Ejercicio 1 del examen de Teoría de Algoritmos del 9 de febrero del 2005.

Este ejercicio tiene una mala leche de cuidado.

No sé si se puede hacer de una forma más sencilla, le he dado veinte vueltas y he obtenido un resultado razonable únicamente de esta manera...

El resultado que he obtenido, por cierto, coincide si se sigue el desarrollo que el libro realiza (el libro no calcula los coeficientes, con lo cual no está bien del todo, porque si los coeficientes valieran 0 se anularían componentes de la ecuación temporal). Incluyo este desarrollo alternativo al final.

En primer lugar, calculamos la expresión del tiempo T(n) (la facilita):

```
       { 1              si n=1
T(n) = {
       { 7T(n/2) + n^2  si n>1, potencia de 2
```

Cambio n=2^i

```
     { 1                si 2^i=1, es decir, si i=0
ti = {
     { 7ti-1 + 4^i si 2^i>1, es decir, si i>0
```

Obtengo la expresión ti con los coeficientes c1 y c2:

```
ti - 7ti-1 = 4^i
(x-7)(x-4) = 0
ti = c1 7^i + c2 4^i
```

Calculo las condiciones iniciales:

```
t0 = 1
t1 = 7+4 = 11
```

Utilizo las condiciones iniciales para obtener los coeficientes:

```
t0 = c1 + c2 = 1  } c1 = 1-c2        }                 }               } c1 = 7/3
t1 = 7c1+4c2 = 11 } 7(1-c2)+4c2 = 11 } 7-7c2+4c2 = 11 } c2 = (11-7)/-3 = -4/3 }
```

Por tanto:

```
ti = 7/3 * 7^i -4/3 * 4^i
```

Deshago el cambio de variable:

```
i = log2n
```

Obtenemos como resultado:

T(n) = 7/3 * 7^(log2n) -4/3 4^(log2n)

Este crece en la forma O(7^log2n)
Que por cierto, por propiedad del logaritmo equivale a O(n^log2(7))

Ahora calculamos la expresión del tiempo T'(n) (la difícil):

Cambio n=4^i (porque con 2^i acaban saliendo raices cuadradas y tres coeficientes, que juntos son muy difíciles de quitar)

```
      { 1                si 4^i=1, es decir, si i=0
t'i = {
      { ati-1 + 4^i      si 4^i>1, es decir, si i>0
```

Obtengo la expresión ti con los coeficientes c1 y c2:

```
t'i - at'i-1 = 4^i
(x-a)(x-4) = 0
t'i = c1 a^i + c2 4^i
```

Calculo las condiciones iniciales:

```
t'0 = 1
t'1 = a+4
```

Utilizo las condiciones iniciales para obtener los coeficientes:

```
t'0=c1+c2=1     } c1=1-c2         }              }                  }} c1 = -a/(4-a)
t'1=ac1+4c2=a+4 } a(1-c2)+4c2=a+4 } a-ac2+4c2=a+4 } c2=(4+a-a)/(4-a) } c2=4/(4-a)}
```

Por tanto:

```
t'i = -a/(4-a) a^i + 4/(4-a) a^i
```

Deshago el cambio de variable (ojo, no sale logaritmo en base 2 sino logaritmo en base 4):

```
i = log4n
T'(n) = -a/(4-a) a^(log4n) + 4/(4-a) 4^(log4n)
```

Y estudiamos los casos:
- a no puede valer negativo porque iría alternando el signo del tiempo cuando log2n sea par o impar.
- Si a vale 0 tenemos:
T'(n) = -0/(4-0) 0^(log4n) + 4/(4-0) 4^(log4n)
O sea, O(4^log4n)
- Si a vale 1 tenemos:
T'(n) = -1/3 1^log4n + 4/3 4^log4n
O sea, O(4^log4n)
- Así sigue igual hasta 4.
- Si a vale 4 tenemos:
T'(n) = -4/(4-4) a^(log4n) + 4/(4-4) 4^(log4n)
O sea, indeterminación (ese a no tiene sentido)
- Si a vale 5 tenemos:
T'(n) = -4/(4-5) 5^(log4n) + 5/(4-5) 4^(log4n)
O sea, O(5^log4n)

De ahí en adelante, O(a^log4n)
Que por cierto, equivale a O(n^log4a)

De donde se puede ver el orden O(n^log4a), y de donde sale la respuesta directa:

Si el algoritmo A tarda un orden O(n^log2(7)) y el algoritmo A' tarda un orden O(n^log4a), A' tardará asintóticamente menos mientras que:

log4a < log2(7)

Despejamos:

a < 4^log2(7)
<u>a = 49</u>

Aquí incluyo la forma de resolver la segunda parte del problema, según el libro (el resultado es el mismo):

Para T'(n), sigo el razonamiento de la solución al problema 1.4 de la relación.

$$T'(n) = \begin{cases} 1 & \text{si } n=1 \\ T(n/4) + n^2 & \text{si } n>1\text{, potencia de 2} \end{cases}$$

En primer lugar hacemos el cambio $n = 4^i$, $i = \log_4 n$, obtenemos

$T(4^i) = aT(4^{(i-1)})+4^{2i}$

Luego llamamos $t_i=T(4^i)$, y la ecuación queda como:

$t_i - at_{i-1} = (4^2)^i$

Su polinomio característico es:

$(x-a)(x-4^2) = 0$

Ahora vamos a estudiar los distintos casos:

- Supongamos que $a = 4^2$. En cuyo caso el polinomio queda: $(x-4^2)^2$

Y por tanto:

$t_i = c_1 \cdot 4^{(2i)}+c_2 \cdot i \cdot 4^{(2i)}$

Sacamos factor común:

$t_i = (c_1+c_2 i) \cdot 4^{(2i)}$

Deshacemos los cambios:

$T(4^i) = (c_1+c_2 i) \cdot 4^{(2i)}$

$T(n) = (c_1+c_2\log_4 n) \cdot n^2$ que es $O(n^2\log n)$ (en el libro no calcula el segundo coeficiente, pero se puede demostrar fácilmente que no puede valer 0 a partir de las condiciones iniciales)

- Supongamos ahora que $a \neq 4^2$. En cuyo caso el polinomio tiene dos raices y queda:

$t_i = c_1 \cdot a^i + c_2 \cdot 4^{(2i)}$

Deshacemos:

$T(2^i) = c_1 \cdot a^i + c_2 \cdot 4^{(2i)}$

$T(n) = c_1 a^{\log_4 n} + c_2 n^2 = c_1 n^{\log_4 a} + c_2 n^2$

De donde se puede ver el orden $O(n^{\log_4 a})$ (ojo, faltaría calcular los coeficientes, no debería hacerse así), y de donde sale la respuesta directa:

Si el algoritmo A tarda un orden $O(n^{\log_2(7)})$ y el algoritmo A' tarda un orden $O(n^{\log_4 a})$, A' tardará asintóticamente menos mientras que:

$\log_4 a < \log_2(7)$

Despejamos:

$a < 4^{\log_2(7)}$
$\underline{a = 49}$