Quicksort

Agustín J. González

ELO320: Estructura de Datos y Algoritmos

1er. Sem. 2002

Quicksort Descripción

- Quicksort, como mergesort, está basado en el paradigma “dividir y conquistar”.
- Pasos:
  - Dividir: el arreglo se particiona en dos sub-arreglos no vacíos, tal que cada elemento de un sub-arreglo sea menor o igual a los elementos del otro sub-arreglo.
  - Conquista: los dos arreglos son ordenados llamando recursivamente a quicksort.
  - Combinar: Como cada arreglo ya está ordenado, no serequiere trabajo adicional.
- Algoritmo
- Quicksor(A,p,r){

```
        if (p<r) {
                q = Partition(A,p,r);
                Quicksort(A,p,q);
                Quicksort(A,q+1, r);
        }
}
```

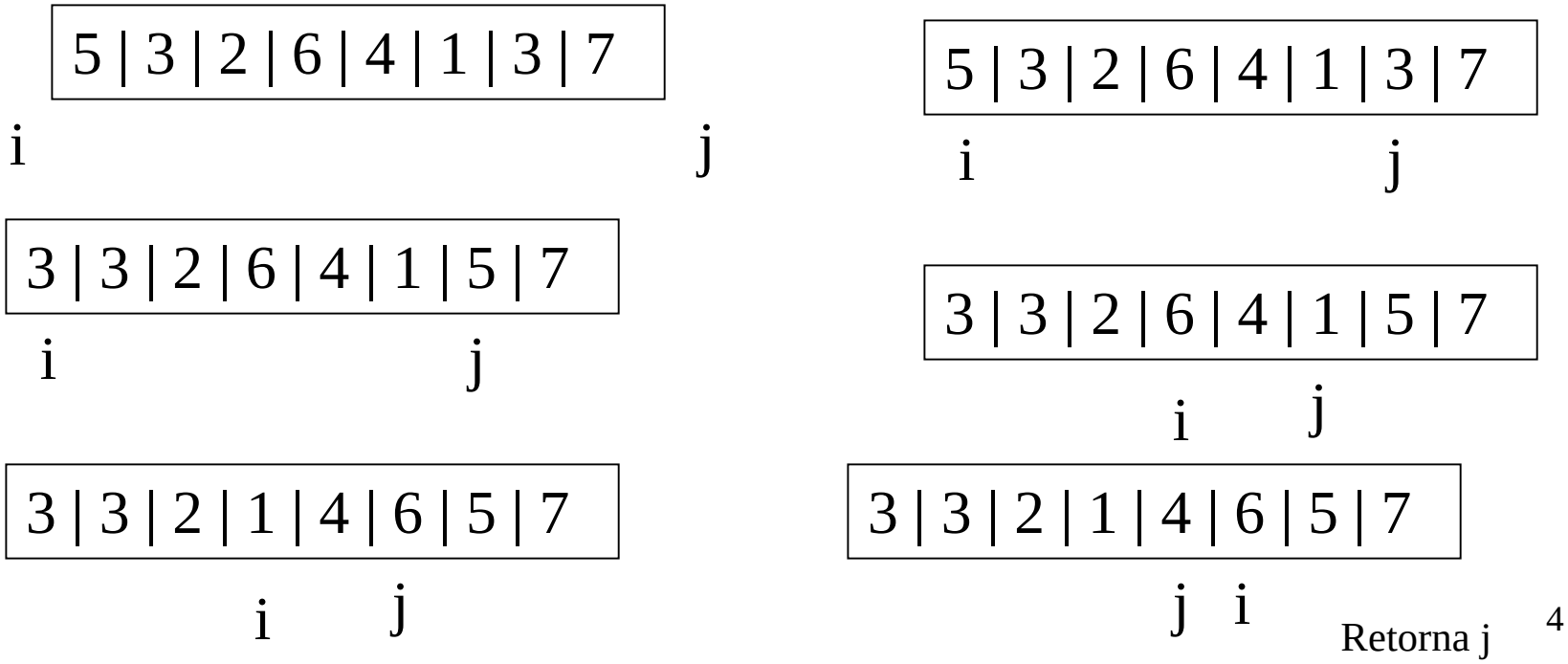
Partition

```
Quicksor(A,p,r){
        if (p<r) {
                q = Partition(A,p,r);
                Quicksort(A,p,q);
                Quicksort(A,q+1, r);
        }
    }
```

```
Partition(A,p,r) {
    x=A[p];
    i=p-1;
    j=r+1;
    while(1) {
        do   j--;
        until (A[j] ≤ x)
        do  i++;
        until (A[i] ≥ x)
        if (i < j)
                exchange A[i] <-> A[j];
        else
                return j;
    }
}
```

Explicación

- Partition efectúa una tarea simple: poner los elementos menores que A[p] en la región inferior del arreglo y los elementos mayores.
- i parte del extremo inferior del arreglo y j parte del extremo superior.
- Cada puntero avanza hacia el extremo opuesto buscando elementos que deba mover hacia el otro extremo.
- Cuando estos elementos son encontrados, son intercambiados.
- Si estos elementos no son encontrados, i terminara por igualar j.



Observaciones

- Los índices i y j nunca acceden al arreglo fuera de rango.
- A[p] debe ser el “pivote”. Si se tomara A[r] y ocurre que este es el mayor valor, partition retornaría q=r y quicksort estaría en un loop infinito.

