Programación 3 - 2021
16 de setiembre

1. Relaciones de recurrencia

Determine el orden de crecimiento de $T(n)$ en las siguientes relaciones de recurrencia. Asuma $T(1) = 1$ y que c es una constante positiva.

a. $T(n) = T(1) + T(n-1) + c\,n$

Solución

Este es el caso extremo en el que el problema se divide en dos subproblemas que no tienen el mismo tamaño. Vamos a compararlo con el conocido

$T(n) = T(n/2) + T(n/2) + c\,n$

del que sabemos que $T(n) = O(nlogn)$.

Primero usamos el método de .Adivinar"

Hacemos la conjetura $T(n) \leq dn^2$ e intentamos demostrar por inducción en n.

- Paso base, $n = 1$. Como $T(1) = 1$ se debe cumplir $1 \leq d$,
- Paso inductivo, $n \geq 2$. La hipótesis inductiva es $T(n-1) \leq d(n-1)^2$ y debemos probar $T(n) \leq dn^2$.

  Por la hipótesis inductiva $T(n) \leq 1 + d(n-1)^2 + c\,n = dn^2 + (c - 2d)n + d + 1$

  Se debe cumplir $(c - 2d)n + d + 1 \leq 0$

  Si $c - 2d < 0$, como $n \geq 2$ se cumple $(c-2d)n+d+1 < (c-2d)\cdot 2+d+1$ que, a su vez, es menor que 0 si $d > \frac{2c+1}{3}$.

  Entonces, eligiendo d mayor al máximo de $1, \frac{d}{2}, \frac{2c+1}{3}$ se cumple $T(n) \leq dn^2$.

Si no se hubiese cumplido $(c - 2d)n + d + 1 \leq 0$ aún podríamos intentar una conjetura menos exigente: $T(n) <= dn^2 + en$.

Demostrar por inducción

- Paso base, $n = 1$. Se debe cumplir $T(1) \leq d + e$ lo cual se cupmle si elegimos $d + e \geq 1$.
- Paso inductivo

Hipótesis inductiva: $T(n-1) \leq d(n-1)^2+e(n-1)$. Entonces se debe cumplir

$T(n) \leq 1 + d(n-1)^2 + e(n-1) + cn = dn^2 + en + (c-2d)n + 1 + d - e$

por lo que hay que probar que se puede elegir d, e para que se cumpla $(c-2d)n+1+d-e \leq 0$ lo que se cumple por ejemplo con $d = c/2$ y $e = d+1$.

Ahora usamos el método "Sustitución"

T(n) = T(1) + T(n-1) + cn

= 1 + 1 + T(n-2) + c(n-1) + cn

= 1 + 1 + 1 + T(n-3) + c(n-2) + c(n-1) +cn

. . . . . .

= T(n-i) + $\sum_{j=0}^{i-1}$ c(n-j) + i

. . . . . .

[se llega al caso base cuando i = n-1]

= T(1) + $\sum_{j=0}^{n-2}$ c(n-j) + n - 1

= c n $\sum_{j=0}^{n-2}$ 1 - c $\sum_{j=0}^{n-2}$ j + n

= cn(n-1) - c (n-2)(n-1) /2 + n

<= dn^2 + en

si, por ejemplo $d > \frac{c}{2}$ y $e > \frac{c+2}{2}$.

b. $T(n) = 8T(n/2) + c\,n^2$

Solución

Podemos conjeturar $T(n) = O(n^3)$ por analogía a que en
$T(n) = 4T(n/2) + c\,n$ se cumple $T(n) = O(n^2)$.

Vamos a usar el método "Sustitución"

T(n) = 8 T(n/2) + cn^2

= 8 * 8 * T(n/4) + 2 c n^2 + cn^2

= 8 * 8 * 8 T(n/8) + 4 c n^2 + 2 c n^2 + cn^2

. . . . . .

= 8^i T(n/ 2^i) + cn^2 $\sum_{j=0}^{i-1} 2^j$

. . . . . .

