



# FUNDAMENTOS DE INFORMATICA

INGENIERA SILVIA PATRICIA BARDELLI

PROFESORA ING. SILVIA PATRICIA BARDELLI

# CLASE NRO 7

Temas:

- § Números al azar
- § Listas: Búsqueda secuencial
- § Ordenamiento: Método de Selección
- § Búsqueda binaria

# NÚMEROS AL AZAR

- § Son números generados, o inventados, por la computadora.
- § Se utilizan cuando se requiere un factor de azar, por ejemplo en videojuegos, criptografía o simulación de eventos.

# NÚMEROS AL AZAR

- § Python tiene la posibilidad de generar números al azar a través del módulo random.
- § Un módulo es un conjunto de funciones que ya vienen listas para usar, y que se agregan a nuestro programa a través de la instrucción **import**.

# EJEMPLO N° 1

Objetivo:

Escribir una función para simular el lanzamiento de un dado, utilizando números al azar.

# EJEMPLO N° 1

```
import random
```

```
def lanzardado( ):
    return random.randint(1, 6)
```

```
# Programa principal
```

```
dado = lanzardado( )
print(dado)
```

# NOVEDADES DEL EJEMPLO N° 1

## *Importación de módulos*

Todo módulo que desee utilizarse debe ser *importado* (incluido) al comienzo del programa.

## *Función randint(mínimo, máximo)*

Genera un número entero al azar entre los límites suministrados, ambos incluidos. El nombre de la función debe ir precedido por el del módulo, separados por un punto.

# APLICACIONES

Los números al azar se suelen utilizar para cargar vectores o listas, a fin de evitar el ingreso de tantos valores a través del teclado.



# BÚSQUEDA SECUENCIAL

- § La búsqueda secuencial es la más sencilla de las búsquedas que pueden realizarse sobre una lista.
- § Consiste en ir recorriendo la lista elemento por elemento hasta encontrar el valor buscado o hasta llegar al final, lo que significa que el valor no se encontraba presente.

## EJEMPLO N° 2

Objetivo:

Cargar una lista con números al azar entre 1 y 100, donde la cantidad de elementos será ingresada por el usuario.

Luego se solicita ingresar un valor y buscarlo en la lista, informando su ubicación o -1 si no se lo encuentra

# EJEMPLO N° 2 - PRIMERA PARTE

## *# Programa principal*

```
cant = int(input("¿Cuántos elementos desea cargar? "))
milista = cargarlista(cant)
imprimirlista(milista)
n = int(input("Ingrese el número a buscar: "))
pos = busquedasecuencial(milista, n)
if pos >= 0:
    print("El elemento", n, "se encontró en la posición", pos)
else:
    print("El valor", n, "no se encontró en la lista")
```

## EJEMPLO N° 2 - SEGUNDA PARTE

```
import random
def cargarlista(cantidad):
    lista = [ ]
    for i in range(cantidad):
        lista.append(random.randint(1, 100))
    return lista
def imprimirlista(lista):
    for i in range(len(lista)):
        print(lista[i], end=" ")
    print()
```

## EJEMPLO N° 2 – TERCERA PARTE

```
def busquedasecuencial(lista, dato):  
    i = 0  
    while i < len(lista) and lista[i] != dato:  
        i = i + 1  
    if i < len(lista):  
        return i  
    else:  
        return -1
```

## EJEMPLO N° 2 – PROGRAMA COMPLETO

```
import random
```

```
def cargarlista(cantidad):  
    lista = []  
    for i in range(cantidad):  
        lista.append(random.randint(1,100))  
    return lista
```

```
def imprimirlista(lista):  
    for i in range(len(lista)):  
        print(lista[i],end=" ")  
    print()
```

```
def busquedasecuencial(lista, dato):  
    i = 0  
    while i < len(lista) and lista[i] != dato:  
        i = i + 1  
    if i < len(lista):  
        return i  
    else:  
        return -1
```

*# Programa principal*

```
cant = int(input("¿Cuántos elementos desea cargar? "))  
milista = cargarlista(cant)  
imprimirlista(milista)  
n = int(input("Ingrese el número a buscar: "))  
pos = busquedasecuencial(milista,n)  
if pos >= 0:  
    print("El elemento",n,"se encontró en la posición", pos)  
else:  
    print("El valor",n,"no se encontró en la lista")
```

# ORDENAMIENTO DE VECTORES

- § El ordenamiento o clasificación de elementos ha sido una necesidad desde mucho antes de la aparición de la informática.
- § La naturaleza monótona y repetitiva de esta tarea la vuelve ideal para ser realizada mediante computadoras.

