Análisis y Diseño de Algoritmos

Gilberto Gutiérrez

DCCTI/UBB

2015

Contenido

1. Técnicas de Programación
   - Dividir para Reinar
     - Búsqueda Binaria
     - Mínimo y Máximo de un conjunto
     - Sort por mezcla
     - Multiplicación de enteros grandes
     - QuickSort

Dividir para Reinar

- Una de las técnicas más importante y más utilizada para el diseño de algoritmos.
- Consiste en descomponer un problema de tamaño n en problemas más pequeños, de modo que a partir de la solución de dichos problemas sea posible construir con facilidad una solución al problema original.

Esquema General

1: $DV(X)$
2: **if** X es suficientemente pequeño **then**
3: Retornar Ad-Hoc(X)
4: **else**
5: Descomponer X en subproblemas más pequeños $X_1, X_2, \ldots, X_k$
6: **for** $i = 1; i \leq k; i++$ **do**
7: $Y_i \leftarrow \text{DV}(X_i)$
8: **end for**
9: Combinar las soluciones Y_i para obtener una solución Y para X
10: **return** Y
11: **end if**

Búsqueda Binaria

Algoritmo

Suponemos un arreglo A ordenado de menor a mayor.

1: **BusquedaBinaria(**A, li, ls, x**))**
2: **if** $li > ls$ **then**
3: **return** -1 $\{x$ no se encuentra en el arreglo $A\}$
4: **else**
5: Sea $pivote = \frac{li+ls}{2}$
6: **if** $A[pivote] = x$ **then**
7: **return** $pivote$
8: **else if** $x < A[pivote]$ **then**
9: **return** BusquedaBinaria$(A, li, pivote - 1, x)$
10: **else**
11: **return** BusquedaBinaria$(A, pivote + 1, ls, x)$
12: **end if**
13: **end if**

Mínimo y Máximo de un conjunto

Algoritmo

1: **MinMax(S)**
2: **if** $\|S\| = 1$ **then**
3: **return** (a, a) { $S = \{a\}$ }
4: **else if** $\|S\| = 2$ **then**
5: **return** $(\min(a, b), \max(a, b))$ $\{S = \{a, b\}\}$
6: **else**
7: Dividir S en dos subconjuntos S_1, S_2
8: (n_1, m_1) =MinMax(S_1)
9: (n_2, m_2) =MinMax(S_2)
10: **return** $(\min(n_1, n_2), \max(m_1, m_2))$
11: **end if**

Sort por mezcla

Algoritmo

1: *MergeSort*($T[1 \ldots n]$)
2: **if** n es suficientemente pequeño **then**
3: *Ad-Hoc*(T)
4: **else**
5: Sea $U[1 \ldots 1 + \lfloor \frac{n}{2} \rfloor], V[1 \ldots 1 + \lfloor \frac{n}{2} \rfloor]$
6: $U[1 \ldots \lfloor \frac{n}{2} \rfloor] \leftarrow T[1 \ldots \lfloor \frac{n}{2} \rfloor]$
7: $V[1 \ldots \lfloor \frac{n}{2} \rfloor] \leftarrow T[1 + \lfloor \frac{n}{2} \rfloor \ldots n]$
8: *MergeSort*($U[1 \ldots \lfloor \frac{n}{2} \rfloor]$)
9: *MergeSort*($V[1 \ldots \lfloor \frac{n}{2} \rfloor]$)
10: *Merge*(U, V, T)
11: **end if**

Sort por Mezcla (continuación)

1: $Merge(U[1 \ldots m+1], V[1 \ldots n+1], T[1 \ldots n+m])$ $\{U[m+1]$ y $V[n+1]$ se usan como centinelas para facilitar algoritmo de mezcla$\}$
2: $i \leftarrow j \leftarrow 1$
3: $U[m+1] \leftarrow V[n+1] \leftarrow \infty$
4: **for** $k \leftarrow 1$ hasta $m+n$ **do**
5: **if** $U[i] < V[j]$ **then**
6: $T[k] \leftarrow U[i]$
7: $i \leftarrow i+1$
8: **else**
9: $T[k] \leftarrow V[j]$
10: $j \leftarrow j+1$
11: **end if**
12: **end for**

