

Marco Ramírez 19588  
 José Hurtarte 19707  
 Christian Pérez 19707  
 Andrei Portales 19825

### Ejercicios

1. Use el método de sustitución para determinar la solución a la siguiente recurrencia:  $T(n) = 4T\left(\frac{n}{2}\right) + n$ . La solución de acuerdo con el *Master Method* es  $\Theta(n^2)$ , pero usar la hipótesis  $cn^2$  falla. Realice el procedimiento bajo esa hipótesis para comprobar que falla y luego modifique la hipótesis para que funcione.

**Nota: para demostrar la cota inferior la hipótesis  $cn^2$  no falla**

$$1) \quad T(n) = 4T\left(\frac{n}{2}\right) + n$$

y por supuesto  $T(n) = \Theta(n^2)$

- usamos hipótesis:

$$\begin{aligned} T(n) &\leq Cn^2 - dn \\ \text{-por inducción} \quad T\left(\frac{n}{2}\right) &\leq C\left(\frac{n}{2}\right)^2 - d\frac{n}{2} \\ T(n) &\leq 4\left(C\left(\frac{n}{2}\right)^2 - d\frac{n}{2}\right) + n = Cn^2 - 2dn + n \end{aligned}$$

$$= Cn^2 - dn + \underbrace{n - dn}_{\text{cuando } d \geq 1}$$

$$\leq Cn^2 - dn$$

dmostrando

$$T(n) = O(n^2) \quad //$$

$$O \subset T(1) \leq Cn^2 - 1d$$

$$O \subset T(1) \leq C - d$$

$$C > d$$

- para poder probar  $T(n) = \Theta(n^2)$   
nos hace falta probar  $T(n) = \Omega(n^2)$

» Para esto tomamos la hipótesis:

$$T(n) \geq Cn^2$$

$$T\left(\frac{n}{2}\right) \geq C\left(\frac{n}{2}\right)^2 = \frac{Cn^2}{4}$$

$$T(n) \geq \cancel{A} \frac{Cn^2}{\cancel{A}} + n = Cn^2 + n \geq \boxed{Cn^2}$$

$$0 < C(n^2) \leq T(1)$$

$$0 < C \leq T(1)$$

$$\cancel{\forall C > 0} //$$

Demostrando así  $T(n) = \Omega(n^2)$

∴  $\boxed{T(n) = \Theta(n^2)}$



2. Resuelva la recurrencia  $T(n) = 3T(\sqrt{n}) + \log_2 n$ . Para hacerlo demuestre primero que se puede convertir en  $S(m) = 3S\left(\frac{m}{2}\right) + m$ ; y luego resuelva esta recurrencia con el método de sustitución. Con este resultado provea la respuesta para la recurrencia original. **Hint:** note que, en  $S(m)$ ,  $m$  parece ocupar el lugar que  $\log_2 n$  tiene en  $T(n)$ .

$$T(n) = 3T(\sqrt{n}) + \log_2 n$$

- Definimos  $S(n) = T(2^n)$

- Entonces  $S(n) = 3T(2^{n/2}) + n = 3S\left(\frac{n}{2}\right) + n$

- Cambiando  $n$  por  $m$ :

$$S(m) = 3S\left(\frac{m}{2}\right) + m$$

$$n = 2^m$$

Hipótesis Prueba Master Method

$$a = 3$$

$$b = 2$$

$$m = O(m^{\log_2 3 - \epsilon})$$

$$S(m) = \Theta(m^{\log_3})$$

Queremos Probar hipótesis  $S(m) = \Theta(m^{\log_2 3})$

$$S(m) \leq cm^{\log_2 3} - dm$$

$$S\left(\frac{m}{2}\right) = 3S\left(\frac{m}{2}\right) + m \leq 3\left(c\left(\frac{m}{2}\right)^{\log_2 3} - d\frac{m}{2}\right) + m$$

