Algoritmos y estructuras de datos

Quicksort

Francisco Javier Zaragoza Martínez

Universidad Autónoma Metropolitana Unidad Azcapotzalco
Departamento de Sistemas

25 de mayo de 2015

Ordenamiento interno

Problema abstracto

Dado un arreglo A con n elementos, se desea reacomodar sus elementos de modo que $A_0 \leq A_1 \leq \ldots \leq A_{n-1}$.

Problema concreto

Dado un arreglo `int a[MAX]` con `int n` elementos en las posiciones `a[0], ..., a[n-1]` se desea reacomodar sus elementos de modo que `a[0] <= a[1] <= ... <= a[n-1]`.

Ordenamiento interno

Un problema, mil soluciones

El problema de ordenamiento interno es uno de los más estudiados en la computación y para el que se conocen muchos algoritmos. Algunos son:

- Burbuja
- Inserción directa.
- Selección directa.
- *Quicksort*.
- Ordenamiento por mezcla.
- Ordenamiento por montículo.

Comparaciones

Todos estos algoritmos comparan los datos en el arreglo.
Existen algoritmos que no comparan los datos en el arreglo.

Selección directa

Ejemplo

3 1 4 1 5 9 2 6 5 3 5 8 9 7 9 3

Selección directa

Ejemplo

3	1	4	1	5	9	2	6	5	3	5	8	9	7	9	3
3	1	4	1	5	9	2	6	5	3	5	8	9	7	3	**9**

Selección directa

Ejemplo

3	1	4	1	5	9	2	6	5	3	5	8	9	7	9	3
3	1	4	1	5	9	2	6	5	3	5	8	9	7	3	**9**
3	1	4	1	5	9	2	6	5	3	5	8	3	7	**9**	**9**

Selección directa

Ejemplo

3	1	4	1	5	9	2	6	5	3	5	8	9	7	9	3
3	1	4	1	5	9	2	6	5	3	5	8	9	7	3	**9**
3	1	4	1	5	9	2	6	5	3	5	8	3	7	**9**	**9**
3	1	4	1	5	7	2	6	5	3	5	8	3	**9**	**9**	**9**

Selección directa

Ejemplo

3	1	4	1	5	9	2	6	5	3	5	8	9	7	9	3
3	1	4	1	5	9	2	6	5	3	5	8	9	7	3	**9**
3	1	4	1	5	9	2	6	5	3	5	8	3	7	**9**	**9**
3	1	4	1	5	7	2	6	5	3	5	8	3	**9**	**9**	**9**
3	1	4	1	5	7	2	6	5	3	5	3	**8**	**9**	**9**	**9**

Selección directa

Ejemplo

3	1	4	1	5	9	2	6	5	3	5	8	9	7	9	3
3	1	4	1	5	9	2	6	5	3	5	8	9	7	3	**9**
3	1	4	1	5	9	2	6	5	3	5	8	3	7	**9**	**9**
3	1	4	1	5	7	2	6	5	3	5	8	3	**9**	**9**	**9**
3	1	4	1	5	7	2	6	5	3	5	3	**8**	**9**	**9**	**9**
3	1	4	1	5	3	2	6	5	3	5	**7**	**8**	**9**	**9**	**9**

Selección directa

Ejemplo

3	1	4	1	5	9	2	6	5	3	5	8	9	7	9	3
3	1	4	1	5	9	2	6	5	3	5	8	9	7	3	**9**
3	1	4	1	5	9	2	6	5	3	5	8	3	7	**9**	**9**
3	1	4	1	5	7	2	6	5	3	5	8	3	**9**	**9**	**9**
3	1	4	1	5	7	2	6	5	3	5	3	**8**	**9**	**9**	**9**
3	1	4	1	5	3	2	6	5	3	5	**7**	**8**	**9**	**9**	**9**
3	1	4	1	5	3	2	5	5	3	**6**	**7**	**8**	**9**	**9**	**9**