# ORDENAMIENTO DE VECTORES

- § Los elementos que se desea ordenar deben estar almacenados en una estructura de datos, y por eso se suelen usar arreglos (listas o vectores).
- § Existen muchos métodos de ordenamiento. En este curso veremos tres: Selección, Intercambio e Inserción.



# MÉTODO DE SELECCIÓN

- § Consiste en buscar el menor elemento de todo el arreglo e intercambiarlo con el de la primera posición.
- § Luego se busca el segundo menor elemento y se lo intercambia con el de la segunda posición, y así sucesivamente

# EJEMPLO

2	7	5	1	6
0	1	2	3	4
↑	↑			

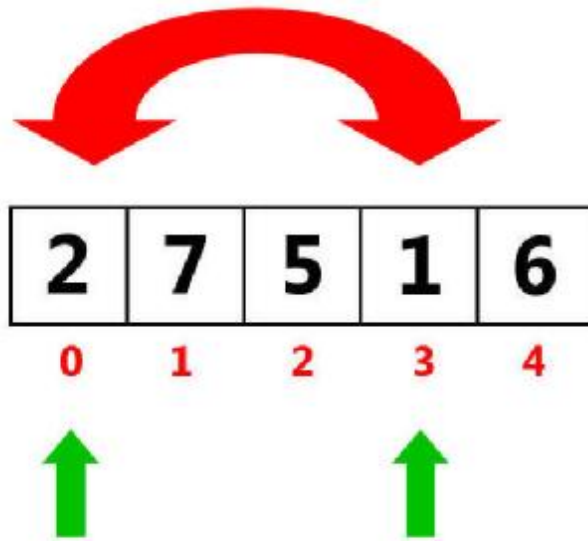
i	j
0	1

# EJEMPLO

<b>2</b>	<b>7</b>	<b>5</b>	<b>1</b>	<b>6</b>
0	1	2	3	4
↑		↑		

i	j
0	1
0	2

# EJEMPLO



i	j
0	1
0	2
0	3

-> intercambio

# EJEMPLO

<b>1</b>	<b>7</b>	<b>5</b>	<b>2</b>	<b>6</b>
0	1	2	3	4


↑                      ↑

i	j
0	1
0	2
0	3

-> intercambio

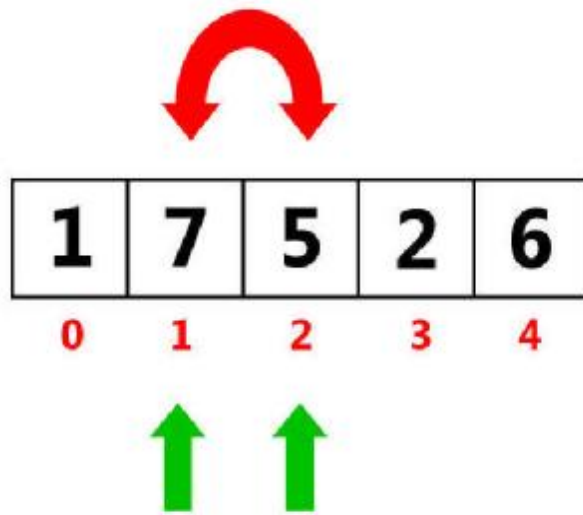
# EJEMPLO

<b>1</b>	<b>7</b>	<b>5</b>	<b>2</b>	<b>6</b>
<b>0</b>	<b>1</b>	<b>2</b>	<b>3</b>	<b>4</b>



<b>i</b>	<b>j</b>
0	1
0	2
0	3
0	4

# EJEMPLO



i	j
1	2

-> intercambio

# EJEMPLO

<b>1</b>	<b>5</b>	<b>7</b>	<b>2</b>	<b>6</b>
0	1	2	3	4

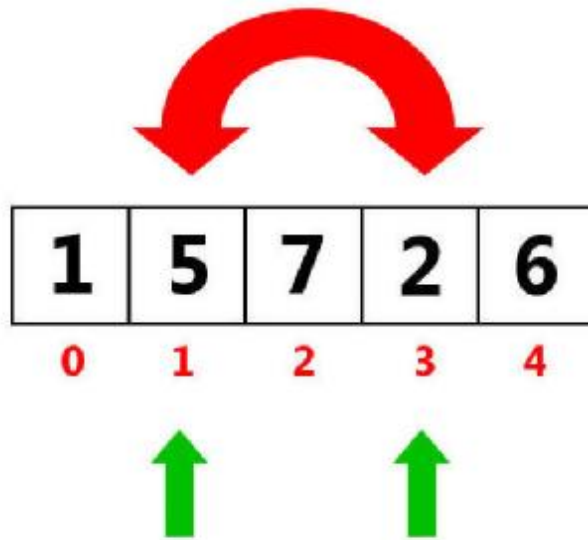


i	j
1	2

-> intercambio



# EJEMPLO



i	j
1	2
1	3

-> intercambio

# EJEMPLO

<b>1</b>	<b>2</b>	<b>7</b>	<b>5</b>	<b>6</b>
<b>0</b>	<b>1</b>	<b>2</b>	<b>3</b>	<b>4</b>




