Algorítimica y Programación

Enero - Mayo 2020

Dr. Iván S. Razo Zapata
(ivan.razo@itam.mx)

ITAM

Contenido

- Búsqueda secuencial
- Búsqueda binaria
- Ordenamiento por selección

ITAM

Numpy

Numpy

- Arreglos de una dimensión
- Creación usando: zeros, empty, arange y array

Numpy

```
import numpy as np

arreglo1 = np.zeros(10)
print(arreglo1)

arreglo2 = np.empty(10)
print(arreglo2)

arreglo3 = np.arange(10,20)
print(arreglo3)

arreglo4 = np.array([99, 88, 55, 22, 62, 72, 45, 11, 36])
print(arreglo4)
```

ITAM

Búsqueda secuencial

- Dado un arreglo con **N** valores
- Recorrer dicho arreglo hasta encontrar el **elemento** buscado

Búsqueda secuencial

```
def busquedaSecuencial(arreglo, elemento):
    pos = 0              # Buscamos desde la primer posición, i.e. 0 (cero)
    encontrado = False   # En cuanto se encuentre el elemento, encontrado debe ser True

    while (pos < arreglo.size) and (encontrado == False):
        if arreglo[pos] == elemento:
            encontrado = True
        else:
            pos += 1

    return encontrado, pos
```

Búsqueda secuencial - Problemas

- Si el elemento buscado está en el arreglo:
  - Mejor caso está en la 1era posición
  - Peor caso está en la última posición
    - Si **N** es muy grande, el tiempo de búsqueda puede ser considerable

En promedio, la búsqueda requiere **N/2** comparaciones

ITAM

Búsqueda secuencial - Problemas

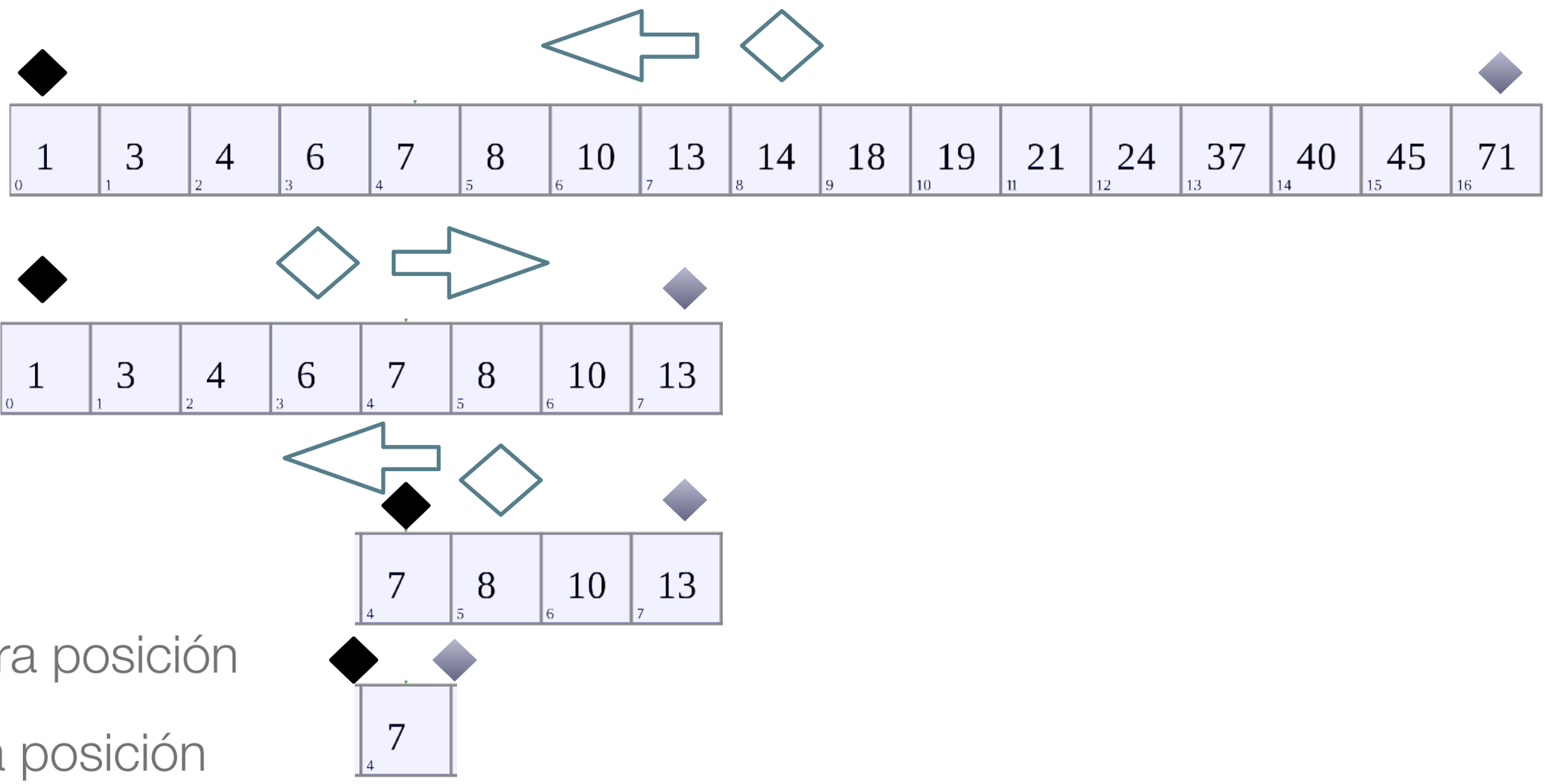
- Si el elemento buscado **NO** está en el arreglo:
  - La búsqueda requiere **N** comparaciones