Selección directa

Ejemplo

3	1	4	1	5	9	2	6	5	3	5	8	9	7	9	3
3	1	4	1	5	9	2	6	5	3	5	8	9	7	3	**9**
3	1	4	1	5	9	2	6	5	3	5	8	3	7	**9**	**9**
3	1	4	1	5	7	2	6	5	3	5	8	3	**9**	**9**	**9**
3	1	4	1	5	7	2	6	5	3	5	3	**8**	**9**	**9**	**9**
3	1	4	1	5	3	2	6	5	3	5	**7**	**8**	**9**	**9**	**9**
3	1	4	1	5	3	2	5	5	3	**6**	**7**	**8**	**9**	**9**	**9**
3	1	4	1	5	3	2	5	3	**5**	**6**	**7**	**8**	**9**	**9**	**9**

Selección directa

Ejemplo

3	1	4	1	5	9	2	6	5	3	5	8	9	7	9	3
3	1	4	1	5	9	2	6	5	3	5	8	9	7	3	**9**
3	1	4	1	5	9	2	6	5	3	5	8	3	7	**9**	**9**
3	1	4	1	5	7	2	6	5	3	5	8	3	**9**	**9**	**9**
3	1	4	1	5	7	2	6	5	3	5	3	**8**	**9**	**9**	**9**
3	1	4	1	5	3	2	6	5	3	5	**7**	**8**	**9**	**9**	**9**
3	1	4	1	5	3	2	5	5	3	**6**	**7**	**8**	**9**	**9**	**9**
3	1	4	1	5	3	2	5	3	**5**	**6**	**7**	**8**	**9**	**9**	**9**
3	1	4	1	5	3	2	3	**5**	**5**	**6**	**7**	**8**	**9**	**9**	**9**

Selección directa

Ejemplo

3	1	4	1	5	9	2	6	5	3	5	8	9	7	9	3
3	1	4	1	5	9	2	6	5	3	5	8	9	7	3	**9**
3	1	4	1	5	9	2	6	5	3	5	8	3	7	**9**	**9**
3	1	4	1	5	7	2	6	5	3	5	8	3	**9**	**9**	**9**
3	1	4	1	5	7	2	6	5	3	5	3	**8**	**9**	**9**	**9**
3	1	4	1	5	3	2	6	5	3	5	**7**	**8**	**9**	**9**	**9**
3	1	4	1	5	3	2	5	5	3	**6**	**7**	**8**	**9**	**9**	**9**
3	1	4	1	5	3	2	5	3	**5**	**6**	**7**	**8**	**9**	**9**	**9**
3	1	4	1	5	3	2	3	**5**	**5**	**6**	**7**	**8**	**9**	**9**	**9**
3	1	4	1	3	3	2	**5**	**5**	**5**	**6**	**7**	**8**	**9**	**9**	**9**

Selección directa

Ejemplo

3	1	4	1	5	9	2	6	5	3	5	8	9	7	9	3
3	1	4	1	5	9	2	6	5	3	5	8	9	7	3	**9**
3	1	4	1	5	9	2	6	5	3	5	8	3	7	**9**	**9**
3	1	4	1	5	7	2	6	5	3	5	8	3	**9**	**9**	**9**
3	1	4	1	5	7	2	6	5	3	5	3	**8**	**9**	**9**	**9**
3	1	4	1	5	3	2	6	5	3	5	**7**	**8**	**9**	**9**	**9**
3	1	4	1	5	3	2	5	5	3	**6**	**7**	**8**	**9**	**9**	**9**
3	1	4	1	5	3	2	5	3	**5**	**6**	**7**	**8**	**9**	**9**	**9**
3	1	4	1	5	3	2	3	**5**	**5**	**6**	**7**	**8**	**9**	**9**	**9**
3	1	4	1	3	3	2	**5**	**5**	**5**	**6**	**7**	**8**	**9**	**9**	**9**
3	1	2	1	3	3	**4**	**5**	**5**	**5**	**6**	**7**	**8**	**9**	**9**	**9**

