Algorítimica y Programación

Enero - Mayo 2020

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

ITAM

Repaso

Elementos a evaluar

- Estructuras
  - Diccionarios, listas, tuplas
- Funciones
  - Recursividad
- Pandas
  - Dataframes
- **Numpy**
  - Arreglos unidimensionales
  - Arreglos bidimensionales

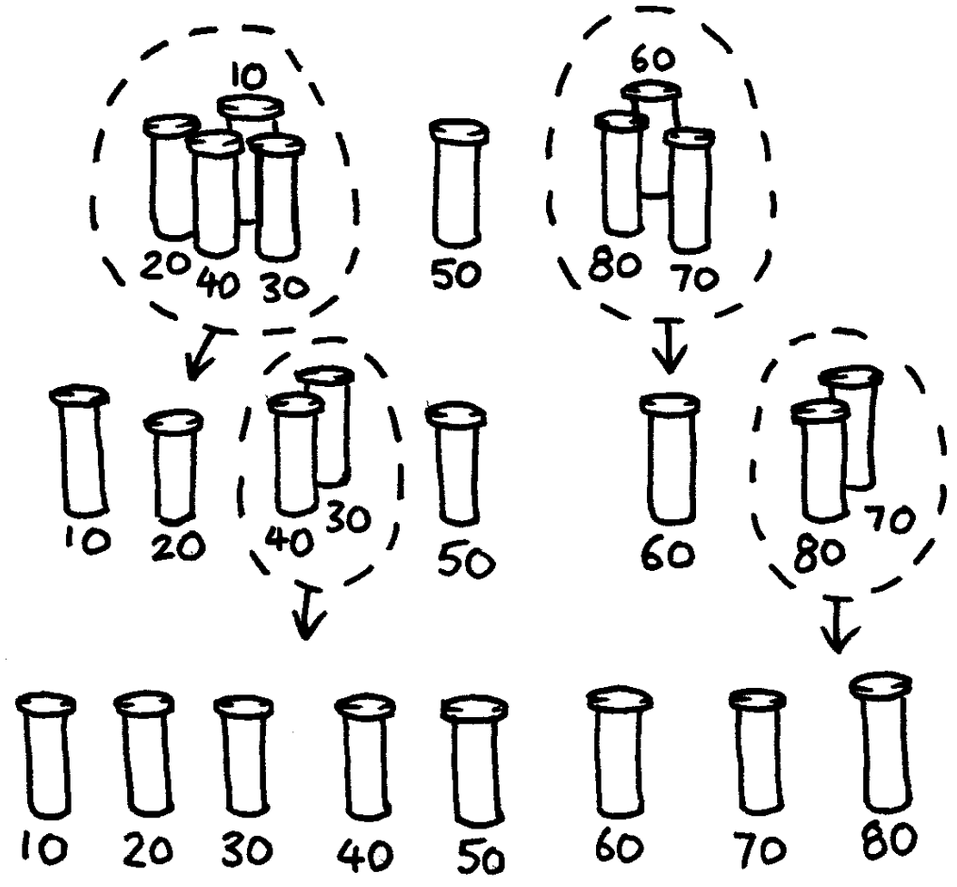
ITAM

Numpy

- Arreglos unidimensionales
- Quicksort
  - El método conocido como Quicksort es más rápido que el ordenamiento por selección
  - De hecho, es uno de los mejores métodos que se conocen
  - Divide y venceras

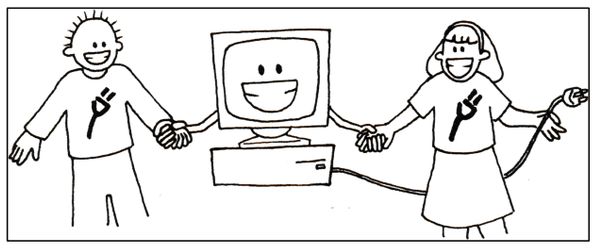
Quicksort

- Selecciona aleatoriamente **uno** de los envases
- Ahora compara éste con cada uno de los envase restantes. Coloca aquellos que son más livianos a la izquierda, luego coloca el envase que seleccionaste en el centro, y los envases más pesados a la derecha