Alternativa: **Búsqueda binaria**

- Suponiendo que el arreglo está ordenado
- Divide and conquer
  - Dividimos el problema en partes más pequeñas, resolvemos dichas partes más pequeñas de alguna manera y luego reensamblamos todo el problema para obtener el resultado

Búsqueda binaria

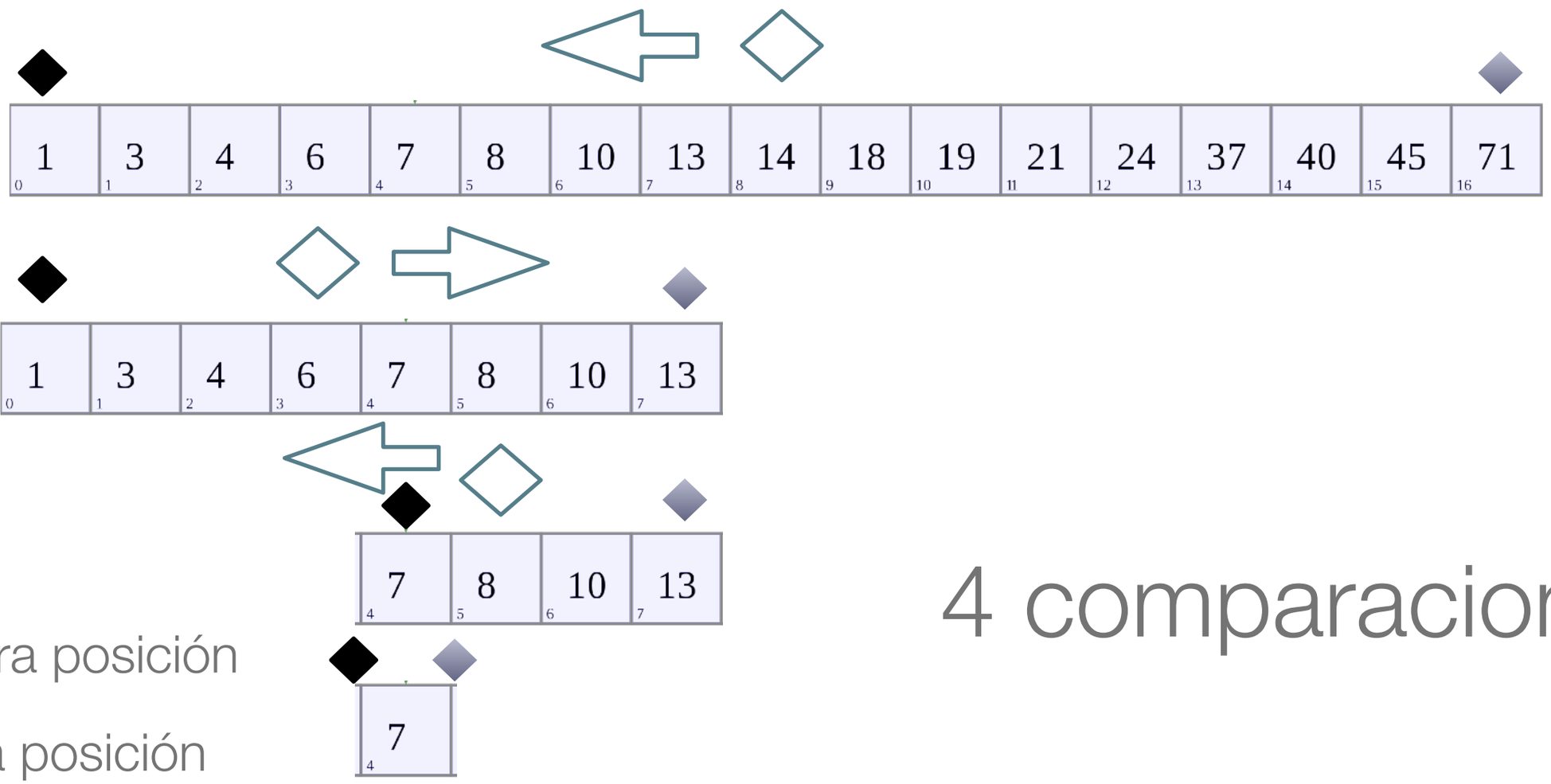