$$= 3\left(c\frac{m^{\log_2 3}}{2}\right) - \frac{3}{2}dm + m$$

Necesitamos un  $C > 0$

$$\begin{aligned} & -\frac{3}{2}dm + m \leq 0 \quad /m \\ & -\frac{3}{2}d + 1 \leq 0 \rightarrow d \geq \frac{2}{3} \end{aligned}$$

entonces

$$S(m) \leq cm^{\log_2 3}$$

Sabemos que  
 $S(1) > 0$

entonces  $0 < S(1) \leq ((1))^{\log_2 3} - d$

$$0 < S(1) \leq c - d$$

↳ por transitividad

$c > d$  y  $c > \frac{2}{3}$

demostmando  $S(m) = O(m^{\log_2 3})$

- faremos demostrar  $S(m) = \Theta(m^{\log_2 3})$  tambien  
 tenemos que demostrar  $\Delta(m^{\log_2 3})$

con hipótesis:  $Cm^{\log_2(3)} - dm - \lambda \leq S(m)$

$$= 3C \frac{m^{\log_2 3}}{2} + \frac{3md}{2} - \lambda + m \leq S(m)$$

$$= Cm^{\log_2 3} + \frac{3md}{2} - \lambda + m \leq S(m)$$

$$-\frac{3}{2}d + m \leq 0$$

$$d \geq \frac{3}{2} + 1$$

para que

mantenga la forma

$$= Cm^{\log_2 3} - dm - \lambda \leq$$

$$0 < C - d \leq S(m)$$

$$C > -(d + \lambda)$$

Comprobando así,

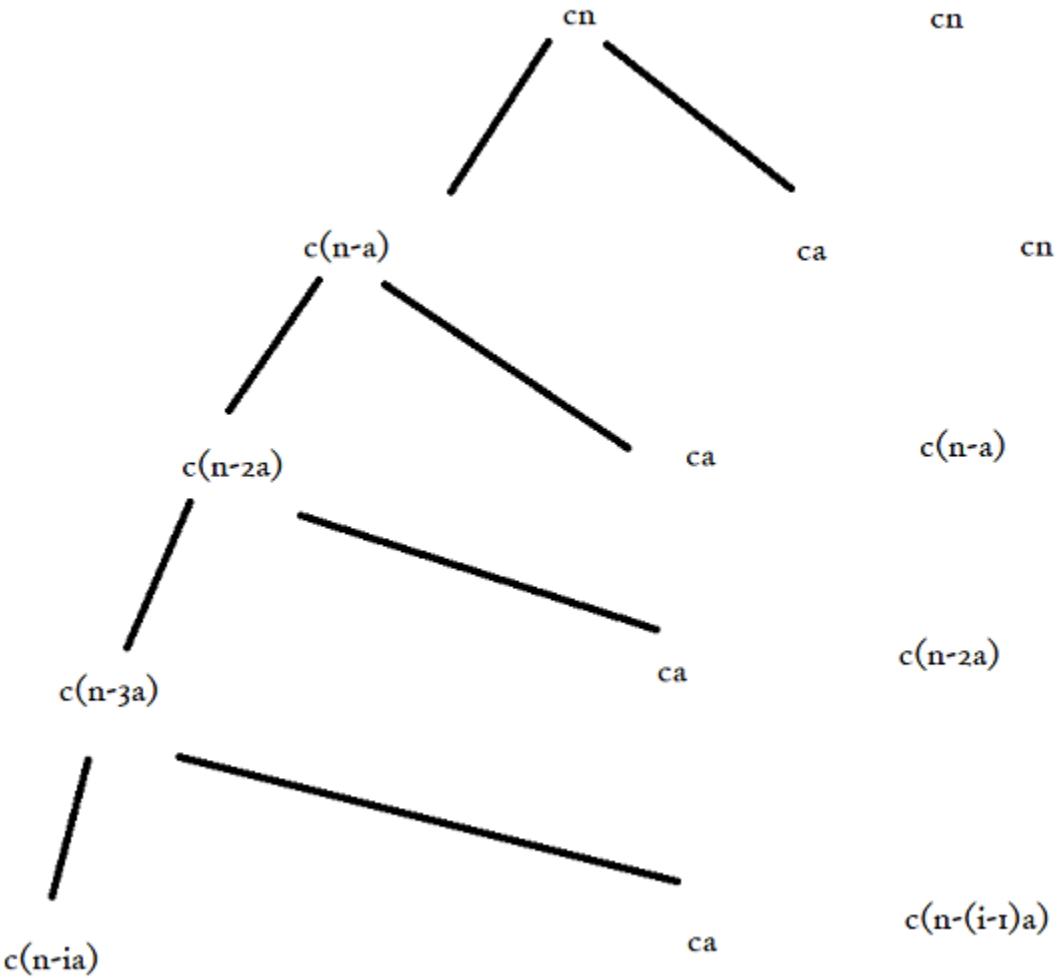
$$S(m) = \Delta(m^{\log_2 3})$$

□

3. Use un árbol de recursión para proveer una cota ajustada a la recurrencia  $T(n - a) + T(a) + cn$ , donde  $a \geq 1$ ,  $c > 0$ ; ambas constantes. Puede suponer que  $n$  es múltiplo de  $a$ .

