Divide-and-Conquer I

Guillermo Palma

Universidad Simón Bolívar
Departamento de Computación y T.I.

CI-2612: Algoritmos y Estructuras II

Plan

La técnica Divide-and-Conquer

Es una técnica de diseño de algoritmos que consiste en dividir una instancia de un problema en subinstancias, para obtener las soluciones de las mismas, para entonces combinarlas para encontrar la solución de la instancia original.

Partes de Divide-and-Conquer

- Se **divide** un problema de tamaño n en varias partes o subproblemas. Generalmente dos partes iguales de tamaño $\frac{n}{2}$
- Se resuelven (**conquistan**) los subproblemas recursivamente. Generalmente se resuelven dos partes recursivamente.
- Se **combinan** las soluciones de los subproblemas para obtener la solución del problema original. Generalmente se combinan las dos soluciones en $O(n)$ para lograr la solución final.

Esquema general de Divide-and-Conquer

Función Divide-and-Conquer(x)

Entrada: Una instancia x de un problema.
Salida : Una solución y de la instancia x.
inicio
  si x *es suficientemente pequeña* **entonces**
    retornar `adhoc(x)` ;
  Descomponer x en instancias más pequeñas $x_1, x_2, .., x_l$;
  para i **a** l **hacer**
    $y_i \leftarrow$ `Divide-and-Conquer(x_i)` ;
  Combinar las y_i soluciones para obtener la solución y de x ;
  retornar y ;

Recurrencias

Relación de recurrencia

Es una ecuación que recursivamente define una secuencia que se caracteriza por dar el término actual, en función de los términos anteriores.

Ejemplos de recurrencias

- $$T(n) = \begin{cases} 0 & \text{si } n = 0 \\ 3T(n \div 2) + n & \text{de lo contrario} \end{cases}$$
- Fibonacci
$$f_n = \begin{cases} n & \text{si } n = 0 \lor n = 1 \\ f_{n-1} + f_{n-2} & \text{de lo contrario} \end{cases}$$

Observaciones del esquema de Divide-and-Conquer

- Para una instancia de tamaño n, cada una de las l subinstancias deberían ser de tamaño n/b para algún entero b.
- Sea $g(n)$ el tiempo `Divide-and-Conquer` para una instancia de tamaño n. sin contar la llamada recursiva. El tiempo de ejecución de este esquema viene dado por la recurrencia:

$$t(n) = lt(n \div b) + g(n).$$

- Si existe un entero k, tal que $g(n) = \Theta(n^k)$, entonces:

$$t(n) = \begin{cases} \Theta(n^k) & \text{si } l < b^k \\ \Theta(n^k \log n) & \text{si } l = b^k \\ \Theta(n^{\log_b l}) & \text{si } l > b^k. \end{cases}$$

Búsqueda Binaria

Planteamiento del problema

Dado un arreglo $T[1..n]$ ordenado de forma ascendente y un elemento x, se desea la posición k de x, en caso de que x se encuentre en el arreglo, o en que posición k debería ser insertado en caso de no pertenecer al arreglo.

Ejemplo: Dado el arreglo:

1	2	3	4	5	6	7	8	9
-1	-2	0	2	3	5	6	8	9

- Si $x = 3$ entonces $k = 5$
- Si $x = 1$ entonces $k = 4$

Búsqueda Binaria

Función BusquedaBinaria(T, x, n)

inicio
  si $n = 0 \vee x > T[n]$ **entonces** **retornar** $n + 1$;
  en otro caso **retornar** `BusquedaBinRec(`T, 1, n, x`)` ;

Función BusquedaBinRec(T, i, j, x)

inicio
  si $i = j$ **entonces** **retornar** i ;
  $k \leftarrow (i + j) \div 2$;
  si $x \leq T[k]$ **entonces**
    retornar `BusquedaBinRec(`T, i, k, x`)` ;
  en otro caso
    retornar `BusquedaBinRec(`T, $k + 1$, j, x`)` ;

Llamada: `BusquedaBinaria`(T, x, n).

Ejemplo de Búsqueda Binaria

1	2	3	4	5	6	7	8	9	10	11	
-5	-2	0	3	8	8	9	12	12	26	31	$x \leq T[k]$?
i					k					j	no
						i		k		j	sí
						i	k	j			sí
						ik	j				no
							ij				$i = j$: alto

Figura: Búsqueda binaria con llamada `BusquedaBinaria`(T, 12, 11). Ejemplo tomado de [1]

Análisis del peor caso de la Búsqueda Binaria

- La búsqueda binaria hace una llamada recursiva con $T\lfloor n/2 \rfloor$ o con $T\lceil n/2 \rceil$ elementos
- Recurrencia $t(n) = t(n/2) + g(n)$
- $g(n) = \Theta(1) = \Theta(n^0)$
- $l = 1$, $b = 2$ y $k = 0$
- Por lo tanto el tiempo es: $t(n) = \Theta(\log n)$

Problema de Ordenamiento

Planteamiento del problema

Dada una secuencia de elementos almacenadas en un arreglo $T[1..n]$, se quiere ordenar los elementos de las secuencia en orden ascendente.

