Solución de recurrencias

Guillermo Palma

Universidad Simón Bolívar
Departamento de Computación y T.I.

CI-2612: Algoritmos y Estructuras de Datos II

Plan

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
$$f(n) = \begin{cases} n & \text{si } n = 0 \vee n = 1 \\ f(n-1) + f(n-2) & \text{de lo contrario} \end{cases}$$

Sobre las recurrencias

- Las recurrencias pueden surgir cuando se quiere caracterizar el tiempo de ejecución de algoritmos con llamadas recursivas
- Se quiere saber cual es el tiempo de ejecución de un algoritmo con llamadas recursivas
- Para determinar el tiempo de un algoritmo con llamadas recursivas, es necesario resolver la recurrencia
- Se quiere encontrar la expresión que es solución de la recurrencia
- Se quiere encontrar la cota superior de la expresión que es solución de la recurrencia

Ejemplos de recurrencias

- $T(n) = T(n/2) + c$. La solución es $\Theta(\log n)$
- $T(n) = 2T(n/2) + n$. La solución es $\Theta(n \log n)$
- $T(n) = T(n/2) + n$. La solución es $\Theta(n)$
- $T(n) = T(n-1) + n$. La solución es $\Theta(n^2)$
- $T(n) = 2T(n/2) + c$. La solución es $\Theta(n)$

Pasos para resolver una recurrencia usando el método de sustitución

1. Se propone una solución
   - La solución propuesta es del tipo $T(n) = O(f(n))$
   - Hipótesis inductiva: $T(k) \leq c\, f(k)$, para todo $k < n$
   - Se quiere probar: $T(n) \leq c\, f(n)$, para algún $c > 0$ y $n \geq n_0$
2. Se prueba la solución propuesta por inducción. La idea es usar la hipótesis inductiva para encontrar valores para las constantes c y n_o, para los cuales se cumple la tesis a probar.

Ejemplo 1 del Método de sustitución

Recurrencia: $T(n) = T(n-1) + n$

Solución

- *Se propone*: $T(n) = O(n^2)$
- *Se quiere probar*: $T(n) \leq c\, n^2$, para algún c y $n \geq n_0$
- *Hipótesis inductiva* : $T(n-1) \leq c\,(n-1)^2$ para todo $k < n$

Prueba por inducción

$$T(n) = T(n-1) + n \leq c\,(n-1)^2 + n$$
$$= cn^2 - 2cn + c + n = cn^2 - (2cn - c - n) \leq cn^2$$

Se tiene que esto se cumple si $2cn - c - n \geq 0$

lo que implica que $c \geq \frac{1}{2-\frac{1}{n}}$

Para $n \geq 1$ se tiene que $2 - \frac{1}{n} \geq 1$ entonces para $c \geq 1$ es válida

Ejemplo 2 del Método de sustitución

Recurrencia: $T(n) = 2T(n/2) + n$

Solución

- *Se propone*: $T(n) = O(n \log n)$
- *Se quiere probar*: $T(n) \leq c\, n \log(n)$, para algún c y $n \geq n_0$
- *Hipótesis inductiva* : $T(n/2) \leq c\,(n/2)\log(n/2)$

Prueba por inducción

$$T(n) = 2T(n/2) + n \leq 2c(n/2)\log(n/2) + n$$
$$= cn \log n - cn + n \leq cn \log n$$

Se tiene que esto se cumple si $-cn + n \leq 0$

lo que implica que para $c \geq 1$ es válida

Pasos para resolver una recurrencia usando el método del árbol de recursión

- Cada nodo representa el costo de la función en los diferentes niveles de recursión
- Se suma el costo de todos los niveles del árbol para obtener el costo de la recurrencia