<b>i</b>	<b>j</b>
1	2
1	3

-> intercambio

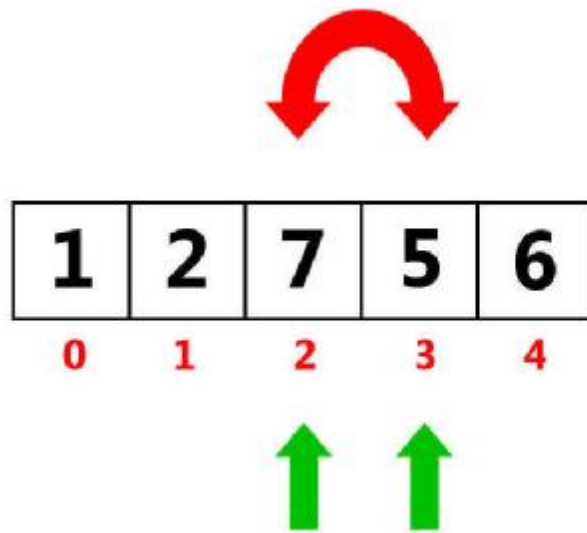
# EJEMPLO

<b>1</b>	<b>2</b>	<b>7</b>	<b>5</b>	<b>6</b>
<b>0</b>	<b>1</b>	<b>2</b>	<b>3</b>	<b>4</b>



<b>i</b>	<b>j</b>
1	2
1	3
1	4

# EJEMPLO



i	j
2	3

-> intercambio

# EJEMPLO


<b>1</b>	<b>2</b>	<b>5</b>	<b>7</b>	<b>6</b>
<b>0</b>	<b>1</b>	<b>2</b>	<b>3</b>	<b>4</b>



<b>i</b>	<b>j</b>	
2	3	-> intercambio

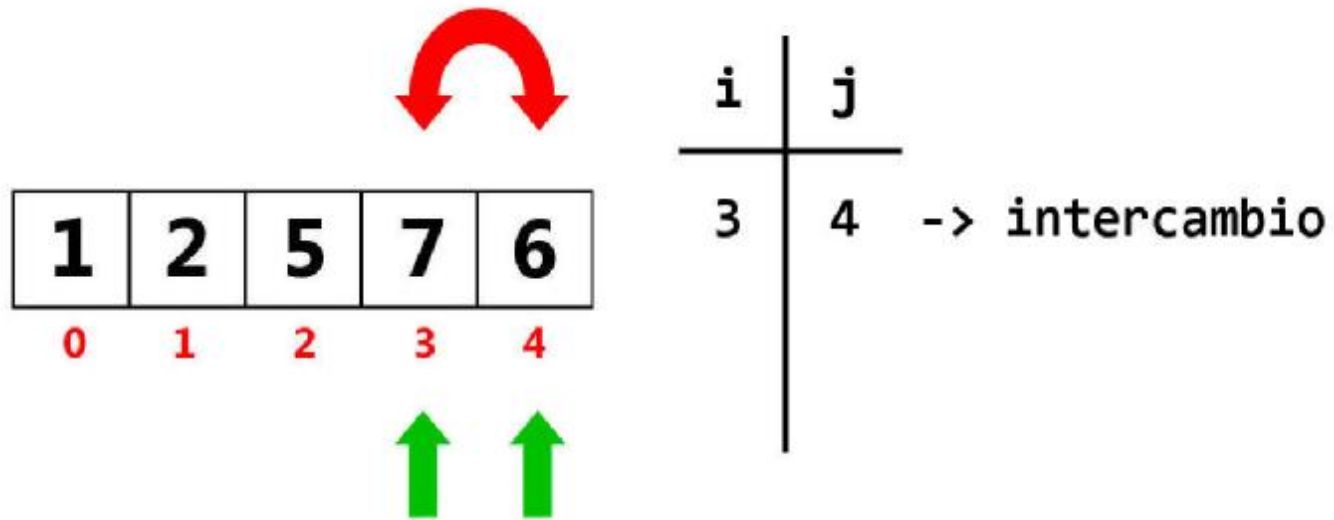
# EJEMPLO

<b>1</b>	<b>2</b>	<b>5</b>	<b>7</b>	<b>6</b>
<b>0</b>	<b>1</b>	<b>2</b>	<b>3</b>	<b>4</b>