Multiplicación de enteros grandes

1: $Mult(X, Y, n)$ {X e Y enteros con signo $\leq 2^n$. n potencia de 2}
2: s {contiene el signo de XY}
3: $m1, m2, m3$ {contiene los 3 productos }
4: A, B, C, D {contiene las mitades izquierda derecha de X e Y}
5: $s = Sign(X) * Sign(Y)$
6: $X = abs(X)$
7: $Y = abs(Y)$
8: **if** $n = 1$ **then**
9: **if** $X = 1$ and $Y = 1$ **then**
10: **return** s
11: **else**
12: **return** 0
13: **end if**
14: **else**
15: $A = \frac{n}{2}$ bits izquierdo de X
16: $B = \frac{n}{2}$ bits derechos de X
17: $C = \frac{n}{2}$ bits izquierdo de Y
18: $D = \frac{n}{2}$ bits derechos de Y
19: $m1 = Mult(A, C, \frac{n}{2})$
20: $m2 = Mult(A - B, D - C, \frac{n}{2})$
21: $m3 = Mult(B, D, \frac{n}{2})$
22: **return** $(a * (m1 * 2^n + (m1 + m2 + m3) * 2^{\frac{n}{2}} + m3))$
23: **end if**

Procedimiento pivote (partición)

1: *Pivote*($T[i \ldots j]$) {Intercambia los elementos del arreglo $T[i \ldots j]$ y retorna un valor l tal que, $i \leq l \leq j$, $T[k] \leq p \; \forall i \leq k < l$, $T[l] = p$ y $T[k] > p \; \forall l < k \leq j$, y p es el valor inicial de $T[i]$}
2: $p \leftarrow T[i]; k \leftarrow i; l \leftarrow j + 1$
3: **repeat**
4: $\quad k \leftarrow k + 1$
5: **until** $T[k] > p$ or $k \geq j$
6: **repeat**
7: $\quad l \leftarrow l - 1$
8: **until** $T[k] \leq p$
9: **while** $k < l$ **do**
10: $\quad$ Intercambiar($T[k]$, $T[l]$)
11: $\quad$ **repeat**
12: $\quad\quad k \leftarrow k + 1$
13: $\quad$ **until** $T[k] > p$
14: $\quad$ **repeat**
15: $\quad\quad l \leftarrow l - 1$
16: $\quad$ **until** $T[k] \leq p$
17: **end while**
18: Intercambiar($T[i]$, $T[l]$)
19: **return** l

Procedimiento QuickSort

1: *quicksort*($T[i \ldots j]$) {Ordena elmentos del arreglo $T[i \ldots j]$ de menor a mayor}
2: **if** $j - i$ es suficientemente pequeño **then**
3: insertar($T[i \ldots j]$)
4: **else**
5: l =pivote($T[i \ldots j]$)
6: quicksort($T[i \ldots l-1]$)
7: quicksort($T[l+1 \ldots j]$)
8: **end if**

Example

Arreglo a ordenar

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

El arreglo se particiona tomando $p = 3$

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

Se busca el primer elemento mayor que el pivote y el último no mayor que el pivote

3	1	**$\underline{4}$**	1	5	9	2	6	5	**$\overline{3}$**	5	8	9

Se intercambian los elementos

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

Se vuelve a explorar

3	1	3	1	**$\underline{5}$**	9	**$\overline{2}$**	6	5	4	5	8	9

Se intercambian

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

Se explora (los índices se cruzan)

3	1	3	1	**$\overline{2}$**	**$\underline{9}$**	5	6	5	4	5	8	9

Se intercambia el pivote con el elemento superrayado

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