Selección directa

Idea más general

Tenemos un arreglo desordenado `a[0], a[1], ..., a[j]` en el que queremos encontrar el máximo `a[i]` e intercambiarlo por `a[j]`.

```
void maximo(int j, int a[])
{
  int i = j, k;

  for (k = j-1; k >= 0; k--)
    if (a[k] > a[i])
      i = k;
  intercambia(&a[i], &a[j]);
}
```

Selección directa

Iterativo

```
void seleccion(int n, int a[])
{
  int j;

  for (j = n-1; j > 0; j--)
    maximo(j, a);
}
```

Selección directa

Iterativo

```
void seleccion(int n, int a[])
{
  int j;

  for (j = n-1; j > 0; j--)
    maximo(j, a);
}
```

Comparaciones

La llamada a `maximo(j,a)` hace `j` comparaciones. En total se hacen $1 + 2 + \cdots + (n-1) = \frac{1}{2}n(n-1)$ comparaciones.

Selección directa

Recursivo

```
void seleccion(int n, int a[])
{
  if (n > 1) {
    maximo(n-1, a);
    seleccion(n-1, a);
  }
}
```

Selección directa

Recursivo

```
void seleccion(int n, int a[])
{
  if (n > 1) {
    maximo(n-1, a);
    seleccion(n-1, a);
  }
}
```

Comparaciones

Si la llamada a `seleccion(n,a)` hace $T(n)$ comparaciones, entonces $T(1) = 0$ y $T(n) = T(n-1) + (n-1)$ si $n > 1$. En total se hacen $(n-1) + \cdots + 2 + 1 = \frac{1}{2}n(n-1)$ comparaciones.

Selección directa

Ejercicios

1. Reescribe `maximo` con apuntadores.
2. Reescribe `seleccion` para que ordene decrecientemente.
3. Reescribe `seleccion` para que inicie con el elemento más pequeño.
4. Reescribe `seleccion` como dos ciclos anidados.
5. ¿Cuántas veces se lee del arreglo `a`?
6. ¿Cuántas veces se escribe en el arreglo `a`?

Quicksort

Ejemplo

3 1 4 1 5 9 2 6 5 3 5 8 9 7 9 3

Quicksort

Ejemplo

3	1	4	1	5	9	2	6	5	3	5	8	9	7	9	3
3	1	1	2	3	**3**	4	5	9	6	5	5	8	9	7	9

Quicksort

Ejemplo

3	1	4	1	5	9	2	6	5	3	5	8	9	7	9	3
3	1	1	2	3	**3**	4	5	9	6	5	5	8	9	7	9
	¿		?		**3**										

Quicksort

Ejemplo

3	1	4	1	5	9	2	6	5	3	5	8	9	7	9	3
3	1	1	2	3	**3**	4	5	9	6	5	5	8	9	7	9
	¿		?		**3**										
1	1	2	3	3	**3**										

Quicksort

Ejemplo

3	1	4	1	5	9	2	6	5	3	5	8	9	7	9	3
3	1	1	2	3	**3**	4	5	9	6	5	5	8	9	7	9
	¿		?		**3**										
1	1	2	3	3	**3**										
					3	4	5	9	6	5	5	8	9	7	9

Quicksort

Ejemplo

3	1	4	1	5	9	2	6	5	3	5	8	9	7	9	3
3	1	1	2	3	**3**	4	5	9	6	5	5	8	9	7	9
	¿		?		**3**										
1	1	2	3	3	**3**										
					3	4	5	9	6	5	5	8	9	7	9
					3				¿		?				

Quicksort

Ejemplo

3	1	4	1	5	9	2	6	5	3	5	8	9	7	9	3
3	1	1	2	3	**3**	4	5	9	6	5	5	8	9	7	9
	¿		?		**3**										
1	1	2	3	3	**3**										
					3	4	5	9	6	5	5	8	9	7	9
					3				¿		?				
					3	4	5	5	5	6	7	8	9	9	9

