Tema 4: Recurrencias

Ing. Margot Edith Cuarán Jaramillo

Escuela de Ingeniería de Sistemas y Computación
Universidad del Valle - Santiago de Cali, Colombia
e–mail: mecuaran@eisc.univalle.edu.co
Agosto 2.006

Definición recursiva

Las funciones se pueden definir recursivamente. La forma más simple de definiciones recursivas de una función f sobre los números naturales especifica:

Regla básica El valor de $f(0)$

Regla de recursión
Cómo obtener $f(n)$ a partir de $f(n-1)$, $\forall n \geq 1$

Ejemplo 1: Factorial de n (n!)

Básica $f(0) = 0! = 1$
Recursión $f(n) = n.(n\text{-}1)!$

Calcule $f(5)$.

En general, un algoritmo recursivo se plantea él mismo un problema con la misma estructura del inicial, pero de tamaño menor.

```
int factorial(int num){
... if (num==0)
..... return 1;
... else
..... return num*factorial(num-1); }
```

Haga el proceso de ejecución de la función factorial, con n=5.

$$
\begin{aligned}
factorial(5) &= \overbrace{5 * factorial(4)}^{} \\
&\Downarrow \\
&= 5 * \overbrace{4 * factorial(3)}^{} \\
&\Downarrow \\
&= 5 * 4 * \overbrace{3 * factorial(2)}^{} \\
&\Downarrow \\
&= 5 * 4 * 3 * \overbrace{2 * factorial(1)}^{} \\
&\Downarrow \\
&= 5 * 4 * 3 * 2 * \overbrace{1 * factorial(0)}^{} \\
&\Downarrow \\
&= 5 * 4 * 3 * 2 * 1 * 1
\end{aligned}
$$

Tiempo de ejecución del algoritmo $T_A(n)$ (*tiempo que se demora el algoritmo A, en el peor de los casos, para encontrar una solución a un problema de tamaño n*).

$$T(num) = \begin{cases} T_k, & num=0 \\ T_k + T(num-1), & num > 0 \end{cases}$$

Solución

$$
\begin{aligned}
T(num) &= T_k + T(num-1) \\
&= T_k + T_k + T(num-2) \\
&= T_k + T_k + T_k + T(num-3) \\
&= \ldots \\
&= num * T_k + T(0) \\
&= T_k * (num+1)
\end{aligned}
$$

Se puede concluir que T(num es O(num+1) $\Rightarrow$ T(num) es O(num)

Ejemplo 2: Mezcla de dos listas ordenadas

A= | 2 | 3 | 5 | 6 |

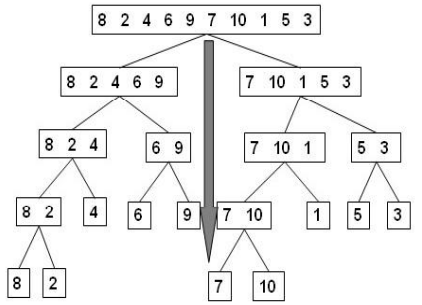
B= | 1 | 4 |

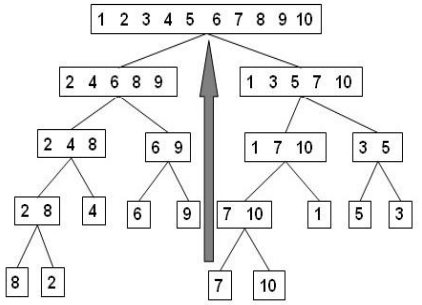
A	B	Fusión	Comparación
2,3,5,6	1,4		$1 < 2$
2,3,5,6	4	1	$2 < 4$
3,5,6	4	1,2	$3 < 4$
5,6	4	1,2,3	$4 < 5$
5,6		1,2,3,4	
		1,2,3,4,5,6	

Tiempo de ejecución del algoritmo $T_A(n)$
Dos listas ordenadas con m y n elementos se pueden mezclar en una lista ordenada realizando a lo sumo m+n-1 comparaciones [2].