<b>i</b>	<b>j</b>
<b>2</b>	<b>3</b>
<b>2</b>	<b>4</b>

# EJEMPLO



# EJEMPLO

<b>1</b>	<b>2</b>	<b>5</b>	<b>6</b>	<b>7</b>
<b>0</b>	<b>1</b>	<b>2</b>	<b>3</b>	<b>4</b>



<b>i</b>	<b>j</b>	
3	4	-> intercambio



# MÉTODO DE SELECCIÓN

```
def metododeseleccion(v):  
    largo = len(v)  
    for i in range(largo - 1):  
        for j in range(i+1, largo):  
            if v[i] > v[j]:  
                aux = v[i]  
                v[i] = v[j]  
                v[j] = aux
```

# BÚSQUEDA BINARIA

- § La búsqueda binaria aprovecha el hecho de contar con la lista ya ordenada.
- § Esto le permite completar el proceso en mucho menos tiempo de lo que tomaría hacerlo con búsqueda secuencial.

# BÚSQUEDA BINARIA

Procedimiento:

- § Se verifica si en la mitad de la lista se encuentra el elemento buscado.
- § Si no está, resulta fácil deducir para qué lado podría llegar a encontrarse debido al ordenamiento.
- § Se descarta una mitad y se repite el proceso sobre la otra.