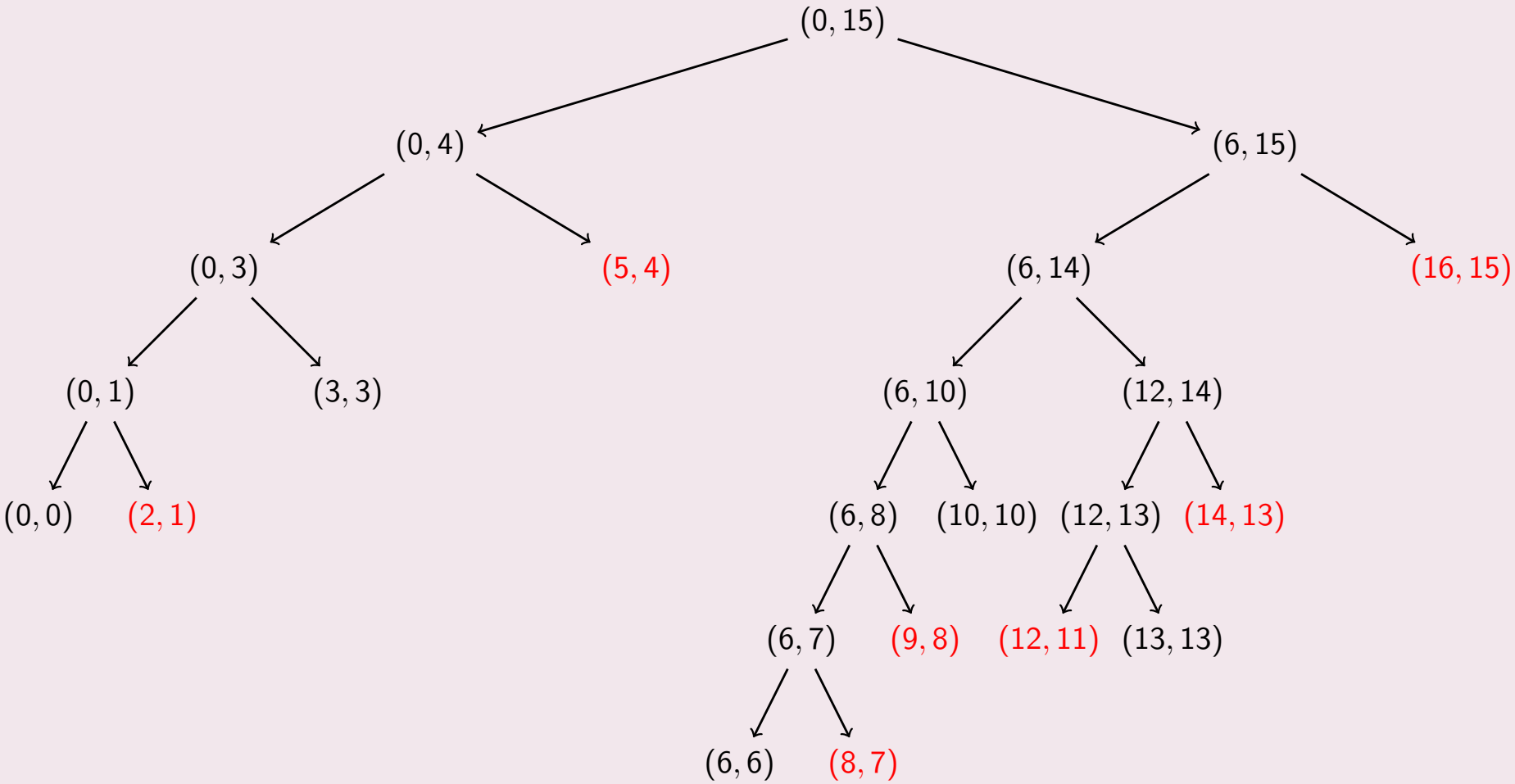
Quicksort

Ejemplo

3	1	4	1	5	9	2	6	5	3	5	8	9	7	9	3
3	1	1	2	3	**3**	4	5	9	6	5	5	8	9	7	9
	¿		?		**3**										
1	1	2	3	3	**3**										
					3	4	5	9	6	5	5	8	9	7	9
					3				¿		?				
					3	4	5	5	5	6	7	8	9	9	9
1	1	2	3	3	**3**	4	5	5	5	6	7	8	9	9	9