Ejemplo 3: Ordenación por mezcla

A= | 8 | 2 | 4 | 6 | 9 | 7 | 10 | 1 | 5 | 3 |





Tiempo de ejecución del algoritmo $T_A(n)$

- n es el número de elementos de la listra. Por conveniencia, sea n potencia de 2 (2^m)
- Partición 1: 2 listas de 2^{m-1} elementos cada una. (Primer nivel del árbol)
- Partición 2: 4 listas de 2^{m-2} elementos cada una. (Segundo nivel del árbol)
- En general, 2^{k-1} listas de 2^{m-k+1} elementos cada una. (k-1 nivel del árbol)
- Al final, 2^m listas de un elemento cada una. (m nivel del árbol)

La mezcla combina pares de 2^m listas de un elemento en 2^{m-1} listas (nivel m-1). La mezcla de cada par requiere una comparación.

El número de comparaciones que requiere la ordenación por mezcla es a lo sumo:

$$\begin{aligned} NumComp &= \sum_{k=1}^{m} 2^{k-1}(2^{m-k+1}-1) \\ &= \sum_{k=1}^{m} 2^m - \sum_{k=1}^{m} 2^{k-1} \\ &= m2^m - (2^m - 1) \\ &= n.log(n) - n + 1 \end{aligned}$$

Sea $m = log(n)$ y $n = 2^m$

El número de comparaciones que necesita el algoritmo de ordenación por mezcla es $O(n.log(n))$

Clase Práctica

- Algoritmo recursivo para hallar a^n, donde a es un número real distinto de cero yu n es un entero no negativo.
- Algoritmo recursivo para calcular el máximo común divisor de dos números no negativos a y b, $a < b$.
- Algoritmo recursivo de búsqueda lineal.
- Algoritmo recursivo para los números fibonacci.

Definición recursiva de conjuntos

Una definición recursiva de un conjunto S se compone de:

Cláusula básica que especifica un conjunto de elementos primitivos.

Cláusula recursiva que especifica cómo los elementos del conjunto se pueden construir desde elementos ya conocidos en el conjunto S; pudiendo ser subclausulas recursivas.

Cláusula exclusión implícita cualquier cosa que no se encuentra en el conjunto como resultado de la clausula base o la clausula recursiva.

Ejemplo 4: Definición rec. conjunto de enteros

Básica $7,\ 10\ \in\ S$

Recursiva $Si\ r \in S\ entonces\ r+7,\ r+10 \in S$

Problema
Suponga, $(\forall n \geq 54)[n \in S]$.
Base: $54 = 2\ .\ 7 + 4\ .\ 10$

Hipótesis de inducción:
Asuma $\mathbf{n = r\ .\ 7 + s\ .\ 10}$ con $n \geq 54$

Paso inducción: Dos casos.
Caso 1: $r \geq 7$.
Entonces n + 1 = (r - 7) . 7 + (s + 5) . 10

Caso 2: $r < 7\ \Rightarrow r \cdot 7 \leq 42\ \Rightarrow s \geq 2$
Entonces n + 1 = (r + 3) . 7 + (s - 2) . 10

Ejemplo 5: Backus Normal Form (BNF)

BNF es un ejemplo de la gramática de libre contexto útil para definiciones recursivas de conjuntos.

BNF para cadenas

$$\langle cadena \rangle := \lambda \mid \langle cadena \rangle\, \langle caracter \rangle$$

BNF para identificadores

$$\begin{aligned}
\langle minuscula \rangle &:= a \mid b \mid \ldots \mid z \\
\langle mayuscula \rangle &:= A \mid B \mid \ldots \mid Z \\
\langle letra \rangle &:= \langle minuscula \rangle \mid \langle mayuscula \rangle \\
\langle digito \rangle &:= 1 \mid 2 \mid \ldots \mid 9 \\
\langle identificador \rangle &:= \langle letra \rangle \\
&:= \mid \langle identificador \rangle\, \langle letra \rangle \\
&:= \mid \langle identificador \rangle\, \langle digito \rangle
\end{aligned}$$

BNF para expresiones aritméticas

$e_1 + e_2,\ e_1 - e_2,\ e_1 * e_2,\ e_1/e_2,\ e_1\ exp\ e_2,\ (e_1)$

Bibliografía

1. Villalobos Jorge. Diseño y manejo de estructura de datos en C. McGrawHill, 1.996. Pags.14–39

2. Rosen Kenneth. Matemática Discreta y sus aplicaciones. 5ta. Edición. McGrawHill, 2.004.