Queremos saber si 7 está en el arreglo



- Primera posición
- Última posición

Búsqueda binaria

Queremos saber si 7 está en el arreglo

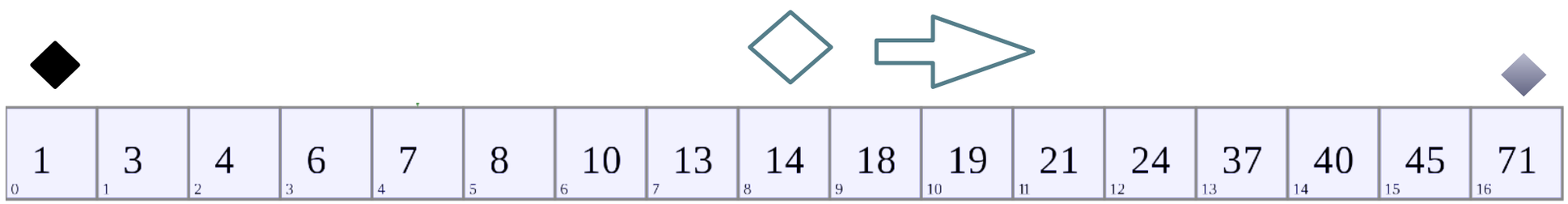


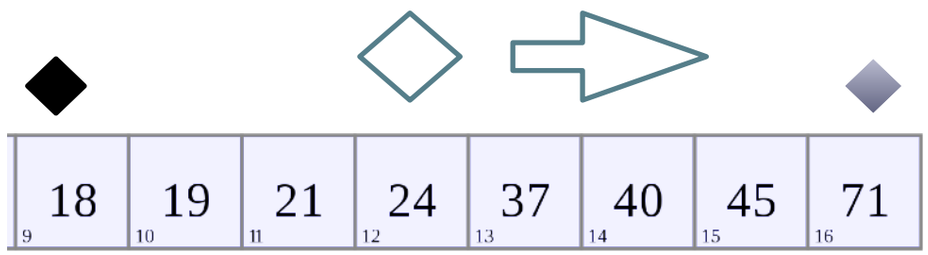
4 comparaciones

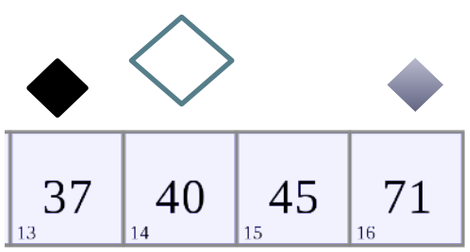
Cuantas si busco el 8?