Quicksort

Árbol de recursión



Quicksort

Colocar el pivote en su lugar

Tenemos un arreglo desordenado `a[i], ..., a[j]` en el que queremos buscar el lugar `p` del elemento `a[j]` (el pivote). Además queremos reacomodar el arreglo de modo que el pivote quede en `a[p]`, los menores en `a[i], ..., a[p-1]` y los mayores en `a[p+1], ..., a[j]`.

```
int pivote(int i, int j, int a[])
{
  int k, p=i-1;

  for (k = i; k <= j; k++)
    if (a[k] <= a[j])
      intercambia(&a[++p], &a[k]);
  return p;
}
```

Quicksort

Recursivo

Esta función deberá llamarse como `qsort(0, n-1, a)`.

```
void qsort(int i, int j, int a[])
{
  int p;

  if (i < j) {                /* dos o mas elementos   */
    p = pivote(i, j, a);      /* el pivote va en a[p]  */
    qsort(i, p-1, a);         /* ordena los menores    */
    qsort(p+1, j, a);         /* ordena los mayores    */
  }
}
```

Quicksort

Comparaciones del pivote

Con un vector de tamaño n, `pivote` hace n comparaciones.

Quicksort

Comparaciones del pivote

Con un vector de tamaño n, `pivote` hace n comparaciones.

Comparaciones de Quicksort

Si $T(n)$ es el número de comparaciones que hace `qsort` con un vector de tamaño n, entonces $T(1) = 0$ y $T(n) = T(p-1) + T(n-p) + n$.

Quicksort

Comparaciones del pivote

Con un vector de tamaño n, `pivote` hace n comparaciones.

Comparaciones de Quicksort

Si $T(n)$ es el número de comparaciones que hace `qsort` con un vector de tamaño n, entonces $T(1) = 0$ y $T(n) = T(p-1) + T(n-p) + n$. El problema es que no sabemos cuanto vale p.

Quicksort

Comparaciones del pivote

Con un vector de tamaño n, `pivote` hace n comparaciones.

Comparaciones de Quicksort

Si $T(n)$ es el número de comparaciones que hace `qsort` con un vector de tamaño n, entonces $T(1) = 0$ y $T(n) = T(p-1) + T(n-p) + n$. El problema es que no sabemos cuanto vale p.

Estimación

Si $p \approx n/2$ entonces $T(n) \approx 2T(n/2) + n$, de donde $T(n) \approx n \log_2 n$.

Quicksort

Comparación

Propiedad	Burbuja	Inserción	Selección	Mezcla	Quicksort
Memoria	n	n	n	$2n$	n
Peor tiempo	n^2	n^2	n^2	$n \log_2 n$	n^2
Mejor tiempo	n	n	n	$n \log_2 n$	$n \log_2 n$
Promedio	n^2	n^2	n^2	$n \log_2 n$	$n \log_2 n$

Quicksort

Comparación

Propiedad	Burbuja	Inserción	Selección	Mezcla	Quicksort
Memoria	n	n	n	$2n$	n
Peor tiempo	n^2	n^2	n^2	$n \log_2 n$	n^2
Mejor tiempo	n	n	n	$n \log_2 n$	$n \log_2 n$
Promedio	n^2	n^2	n^2	$n \log_2 n$	$n \log_2 n$

Otra comparación

Tiempo promedio para ordenar n datos si cada comparación tarda 1 ns.

n	Burbuja	Inserción	Selección	Mezcla	Quicksort
10^3	0.5 ms	0.5 ms	0.5 ms	0.01 ms	0.01 ms
10^4	50 ms	50 ms	50 ms	0.1 ms	0.1 ms
10^5	5 s	5 s	5 s	1 ms	1 ms
10^6	500 s	500 s	500 s	20 ms	20 ms

Quicksort

Ejercicios

1. Reescribe `pivote` usando apuntadores.
2. Escribe una función `seleccion(int i, int j, int k, int a[])` que coloque en `a[k]` al elemento que quedaría en esta posición si ordenaras `a[i]..a[j]`. ¡No ordenes el arreglo! Aprovecha `pivote`.