Sabemos que la altura del árbol es  $\frac{n}{a}$  y cada nivel es  $c(n - ia)$  entonces

$$T(n) = \sum_{i=0}^{\frac{n}{a}} c(n - ia) + (n/a)ca = \sum_{i=0}^{\frac{n}{a}} cn - \sum_{i=0}^{\frac{n}{a}} cia + (n/a)ca = cn^2/a - \Theta(n) + \Theta(n) = \Theta(n^2)$$



4. Use el *Master Method* (si es posible) para dar cotas ajustadas a las siguientes recurrencias:

a.  $T(n) = 2T\left(\frac{n}{4}\right) + \sqrt{n}$

- i  $S(m) = T(4^m p)$  para  $p > 0$
- ii  $S(m) = 2S(m-1) + \sqrt{p2^m}$
- iii  $2^{-m}S(m) - 2^{-(m-1)}S(m-1) = \sqrt{p}$
- iv  $\sum_{m=1}^k (2^{-m}S(m) - 2^{-(m-1)}S(m-1)) = (2^{-m}S(m) - 2^{-(m-1)}S(m-1)) + (2^{-(m-1)}S(m-1) - 2^{-(m-2)}S(m-2) + \dots + (2^{-2}S(2) - 2^{-1}S(1)) + (2^{-1}S(1) - 2^0S(0))$
- v  $= 2^{-m}S(m) - S(0)$
- vi  $= S(m) = 2^m(m\sqrt{p} + S(0))$

ix

ya que

$$m = \log_4\left(\frac{n}{p}\right) = T(n) = \sqrt{\frac{n}{p}}(\sqrt{p} \log_4\left(\frac{n}{p}\right) + S(0)) = \sqrt{n} \log_4(n) + d\sqrt{n}$$

x

Donde  $d$  es una constante que determina la condición inicial.

i

- b.  $T(n) = 4T\left(\frac{n}{2}\right) + n^2 \log_2 n$
- No se puede realizar mediante master method, porque no tenemos una constante al cual aplicar.