- Selecciona uno de los grupos y repite este procedimiento. Haz lo mismo con el otro grupo
- Repite este procedimiento en cada grupo que vayas formando hasta que ninguno de los grupos tenga más de un envase
- Una vez que todos los grupos hayan sido divididos en **grupos de un solo envase**, entonces los envases estarán ordenados del más liviano al más pesado

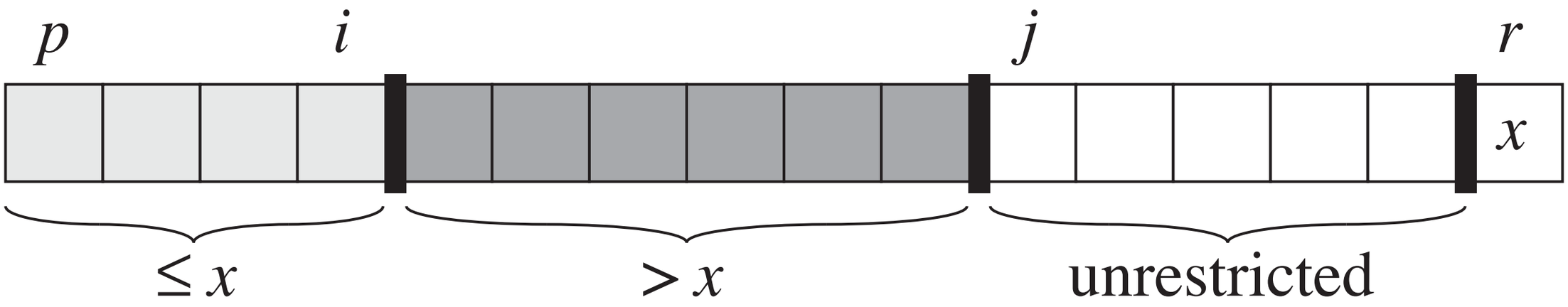
CS UNPLUGGED



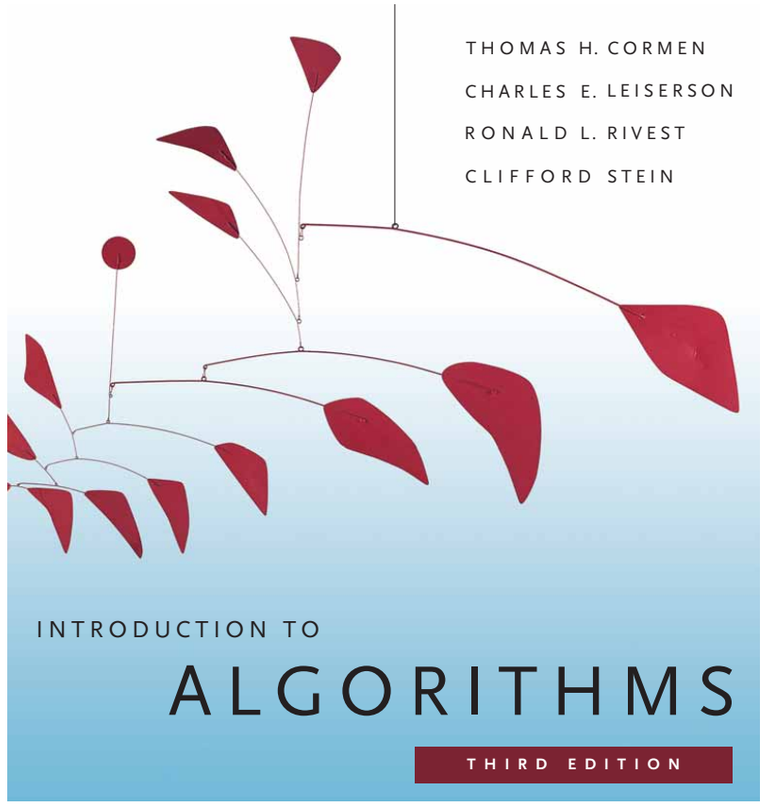
An enrichment and extension programme for primary-aged students

ITAM

Quicksort



- p - primera posición
- r - última posición
- x - pivote
- i - posición de frontera, i.e. valores menores que x
- j - posición que se va verificando en el arreglo