Mergesort

- Se dividen en dos partes iguales (o casi) los elementos del arreglo
- Se conquista ordenando estas partes por medio de llamadas recursivas.
- Se combinan (mezclan) las soluciones de cada parte conservando el orden ascendente.

Procedimiento de mezcla de dos arreglos ordenados

Procedimiento merge(U, V, T, m, n)

inicio

```
/* Se mezclan los arreglos U[1..m], V[1..n] en el
   arreglo T[1..n+m]                                   */
i, j ← 1 ;
U[m+1], V[n+1] ← ∞ ; /* Centinelas                     */
para k ← 1 a m+n hacer
    si U[i] < V[j] entonces
        T[k] ← U[i] ;
        i ← i+1
    en otro caso
        T[k] ← V[j] ;
        j ← j+1
```

Ejemplo del procedimiento de mezcla

$V[1..4]$:

3	7	14	∞

$U[1..4]$:

2	11	16	∞

$T[1..6]$:

3	7	14	2	11	16

Ejemplo del procedimiento de mezcla

$V[1..4]$:

3	7	14	∞

$U[1..4]$:

2	11	16	∞

$T[1..6]$:

2	7	14	2	11	16

Ejemplo del procedimiento de mezcla

$V[1..4]$:

3	7	14	∞

$U[1..4]$:

2	11	16	∞

$T[1..6]$:

2	3	14	2	11	16

Ejemplo del procedimiento de mezcla

$V[1..4]$:

3	7	14	∞

$U[1..4]$:

2	11	16	∞

$T[1..6]$:

2	3	7	2	11	16

Ejemplo del procedimiento de mezcla

$V[1..4]$:

3	7	14	∞

$U[1..4]$:

2	11	16	∞

$T[1..6]$:

2	3	7	11	11	16

Mergesort

Ejemplo del procedimiento de mezcla

$V[1..4]$:

3	7	14	∞

$U[1..4]$:

2	11	16	∞

$T[1..6]$:

2	3	7	11	14	16

Ejemplo del procedimiento de mezcla

$V[1..4]$:

3	7	14	∞

$U[1..4]$:

2	11	16	∞

$T[1..6]$:

2	3	7	11	14	16

Procedimiento Mergesort

Procedimiento mergesort(T, n)

inicio

  si *n es "pequeña"* **entonces**

    `insertionsort` (T, n)

  en otro caso

    array $U[1..1 + \lfloor n/2 \rfloor]$;

    array $V[1..1 + \lceil n/2 \rceil]$;

    $U[1..\lfloor n/2 \rfloor] \leftarrow T[1..\lfloor n/2 \rfloor]$;

    $V[1..\lceil n/2 \rceil] \leftarrow T[1 + \lfloor n/2 \rfloor..n]$;

    `mergesort` (U, $\lfloor n/2 \rfloor$) ;

    `mergesort` (V, $\lceil n/2 \rceil$) ;

    `merge` (U, V, T, $\lceil n/2 \rceil$, $\lfloor n/2 \rfloor$) ;

Ejemplo del procedimiento Mergesort

A	L	G	O	R	I	T	H	M	S	
A	L	G	O	R	I	T	H	M	S	divide
A	G	L	O	R	H	I	M	S	T	sort
A	G	H	I	L	M	O	R	S	T	merge

Figura: Ejemplo tomado de [2]

Análisis del peor caso de Mergesort

- Recurrencia de Mergesort: $t(n) = t(\lceil n/2 \rceil) + t(\lfloor n/2 \rfloor) + g(n)$
- El procedimiento `merge` es $\Theta(n)$
- Recurrencia de Mergesort: $t(n) = t(\lceil n/2 \rceil) + t(\lfloor n/2 \rfloor) + \Theta(n)$
- Se tiene que $l = 2$, $b = 2$ y $k = 1$
- Como $l = b^k$ se aplica el segundo caso, esto es $t(n) = \Theta(n \log n)$
- Por lo tanto Mergesort es $t(n) = \Theta(n \log n)$ en el peor caso.

Problemas vistos

- Esquema general de Divide-and-Conquer.
- Búsqueda Binaria
- Mergesort

Referencias

G. Brassard and P. Bratley.
Fundamentals of Algorithmics.
Prentice Hall, 1996.

Jon Kleinberg and Eva Tardos.
Algorithm design.
Pearson Education, 2006.