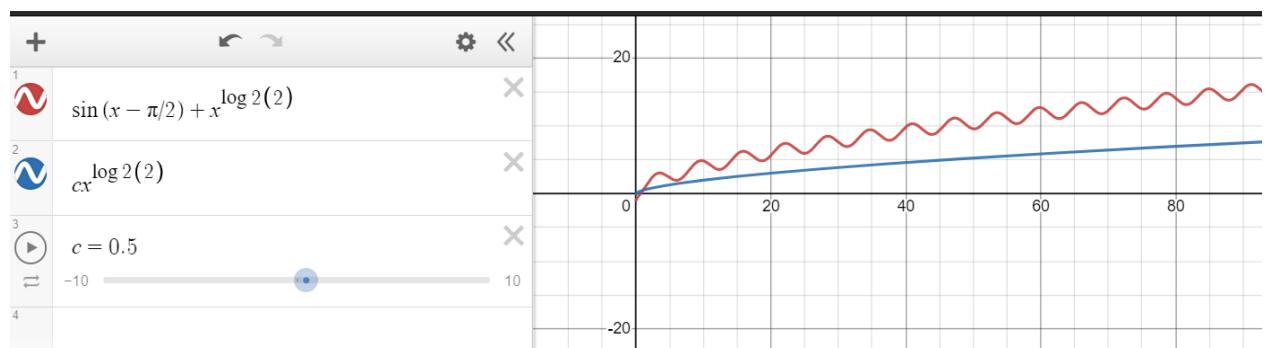
5. Dé una recurrencia que cumpla con las condiciones del tercer caso del *Master Method* excepto la condición de regularidad.

Una recurrencia que cumple con la condición de  $f(n) = \Omega(n^{\log_b a + \epsilon})$  con  $T(n) = aT\left(\frac{n}{b}\right) + f(n)$

es:

$$T(n) = T\left(\frac{n}{2}\right) + \sin(n - \pi)n^{\log_2 2}$$

ya que:  $\sin(n - \pi)n^{\log_2 2}$  si esta está acotada por inferiormente por  $\log_2(1 + \epsilon)$  con  $\epsilon = 1$ , a partir de  $n_0 = 2$  y  $c = 0.5$



Debido a que  $f(n)$  es oscilatoria no se puede garantizar la condición de regularidad, ya que no se puede garantizar que el nodo hijo del árbol de recursión será menor al nodo padre. Ya que no siempre, como se observa en la gráfica  $f(n)/2$  es menor a  $f(n)$ .

6. Sea  $G = (V, E)$  un grafo dirigido. Deseamos determinar si existe un camino que conecte a dos nodos,  $u, v \in V$ ; esto se conoce como el **problema de conectividad-st** o **STCON**. El algoritmo de Savitch, presentado a continuación, determina si existe un camino con tamaño máximo  $2^i$  entre dos nodos  $u, v$  del grafo  $G$ :

---

**Algorithm 3.4** Savitch

---

```
1: if  $i = 0$  then
2:   if  $u = v$  then
3:     return T
4:   else if  $(u, v)$  is an edge then
5:     return T
6:   end if
7: else
8:   for every vertex  $w$  do
9:     if  $R(G, u, w, i - 1)$  and  $R(G, w, v, i - 1)$  then
10:      return T
11:    end if
12:  end for
13: end if
14: return F
```

---

Identifique las partes *Divide*, *Conquer* y *Combine* de este algoritmo, y determine (con notación asintótica) una cota superior para su tiempo de ejecución si se ejecuta para  $i = \log_2 n$ , donde  $n$  es el número de vértices en el grafo. El tiempo de ejecución que encuentre,

¿será indicador de eficiencia (es decir, será que el algoritmo es “rápido”) o de ineficiencia (“lento”)?

- $R(G, u, v, i)$  es Verdadero si: hay un camino en  $G$  de  $u$  a  $v$  con longitud a lo sumo  $2^i$
- Si existe un camino de  $u$  a  $v$ , ese camino tiene un punto medio  $w$
- a lo más  $v_i$  a haber caminos de distancia  $2^{i-1}$  de  $u$  a  $w$  y de  $w$  a  $v$

- Tambien Sabemos que:

$$R(G, u, v, i) \Leftrightarrow (\exists w)[R(G, u, w, i-1) \wedge R(G, w, v, i-1)]$$

