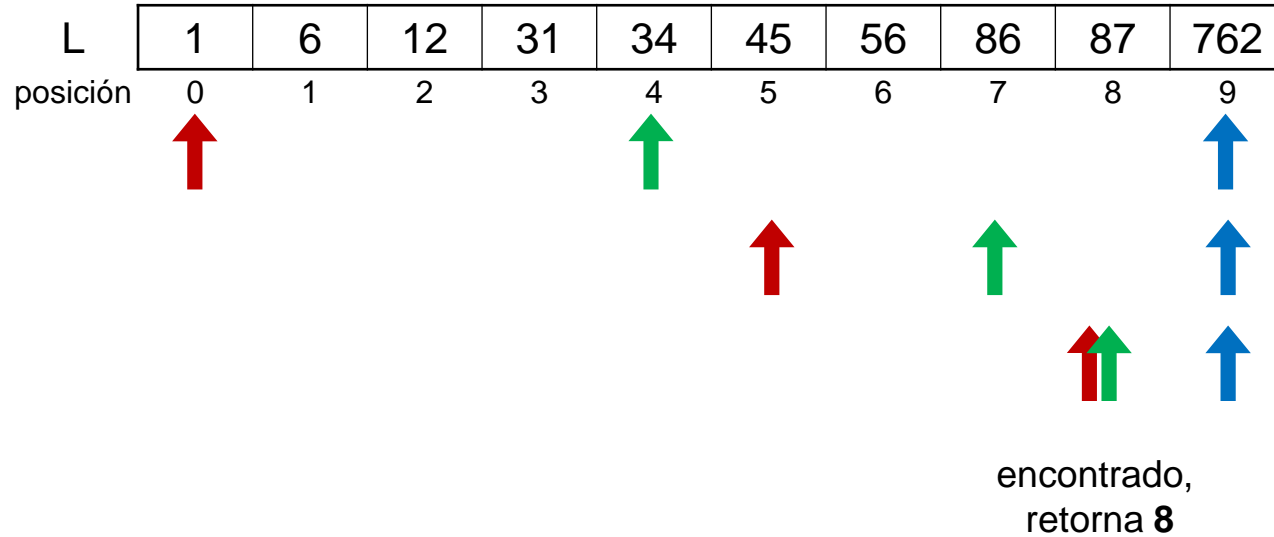
- Si no, actualizamos **superior**:

```
superior = centro - 1
```

5. Volvemos al paso 2.

Búsqueda binaria

- Ejemplo: buscamos el número **87**



Algoritmos en Python

```
def BusquedaLineal(lista, elem):  
    for pos in range(0, len(lista)):  
        if lista[pos] == elem:  
            return pos  
    return -1 # si no está en la lista
```

```
def BusquedaBinaria(lista, elem):  
    pos = -1 # si no está en la lista  
  
    inferior = 0  
    superior = len(lista)-1  
  
    while pos == -1 and inferior <= superior:  
        centro = (inferior + superior) // 2  
        if lista[centro] == elem:  
            pos = centro  
        elif lista[centro] < elem:  
            inferior = centro + 1  
        else: # lista[centro] > elem  
            superior = centro - 1  
  
    return pos
```


Búsqueda de Objetos

- ▶ Supongamos que se tiene una lista de **objetos** de una cierta clase, por ejemplo:
 - **Persona**, que contiene dos atributos: **nombre** y **edad**
 - **nombre** es un string que está compuesto por su nombre y apellido
 - **edad** que es la edad de esta persona.
- ▶ Así, la lista tendría la estructura:
 - `L=[persona_1, persona_2, ... , persona_n]`
- ▶ Para facilitar la búsqueda y el ordenamiento de listas de objetos de la clase **Persona**, es útil sobrecargar los operadores `__eq__` y `__lt__`.

Búsqueda de Objetos

```
class Persona:
    nombre = ""
    edad = 0

    def __init__(self, nombre, edad):
        self.nombre = nombre
        self.edad = edad

    def __eq__(self, otro):
        return self.edad == otro.edad

    def __lt__(self, otro):
        return self.edad < otro.edad

    def __str__(self):
        return "(" + self.nombre + "," + str(self.edad) + ")"
```

Búsqueda de Objetos

```
def BusquedaLineal(lista, elem):  
    for pos in range(0, len(lista)):  
        if lista[pos] == elem:  
            return pos  
    return -1 # si no está en la lista
```

```
def BusquedaBinaria(lista, elem):  
    pos = -1 # si no está en la lista  
  
    inferior = 0  
    superior = len(lista) - 1  
  
    while pos == -1 and inferior <= superior:  
        centro = (inferior + superior) // 2  
        if lista[centro] == elem:  
            pos = centro  
        elif lista[centro] < elem:  
            inferior = centro + 1  
        else: # lista[centro] > elem  
            superior = centro - 1  
  
    return pos
```

Búsqueda de Objetos

```
print("---- Búsqueda lineal ----")
```

```
L = [Persona("juan perez", 34), Persona("pedro lopez", 46),  
     Persona("matias donoso", 18), Persona("andres jara", 21),  
     Persona("jose tapia", 19), Persona("julio soto", 28)]  
edad = 21
```

```
pos = BusquedaLineal(L, Persona("", edad))
```

```
print("Lista:", *L, sep='\n')  
print("Persona con edad", edad, "está en la posición", pos)
```

```
print("---- Búsqueda binaria ----")
```

```
L.sort()
```

```
pos = BusquedaBinaria(L, Persona("", edad))
```

```
print("Lista:", *L, sep='\n')  
print("Persona con edad", edad, "está en la posición", pos)
```

Búsqueda de Objetos – usando index()

```
def BusquedaUsandoIndex(lista, elem):  
    if elem in L:  
        return L.index(elem)  
    else:  
        return -1  
  
print("---- Búsqueda usando index ----")  
  
pos = BusquedaUsandoIndex(L, Persona("", edad))  
  
print("Lista:", *L, sep='\n')  
print("Persona con edad", edad, "está en la posición", pos)
```

Búsqueda de Objetos

```
>>>
---- Búsqueda lineal ----
Lista:
(juan perez,34)
(pedro lopez,46)
(matias donoso,18)
(andres jara,21)
(jose tapia,19)
(julio soto,28)
Persona con edad 21 está en la posición 3
---- Búsqueda binaria ----
Lista:
(matias donoso,18)
(jose tapia,19)
(andres jara,21)
(julio soto,28)
(juan perez,34)
(pedro lopez,46)
Persona con edad 21 está en la posición 2
---- Búsqueda usando index ----
Lista:
(matias donoso,18)
(jose tapia,19)
(andres jara,21)
(julio soto,28)
(juan perez,34)
(pedro lopez,46)
Persona con edad 21 está en la posición 2
```