ITAM

Quicksort

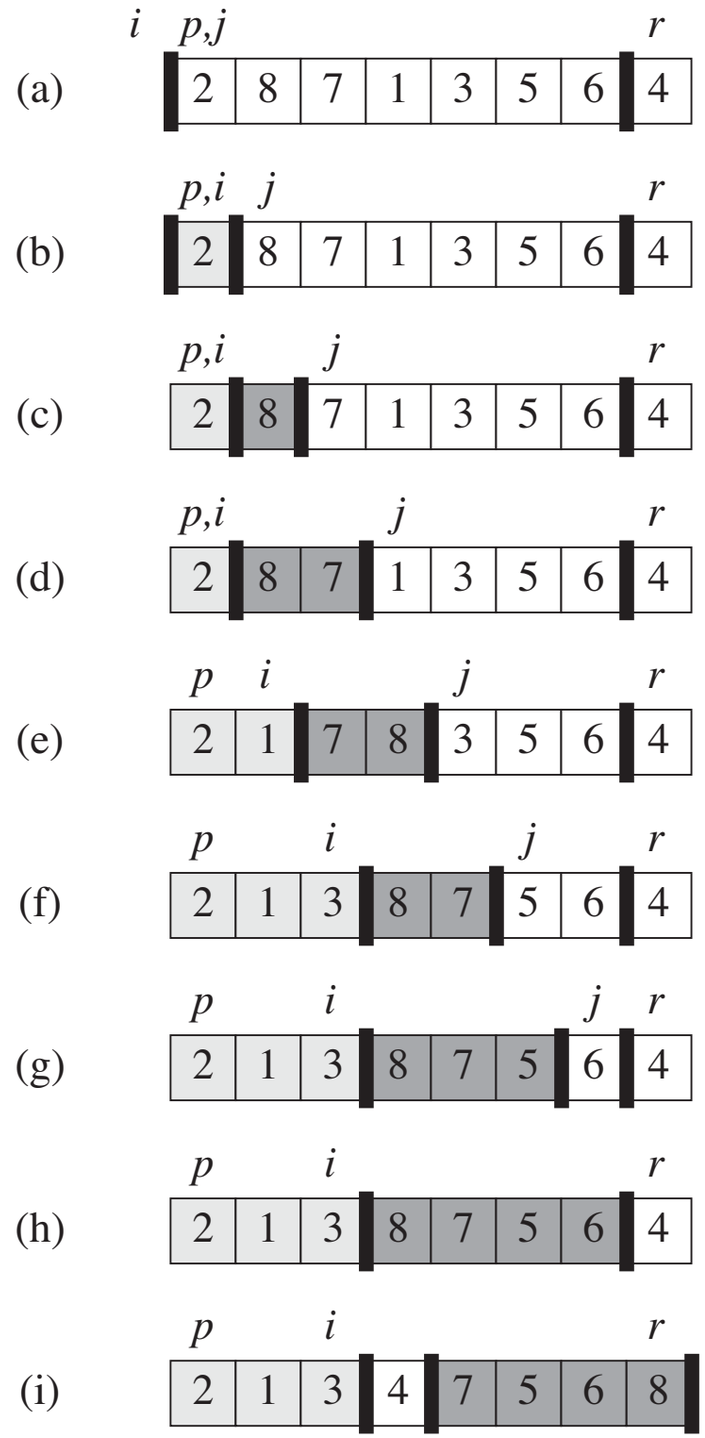
The key to the algorithm is the PARTITION procedure, which rearranges the subarray $A[p \,..\, r]$ in place.

```
PARTITION(A, p, r)
1   x = A[r]
2   i = p − 1
3   for j = p to r − 1
4       if A[j] ≤ x
5           i = i + 1
6           exchange A[i] with A[j]
7   exchange A[i + 1] with A[r]
8   return i + 1
```

Quicksort - ejemplo (primera partición)

```
PARTITION(A, p, r)
1   x = A[r]
2   i = p − 1
3   for j = p to r − 1
4       if A[j] ≤ x
5           i = i + 1
6           exchange A[i] with A[j]
7   exchange A[i + 1] with A[r]
8   return i + 1
```

