Complexidade assintótica de Algoritmos Recursivos
Método da Árvore de Recursão

Exemplo 1

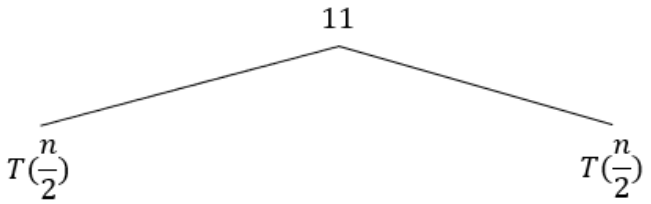
$$\begin{cases} T(n) = 2 & n = 1 \\ T(n) = 2T\left(\frac{n}{2}\right) + 11 & n > 1 \end{cases}$$

Inicialmente tem-se a primeira execução do método. Como o tamanho do problema ainda não atinge o critério de parada (n>1), é feita a chamada recursiva.

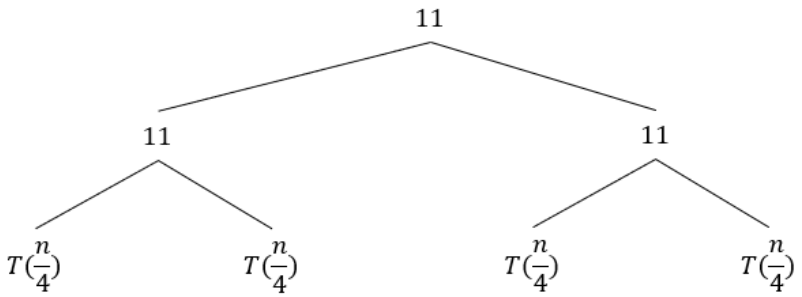
$$T(n)$$

Na primeira execução do problema serão chamadas duas novas instâncias do mesmo problema tendo a entrada, no entanto, metade do tamanho da iteração anterior $T\left(\frac{n}{2}\right)$. Além disso, cada vez que o problema é executado, são chamadas duas instâncias do problema. Como em cada etapa o número de problemas dobra, a representação visual da estrutura hierárquica será uma árvore binária. Caso fossem disparadas três novas instâncias do problema teríamos uma árvore

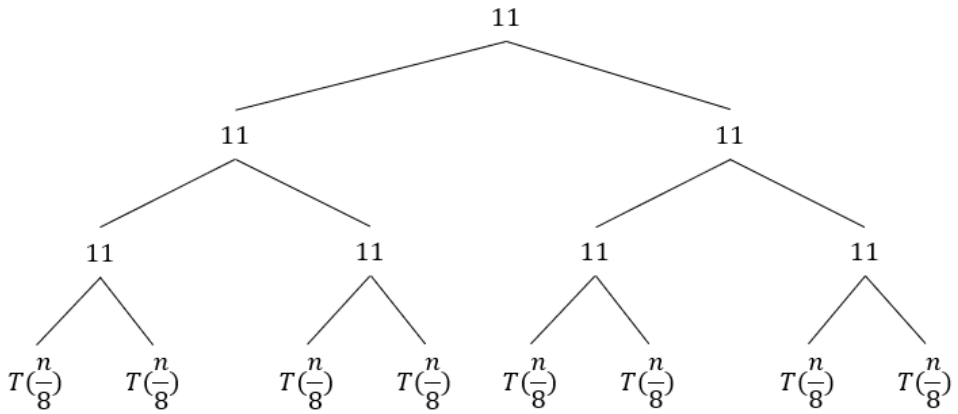
Porém, cada vez que uma chamada do problema é executada, o custo atrelado à usa execução deve ser pago. Neste caso, o custo de cada chamada é fixo em 11. Assim, a execução da primeira recursão terá o comportamento expresso a seguir.



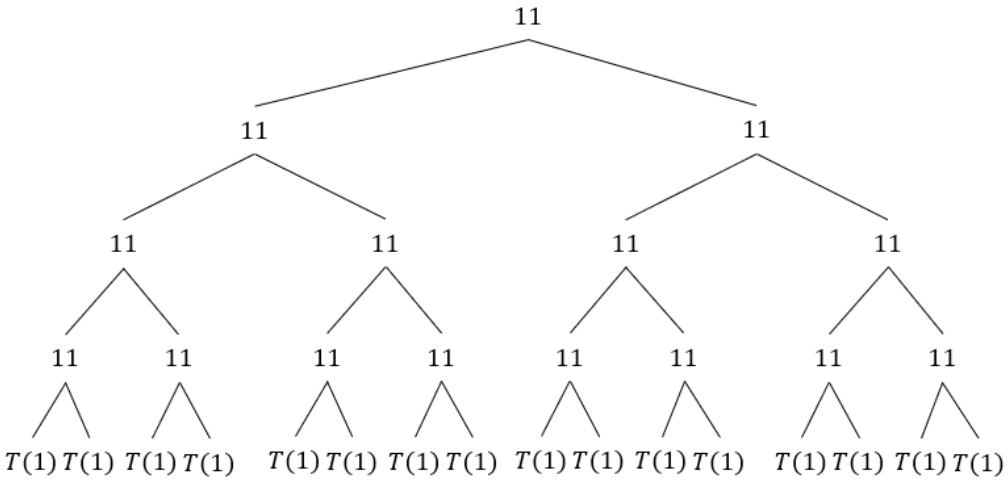
Em seguida, cada recursão do problema é executada, pagando seu custo próprio e disparando mais duas chamadas recursivas. O tamanho do problema é diminuído pela metade a cada nova iteração.



