Quicksort

Ernesto Rodriguez - Juan Roberto Alvaro Saravia

Universidad Francisco Marroquin

ernestorodriguez@ufm.edu - juanalvarado@ufm.edu

Quicksort

- Tiene tiempo de ejecución de $\mathcal{O}(n^2)$
- Teoricamente, no es el algoritmo más eficiente
- Es a menudo la mejor opcion en practica
- Puede ordenar los arreglos eficientemente *en la misma memoria* (in-place)
- Su tiempo promedio es $\mathcal{O}(nlog(n))$ y las constantes escondidas son pequeñas.
- Es el algoritmo de ordenamiento más analizado en la academia y utilizado en la industria

Un parentesis

- Algoritmos *in-place* y no *in-place*
- Algoritmos *estables* e *inestables*

Intuición

Dado un arreglo Xs;

1. Seleccionar un indice p, llamado pivote
2. Colocar los elementos $x \leq Xs[p]$ a la izquierda del arreglo
3. Colocar los elementos $x > Xs[p]$ a la derecha del arreglo
4. Ordenar recursivamente los arreglos $Xs[1, p-1]$ y $Xs[p+1, \texttt{len}(Xs)]$
5. Juntar los arreglos resultantes y el pivote en el orden correcto

Algoritmo

Algorithm 1 Quicksort

```
1: procedure QUICKSORT(Ns,p,r)
2:     if p < r then
3:         q ← particion(Ns, p, r)
4:         quicksort(Ns, p, q − 1)
5:         quicksort(Ns, q + 1, r)
```

Para ordenar un arreglo, se ejecuta: $\texttt{quicksort}(Ns, 1, \texttt{len}(Ns))$

Partición

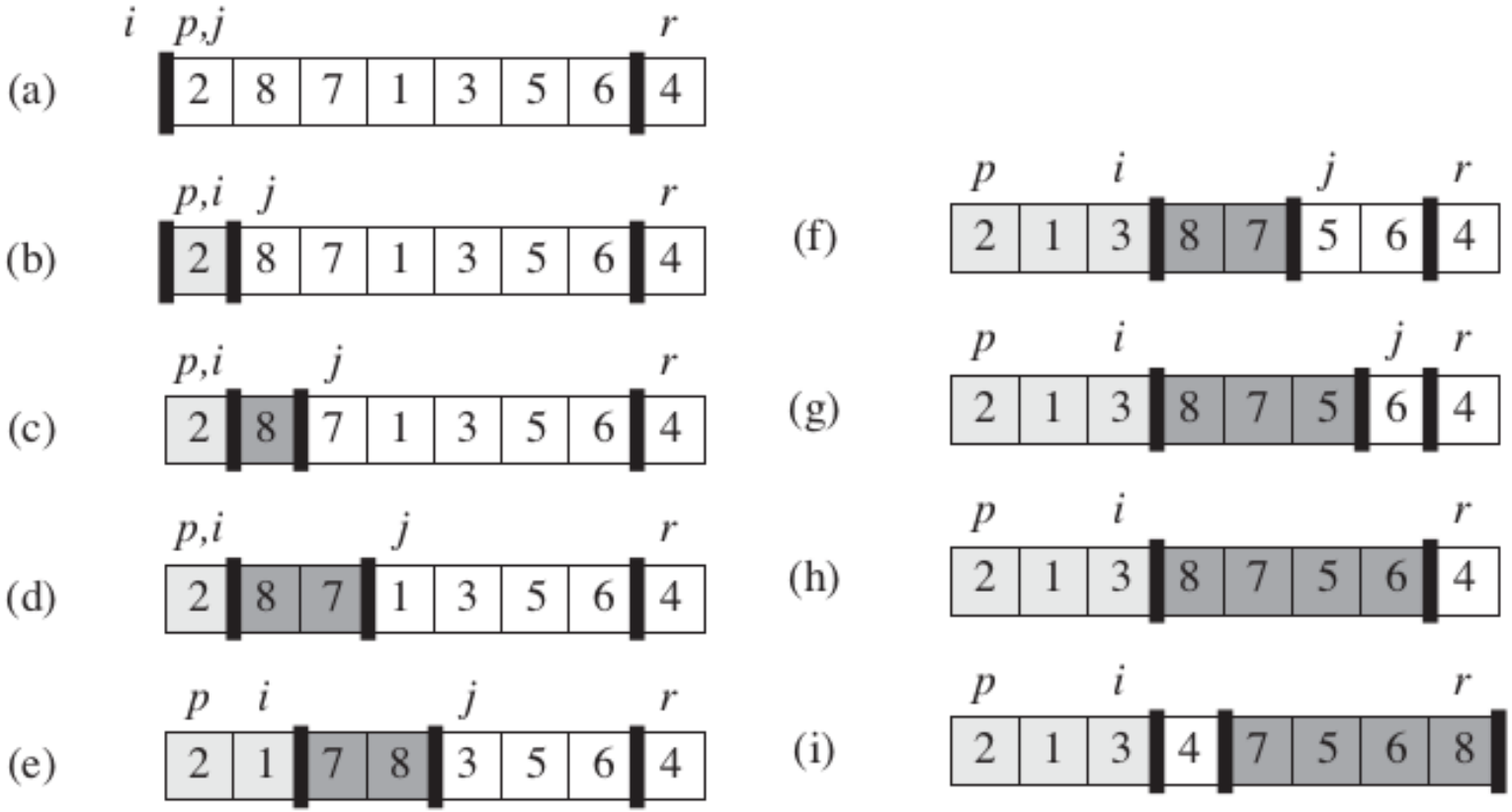
Partición

Algorithm 2 Partición

```
1: procedure PARTICION(Ns,p,r)
2:     x ← Ns[r]
3:     i ← p − 1
4:     for j ← p to r − 1 do
5:         if A[j] ≤ x then
6:             i ← i + 1
7:             intercambiar(Ns, i, j)
8:     intercambiar(Ns, i + 1, r)
9:     return i + 1
```

- Similitudes entre esta implementación y la suya
- Analisis asintoticio entre ambos

Partición


(a)
i p,j r
2 8 7 1 3 5 6 4
(b)
p,i j r
2 8 7 1 3 5 6 4
(c)
p,i j r
2 8 7 1 3 5 6 4
(d)
p,i j r
2 8 7 1 3 5 6 4
(e)
p i j r
2 1 7 8 3 5 6 4
(f)
p i j r
2 1 3 8 7 5 6 4
(g)
p i j r
2 1 3 8 7 5 6 4
(h)
p i r
2 1 3 8 7 5 6 4
(i)
p i r
2 1 3 4 7 5 6 8