Búsqueda binaria

Queremos saber si 40 está en el arreglo







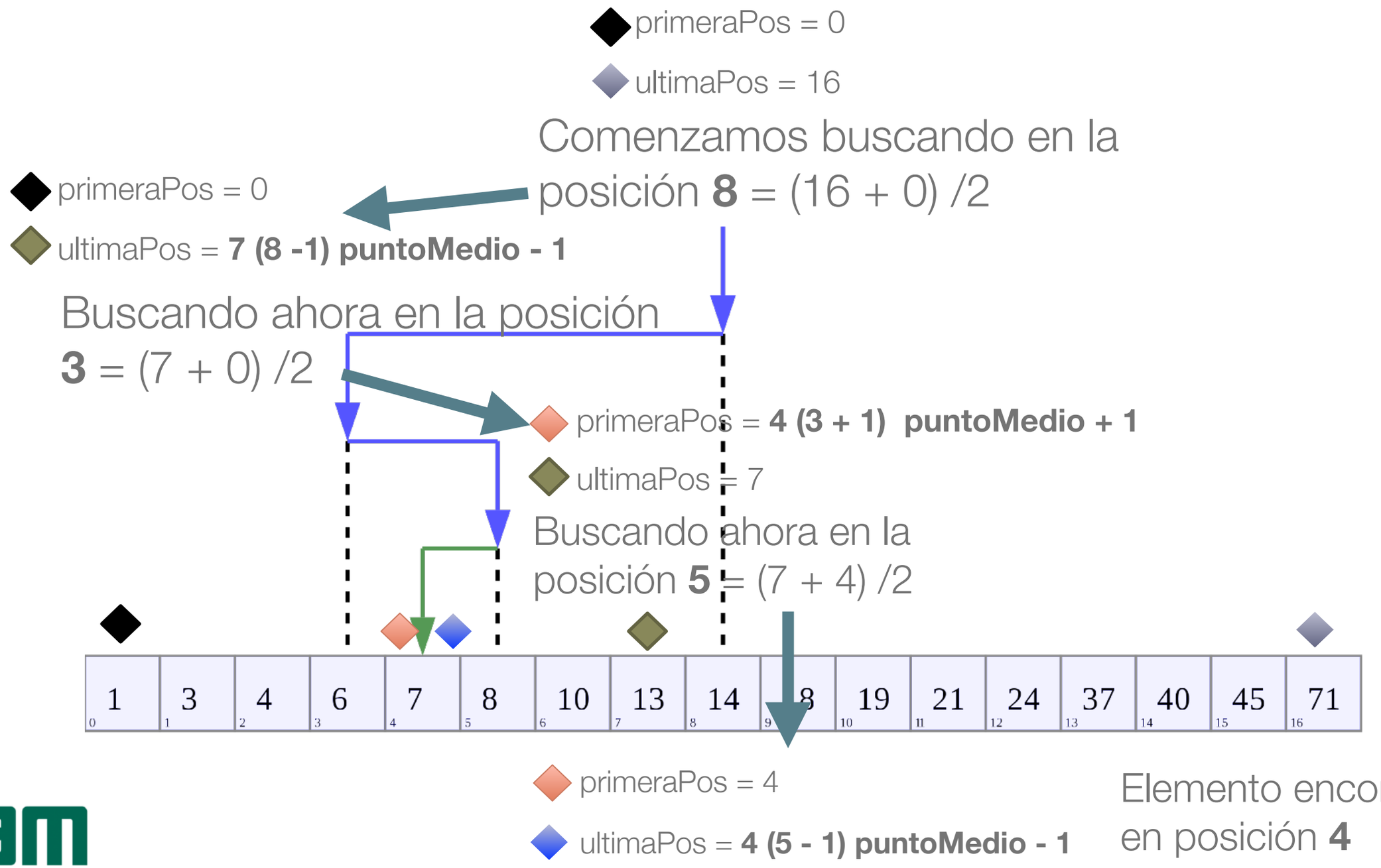
- Primera posición
- Última posición

3 comparaciones

Cuantas con búsqueda secuencia?

Búsqueda binaria

Queremos saber si 7 está en el arreglo



Búsqueda binaria

```
def busquedaBinaria(arreglo, elemento):
    encontrado = False
    primero = 0
    ultimo = arreglo.size - 1
```

- 10min para el código
- Hints
  - Primera posición siempre deber ser menor que ultima posición

ITAM

Ordenamiento por selección

- Dado un arreglo desordenado
- Recorrer el arreglo, seleccionar los valores mas pequeños y ponerlos en las primeras posiciones
- Ejemplo:
  - Para el arreglo A (de tamaño N), seleccionar el valor más pequeño e intercambiarlo por el valor en A[1]
  - Después, buscar el segundo valor más pequeño e intercambiarlo por el valor en A[2]
  - and so on!
  - Así hasta N-1

Ordenamiento por selección

Arreglo inicial: [**99 88 55 22 62 72 45 11 36**]

Después de 1era selección: [**11 88 55 22 62 72 45 99 36**]

Después de 2da selección: [**11 22 55 88 62 72 45 99 36**]

Después de 3era selección: [**11 22 36 88 62 72 45 99 55**]