[Se llega al caso base cuando n = 2^i =>i = log n]

= $8^{\log\ n}$ + cn^2 $\sum_{j=0}^{\log n\ -\ 1}$ 2^j

= $8^{\log\ n}$ + cn^2 ($2^{\log n}$-1)

= $n^{\log\ 8}$ + cn^3 - cn^2

= n^3 + cn^3 - cn^2

[$a^{\log_b\ n} = n^{\log_b\ a}$]

Esta relación de recurrencia es la que corresponde a la multiplicación de matrices diviendolas en bloques. Con el método de Strassen se resuelve con solo 7 multiplicaciones en lugar de 8, y se cumple $T(n) = O(n^{\log 7})$

Hay que tener en cuenta que el tamaño de la entrada es $m = 2n^2$. Por lo tanto un tiempo de ejecución $O(n^3)$ es $O(m^{3/2})$.

c. $T(n) = a\,T(n/b) + c\,n^E$, donde $E = \log_b a$, $b \geq 2, a \geq 1$

Solución

En base al Master Theorem se puede saber que $T(n) = \Theta(n^E \log n)$.

El Master Theorem resulve la relación de recurrencia $T(n) = aT(n/b) + f(n)$ si se cumplen determinadas condiciones.

El factor n^E es la cantidad de casos base, que, como su costo es O(1), es también el costo de procesar todos los casos base. De su relación con $f(n)$ se determina cuál es el factor que determina el costo.

1. Si f(n) = $\Theta(\mathrm{n^E})$ entonces $T(n) = \Theta(n^E \log n)$
2. Si f(n) = $\mathrm{O(n^{E-\epsilon})}$ para algún $\epsilon > 0$ entonces $T(n) = \Theta(n^E)$
3. Si f(n) = $\Omega(\mathrm{n^{E+\epsilon}})$ para algún $\epsilon > 0$, y se cumplen ciertas condiciones de regularidad entonces $T(n) = \Theta(f(n))$

2. Mediana de la unión de secuencias ordenadas

Sean A y B dos secuencias de largo n de enteros. Asuma que las secuencias están ordenadas, n es potencia de 2, no hay elementos repetidos y son dadas en arreglos. Diseñe un algoritmo que permite encontrar la *mediana* (el valor que si se ordenara toda la secuencia qudaría en la mitad) de $A \cup B$. El algortimo debe admitir una implementación con tiempo de ejecución $O(\log n)$.

Solución

El tiempo requerido corresponde a la relación de recurrencia $T(n) = T(n/2) + 1$, lo que hace pensar en un algortimo que en cada paso, con costo O(1), logre reducir el tamaño a la mitad. En cada una de las secuencias la posición de su mediana es $m = \lfloor \frac{n+1}{2} \rfloor$.

En $A \cup B$, de tamaño $2n$, la mediana estaría en la posición n, por lo que habría exactamente $n-1$ elementos menores y exactamente n elementos mayores.

Consideremos cada secuencia A y B separada en dos segmentos iguales:

$\mathrm{a_1 \ldots a_m \mid a_{m+1} \ldots a_n}$

$\mathrm{b_1 \ldots b_m \mid b_{m+1} \ldots b_n}$

Comparamos $\mathrm{a_m}$ y $\mathrm{b_m}$ y suponemos sin pérdida de generalidad $a_m < b_m$ (el otro caso es similar)

Entonces $\mathrm{a_m}$ es menor que al menos n + 1 elementos, $\mathrm{a_{m+1} \ldots a_n}$ y $\mathrm{b_m}$, $\mathrm{b_{m+1} \ldots b_n}$, por lo que no puede ser la mediana. Lo mismo vale para los $\mathrm{a_i}$, $i < m$. Por lo tanto el segmento $\mathrm{a_1 \ldots a_m}$ no contiene la mediana

Con un argumento similar b_{m+1} y los b_i, $i \geq m+1$ es mayor que al menos n elementos, $a_1 \ldots a_m$ y $b_1 \ldots b_m$. Por lo tanfo el segmento $b_{m+1} \ldots b_n$ no contiene la mediana.

Por lo tanto el problema pasa a dos secuencias ordenadas, ambas de largo $n/2$, que también es potencia de 2, por lo que se siguen cumpliendo las condiciones del problema original. En este problema hay n elementos por lo que su mediana está en la posición $n/2$. Como se habían descartado exactamente $n/2$ la mediana de este conjunto estaría en la posición n del conjunto original, por lo que es su mediana.

El problema se puede resolver en O(1) en cada paso, sin necesidad de copiar segmentos, si se generaliza el problema y se incluyen los límites del arreglo como parte de los datos de entrada.