Rendimiento de Quicksort:

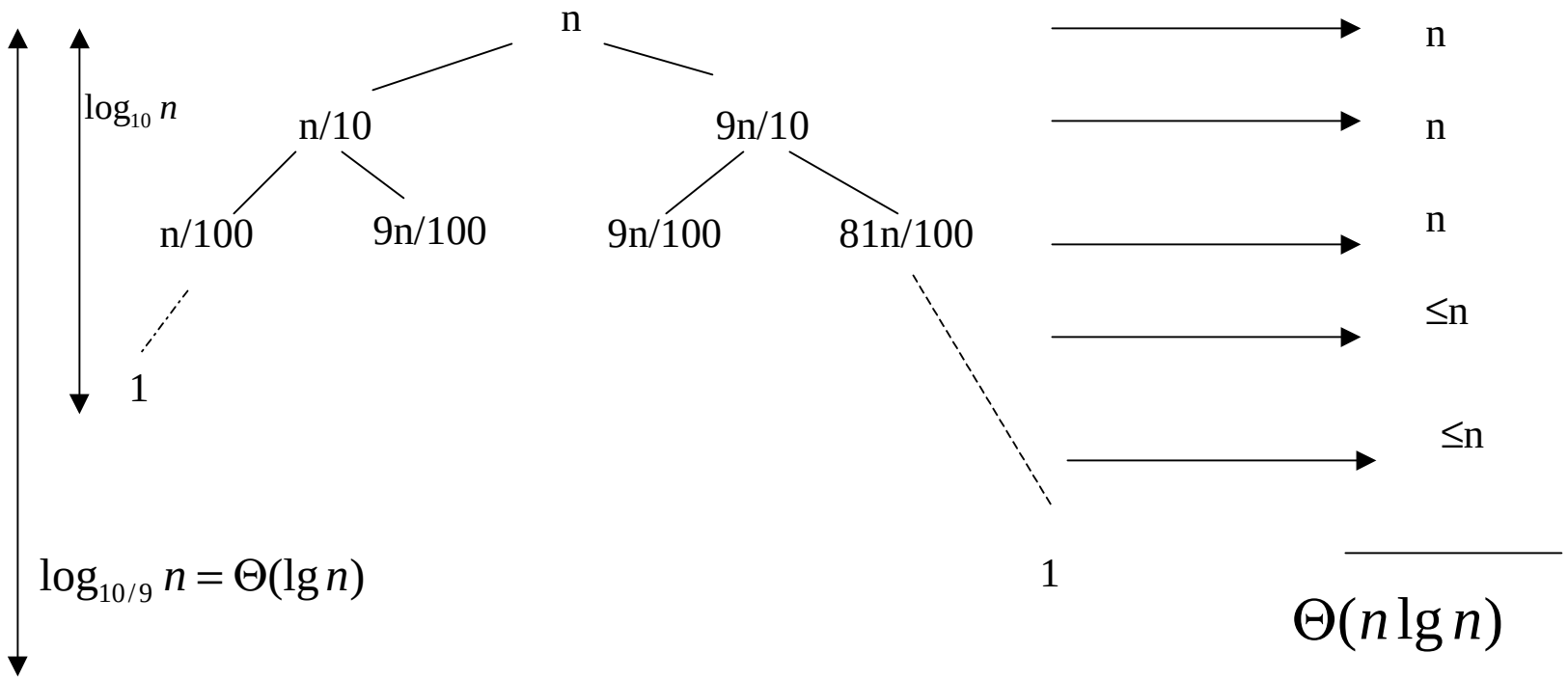
- Partition tiene costo $\Theta(n)$.
- Partición de peor caso
  Ocurre cuando el arreglo es particionado en n-1 y 1 elemento cada vez.
  En este caso se tiene: $T(n)=T(n-1) + \Theta(n) =$

$$\sum_{k=1}^{n} \Theta(k) = \Theta\left(\sum_{k=1}^{n} k \right) = \Theta(n^2)$$

- Partición de mejor caso:
  Ocurre cuando partition produce dos regiones de igual tamaño.
  En este caso se tiene: $T(n)=2T(n/2)+ \Theta(n)$,
  de lo cual resulta $T(n) = \Theta(n \lg n)$ (caso 2 teorema maestro)

Observaciones

- Si consideramos que la partición conduce a arreglos desbalanceados en proporción 9 : 1, de todas formas el algoritmo tiene tiempo θ(n lg n).
- En este caso T(n) = T(9n/10) +T(n/10) + Θ(n)



El caso promedio también conduce a tiempos θ(n lg n).

Versión “Aleatorizada” de Quicksort

- Objetivo: Lograr buen desempeño aun cuando la entrada no sea aleatoria.
- Idea: Tomar aleatoriamente un elemento dentro del arreglo y usarlo como pivote.
- Randomized_Partition(A,p,r) {

```
        i = random(p,r); /* retorna un valor en el rango [p,r] */
        eschange A[p] <--> A[i];
        return Partition(A, p, r);
}
```

- Ramdomized_Quicksort(A,p,r) {

```
        if (p < r) {
                q = Randomized_Partition(A,p,r);
                Randimized_Quicksort(A, p, q);
                Randomized_Quicksort(A, q+1, r);
        }
}
```