Después de 4ta selección: [**11 22 36 45 62 72 88 99 55**]

Después de 5ta selección: [**11 22 36 45 55 72 88 99 62**]

Después de 6ta selección: [**11 22 36 45 55 62 88 99 72**]

Después de 7ma selección: [**11 22 36 45 55 62 72 99 88**]

Después de 8va selección: [**11 22 36 45 55 62 72 88 99**]

ITAM

Ordenamiento por selección

```
def ordenamientoPorSeleccion(arreglo):
    # Primer ciclo for
    # Recorrer el arreglo hasta size - 1
    for i in range(arreglo.size):
        # Asumimos el minimo está en la posición i
        indiceMin = i
```

- 10min para el código
- Hints
  - Dos ciclos for
  - Segundo ciclo comienza en i+1

Búsqueda binaria - con **recursividad**

- Recursividad o recursión
- Cuando una función se llama/invoca a si misma

Recursividad

Recursividad

- Ejemplo

```
def factorial(n):
    if n == 0:
        return 1
    else:
        return n * factorial (n-1)
```

Caso base a.k.a. kick

Sin caso base → Limbo

Recursividad

- Ejemplo 2

```
def fibbonacci(n):
    if (n == 0) or (n == 1):
        return n
    else:
        return fibbonacci(n-1) + fibbonacci(n-2)
```

Caso base ???