# BÚSQUEDA BINARIA

1	3	4	7	9	12	16	19	21	22
0	1	2	3	4	5	6	7	8	9

16

Buscar

# BÚSQUEDA BINARIA

1	3	4	7	9	12	16	19	21	22
0	1	2	3	4	5	6	7	8	9

16
----

Buscar

--

izq

--

centro

--

der

# BÚSQUEDA BINARIA

1	3	4	7	9	12	16	19	21	22
0	1	2	3	4	5	6	7	8	9

16

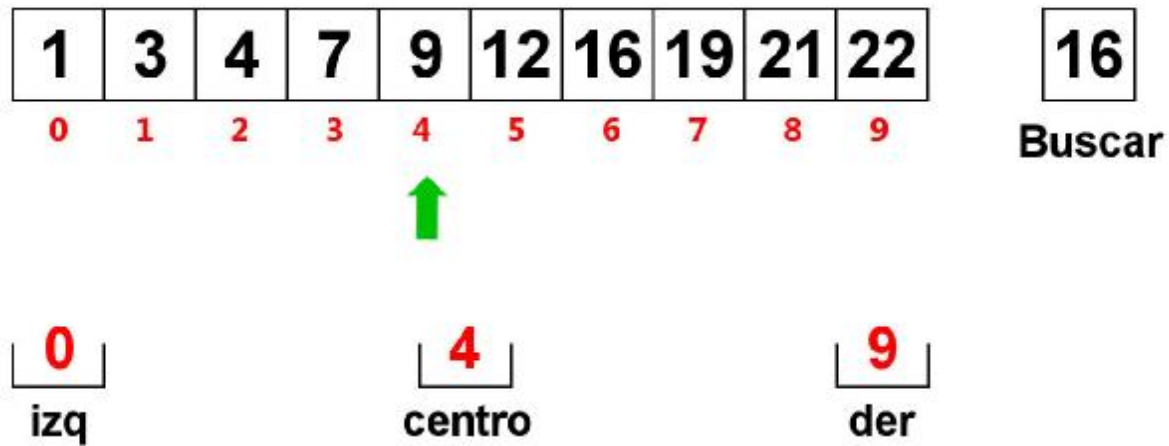
Buscar

0  
izq

centro

9  
der

# BÚSQUEDA BINARIA



# BÚSQUEDA BINARIA

1	3	4	7	9	12	16	19	21	22
0	1	2	3	4	5	6	7	8	9

16

Buscar

5  
~~0~~  
izq

7  
~~4~~  
centro

9  
  
der



# BÚSQUEDA BINARIA

1	3	4	7	9	12	16	19	21	22
0	1	2	3	4	5	6	7	8	9

**16**  
Buscar



**5**  
izq

**7**  
centro

**9**  
der

# BÚSQUEDA BINARIA

1	3	4	7	9	12	16	19	21	22
0	1	2	3	4	5	6	7	8	9

16

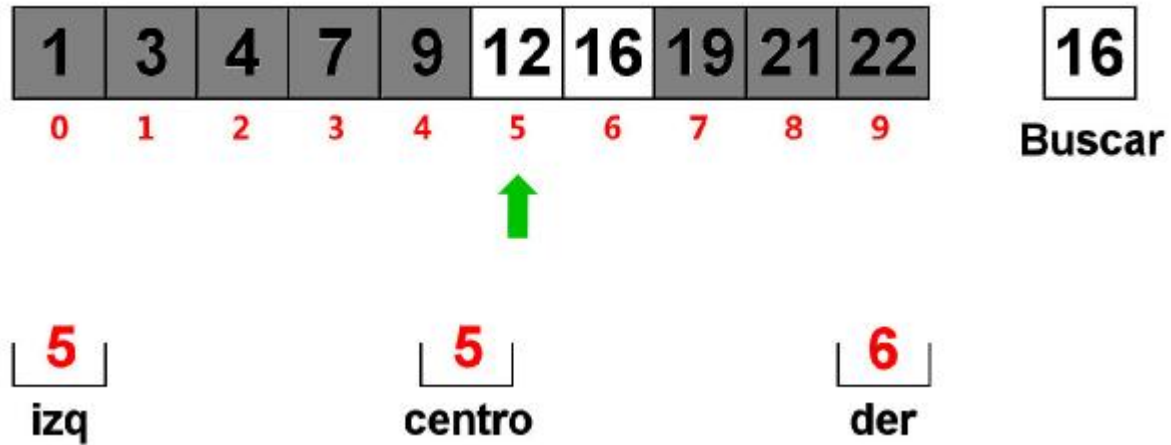
Buscar

5  
izq

5  
~~7~~  
centro

6  
~~9~~  
der

# BÚSQUEDA BINARIA



# BÚSQUEDA BINARIA

1	3	4	7	9	12	16	19	21	22
0	1	2	3	4	5	6	7	8	9

16

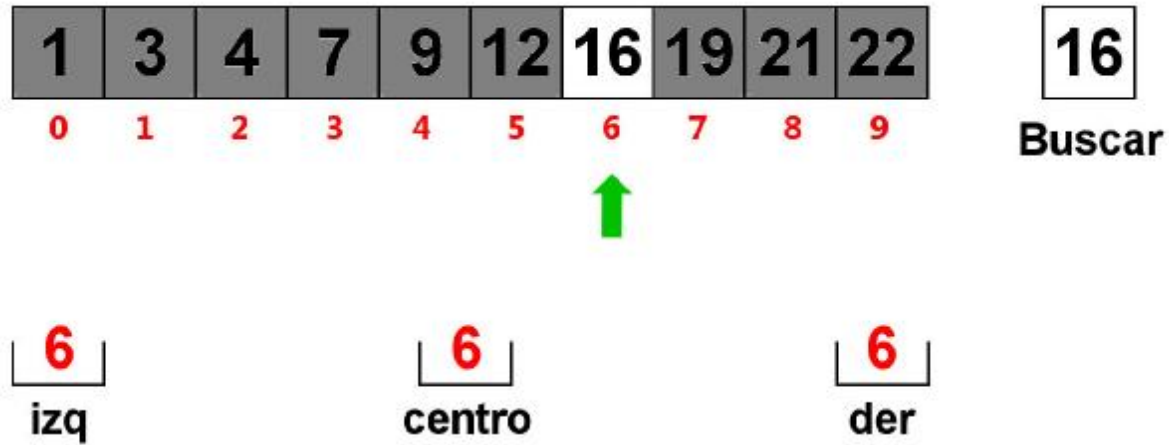
Buscar

6  
~~5~~  
izq

6  
~~5~~  
centro

6  
der

# BÚSQUEDA BINARIA



# BÚSQUEDA BINARIA

- § En sólo 4 comparaciones encontramos el valor buscado. Con búsqueda secuencial habrían sido necesarias 7.
- § La diferencia aumenta a medida que crece la cantidad de elementos.
- § Con una lista de 2000 elementos, a lo sumo se necesitan 11 comparaciones

# BÚSQUEDA BINARIA

```
def busquedabinaria(v, dato):  
    izq = 0  
    der = len(v) - 1  
    pos = -1  
    while izq <= der and pos == -1:  
        centro = (izq + der) // 2  
        if v[centro] == dato:  
            pos = centro  
        elif v[centro] < dato:  
            izq = centro + 1  
        else:  
            der = centro - 1  
    return pos
```

# •Ejercitación