Método del árbol de recursión

Ejemplo del método del árbol de recursión

Recurrencia: $W(n) = 2\ W(n/2) + n^2$

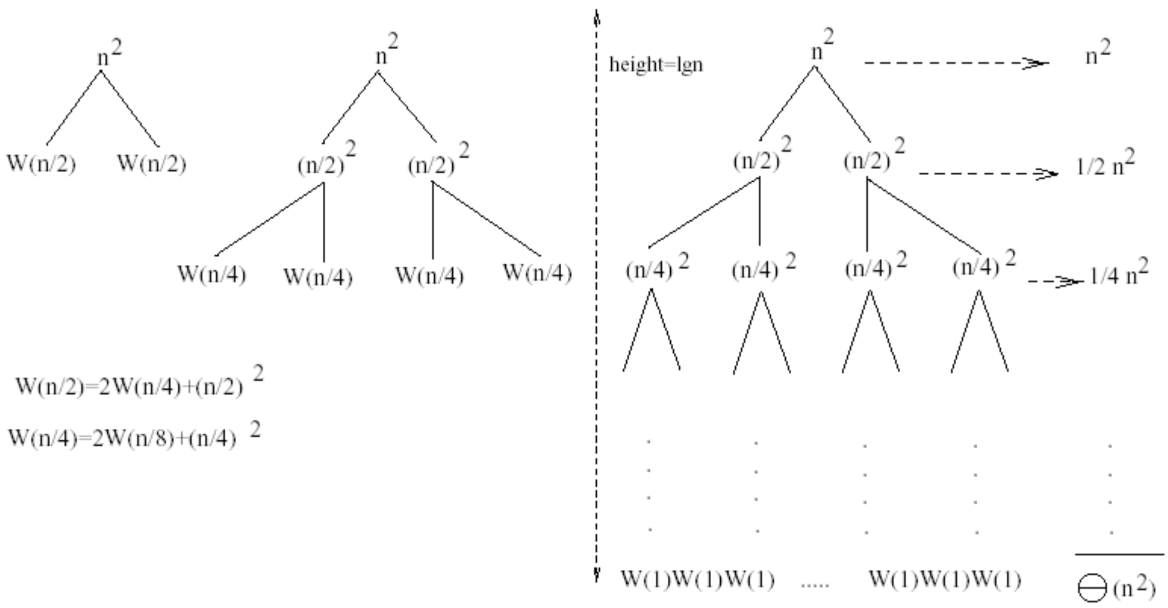

Figura: Árbol de recursión generado por las llamadas a la función. Fuente [1]

Ejemplo del método del árbol de recursión

Recurrencia: $W(n) = 2\ W(n/2) + n^2$

- El tamaño del problema de la entrada de la recurrencia a nivel i es $n/2^i$
- El tamaño del problema de la entrada de la recurrencia es 1 cuando $1 = n/2^i$, esto es cuando el nivel i es $i = \log n$
- El número de nodos al nivel i es $i = 2^i$
- El costo del problema al nivel i es $(n/2^i)^2$.
- $W(n) = \sum_{i=0}^{\log n-1} \frac{n^2}{2^i} + 2^{\log n} W(1) = n^2 \sum_{i=0}^{\log n-1} \frac{1}{2^i} + n \leq n^2 \sum_{i=0}^{\infty} \frac{1}{2^i} + O(n) = n^2(1 + \sum_{i=1}^{\infty} \frac{1}{2^i}) + O(n) = n^2(1+1) + O(n) = 2n^2 + O(n)$
- Por lo tanto, $W(n) = O(n^2)$

Método maestro

Ecuación de recurrencia del método maestro

$$T(n) = aT\left(\frac{n}{b}\right) + f(n)$$

donde $a \geq 1$, $b > 1$ y $f(n)$ positiva

- Caso 1: $f(n) = O(n^{\log_b a-\epsilon})$ para $\epsilon > 0$ entonces $T(n) = \Theta(n^{\log_b a})$
- Caso 2: $f(n) = \Theta(n^{\log_b a})$ entonces $T(n) = \Theta(n^{\log_b a} \log n)$
- Caso 3: $f(n) = \Omega(n^{\log_b a+\epsilon})$ para $\epsilon > 0$ y si $af(n/b) \leq cf(n)$ para algun $c < 1$, entonces $T(n) = \Theta(f(n))$

Ejemplo del método maestro

- Recurrencia $T(n) = 2T(n/2) + n$
- $a = 2$, $b = 2$ y $\log_2 2 = 1$
- Se compara $n^{\log_2 2}$ con $f(n) = n$
- $f(n) = \Theta(n)$, esto es caso 2
- $T(n) = \Theta(n \log n)$

Referencias

T. Cormen, C. Leirserson, R. Rivest, and C. Stein.
Introduction to Algorithms.
McGraw Hill, 3ra edition, 2009.