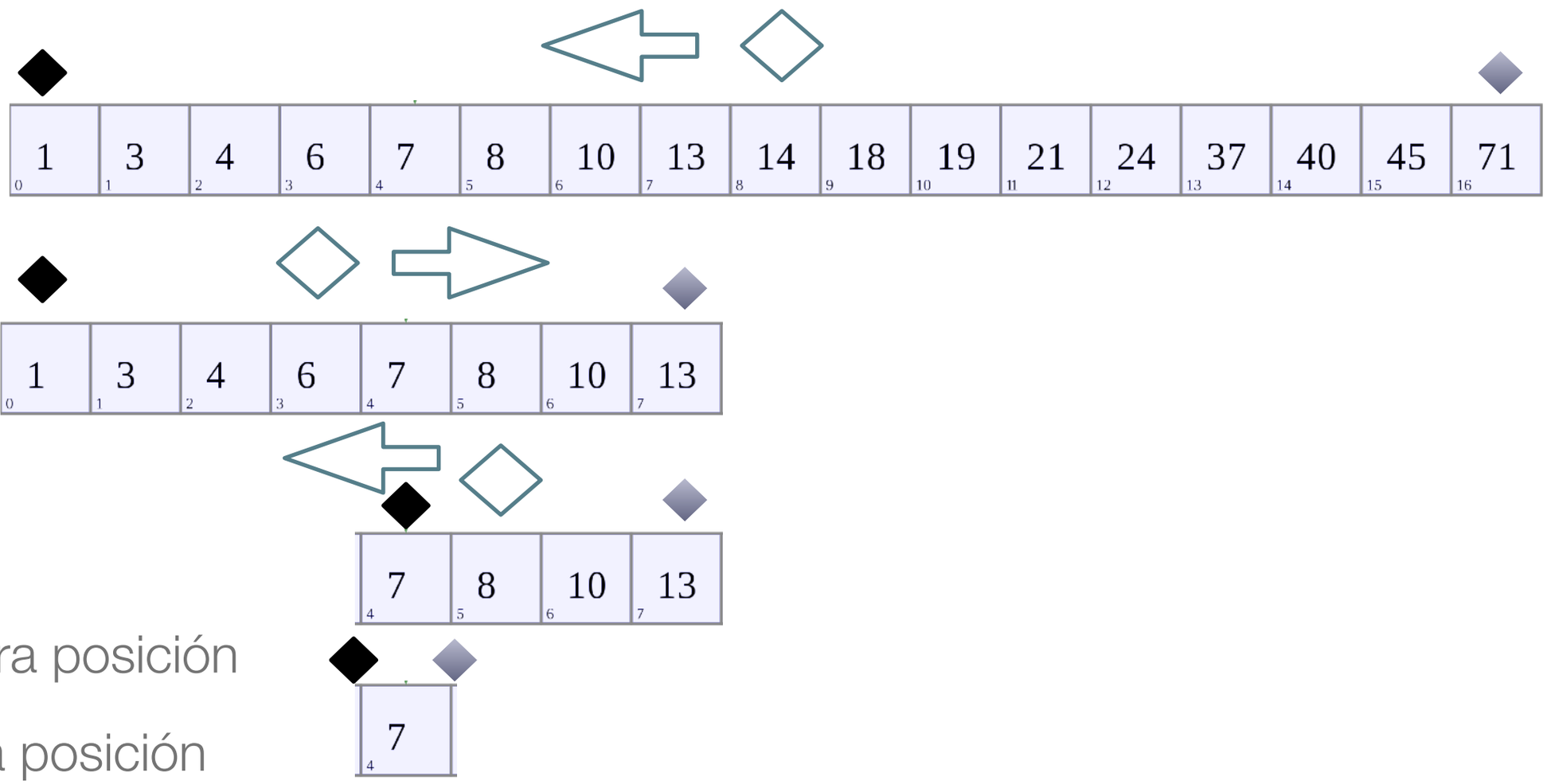
Recursividad

Caso base ???

Recursividad

Búsqueda binaria - con **recursividad**

Queremos saber si 7 está en el arreglo



- Primera posición
- Última posición

Búsqueda binaria - con **recursividad**

```
def busquedaBinariaRecursiva(arreglo, primero, ultimo, elemento):
    if (ultimo < primero):
        posicion = -1
```