- p - primera posición
- r - última posición
- x - pivote = **4**
- i,j = 0



	i = -1	j = 0
Comparar 2 con 4	**i = 0**	**2X2**
		j = 1
Comparar 8 con 4		
		j = 2
Comparar 7 con 4		
		j = 3
Comparar 1 con 4	**i = 1**	**1X8**
		j = 4
Comparar 3 con 4	**i = 2**	**3X7**
		j = 5
Comparar 5 con 4		
		j = 6
Comparar 6 con 4		
		j = 7
Intercambiar 8 con 4		

ITAM

Quicksort - ejemplo (primera partición)

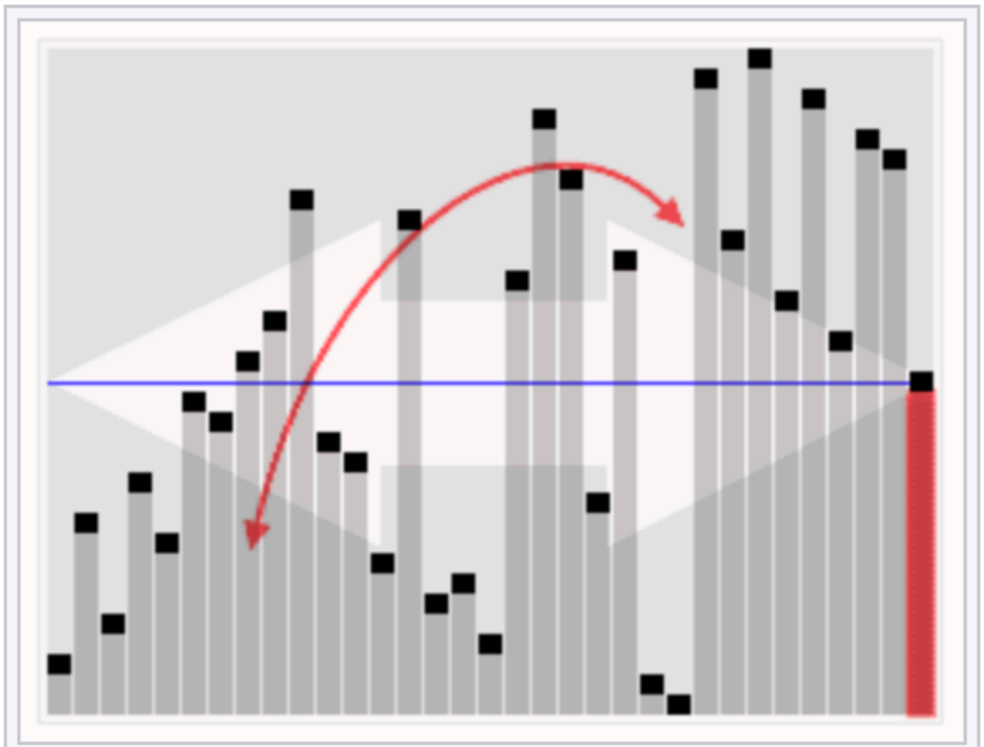
	i = -1	j = 0
Comparar 2 con 3	**i = 0**	**2X2**
		j = 1
Comparar 1 con 3	**i = 1**	**1X1**
		j = 2
		3X3

2	1	3	4	7	5	6	8
2	1	**3**					
2	1	**3**					
2	**1**						
1	2	3	4				
				?	?	?	?

```
PARTITION(A, p, r)
1  x = A[r]
2  i = p − 1
3  for j = p to r − 1
4       if A[j] ≤ x
5            i = i + 1
6            exchange A[i] with A[j]
7  exchange A[i + 1] with A[r]
8  return i + 1
```

Quicksort - visualización

https://es.wikipedia.org/wiki/Quicksort



Quicksort en acción sobre una lista de números aleatorios. Las líneas horizontales son valores **pivote**.

Quicksort

```
QUICKSORT(A, p, r)
1   if p < r
2       q = PARTITION(A, p, r)
3       QUICKSORT(A, p, q − 1)
4       QUICKSORT(A, q + 1, r)
```

To sort an entire array A, the initial call is QUICKSORT$(A, 1, A.length)$.