Partes divide and conquer

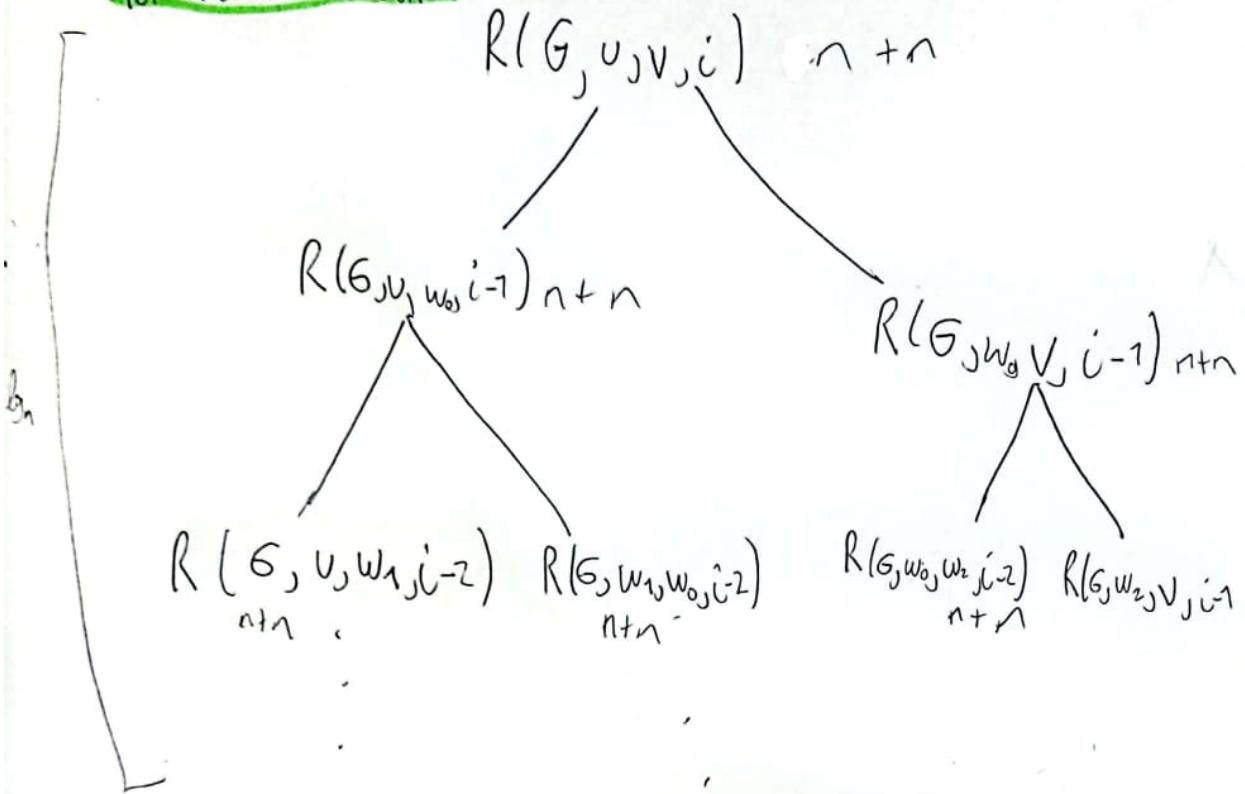
- Divide; Al momento de buscar un Punto Medio w con

la instrucción q: ;FR( $G, u, w, i-1$ ) and R( $G, w, v, i-1$ )

- Conquer; Cuando Se encuentra si  $u=v$  o si  $w,v$  es una  
arista en las instrucciones 1-7

- Combine; Cuando al retornar las Recursiões se encuentra  
un camino de  $u$  a  $w$  y de  $w$  a  $v$

Por árbol de recursión



- Podemos ver que este es un hiper-árbol de recursión en donde cada rama desemboca n ramas 2 veces,
- en el peor de los casos, sabiendo que la distancia máxima es  $2^i$ , todos los nodos desembocan al final de la recursión dandoles un tamaño de  $\lg(n)$   
 $\therefore T(n) = n^{\log_2 n} = O(n^{\log_2 n})$

que por transformaciones:  $n^{\log_2 n} = 2^{\log_2(n^{\log_2 n})} = 2^{\log_2(\log_2 n)}$

$\therefore T(n) = 2^{O(\log_2^2 n)}$

Siendo Super-Polinomial y poco eficiente en términos de tiempo