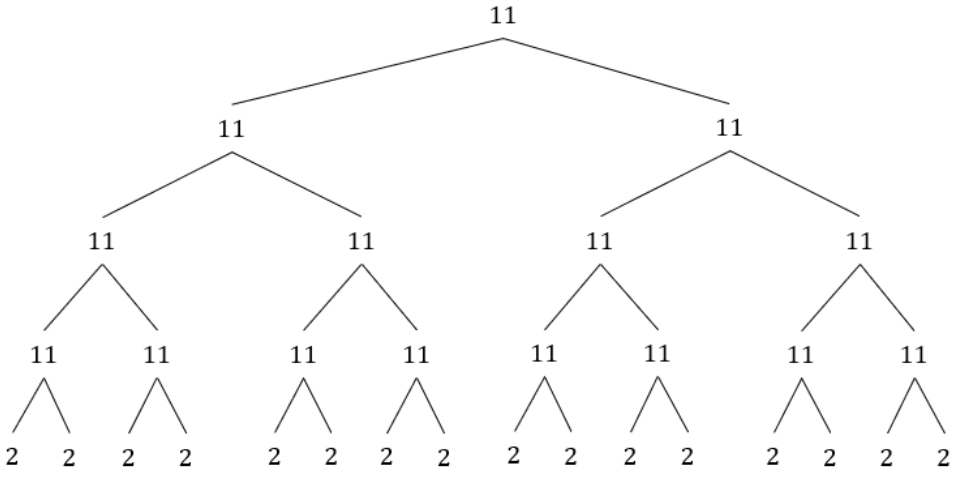
Assim, cada nova etapa da recursão tem o dobro de problemas da iteração anterior e os problemas terão metade do tamanho da recursão anterior.



Considere, para critério de ilustração, que o problema inicial tenha tamanho $n = 16$. Neste caso, o processo recursivo é repetido até que tenhamos alcançado o termo $T\left(\frac{16}{16}\right)$ que leva à chamada do problema $T(1)$.



Esse processo recursivo é realizado até que o tamanho do problema atinja o critério de parada que, neste caso, é um $n = 1$. Quando o critério de parada é alcançado a recursão deixa de ser chamada e se paga o custo para este caso específico. Segundo a equação de recorrência, o custo para este caso é 2. Assim, o comportamento da árvore pode ser expresso da seguinte forma:



Para obtermos então o custo assintótico de execução do método, precisamos somar todos os custos pagos. Assim, a estratégia é somar todos os elementos presentes na árvore. Neste caso teremos 15 elementos com custo 11 e outros 16 elementos com custo 2, totalizando um custo total de 197. Porém, queremos descobrir o custo do método em função do tamanho original n. Para tanto, é necessário descrever o comportamento dos elementos ao longo da árvore, tanto nós internos quanto externos.

O primeiro passo é entender quantos elementos estão dispostos em cada nível da estrutura. Para tanto, analisamos o comportamento do fator a da equação de recorrência. Neste caso, o valor de a é dois, indicando que cada chamada do problema dispara duas chamadas para novos problemas, os quais têm metade do tamanho da iteração atual ($2T\left(\frac{n}{2}\right)$). Assim, podemos perceber que cada nível possui 2x mais elementos que no nível anterior: primeiro nível contém um elemento, no segundo nível temos 2 elementos, no terceiro nível temos 4 elementos e assim sucessivamente. Ou seja, o crescimento do número de elementos dá-se na razão 2^i (2^0,2^1,2^2,...).

O processo recursivo é executado até que o problema possua tamanho 1, quando alcança o caso base, ou seja, até que $\frac{n}{2^i} = 1$. Assim pode-se descobrir qual é altura das folhas da árvore e consequentemente a sua altura.

$$\frac{n}{2^i} = 1$$

$$n = 2^i$$

Usando a propriedade logarítmica $a^x = b \rightarrow \quad log_a b = x$

$$i = log_2 n$$

Assim, a altura das folhas (e da árvore) será $log_2 n$. Ou seja, a altura interna (da raiz até o último nível antes das folhas) será $log_2 n - 1$.

O custo total de execução do algoritmo pode ser calculado pela seguinte fórmula:

$$T(n) = Custo\ (Interno) + Custo(Folhas)$$

O custo das folhas ocorre quando os casos bases são executados, ou seja, quanto o tamanho do problema é 1. Neste caso, o custo de cada execução é 2. O número de folhas é determinado pelo tamanho do problema. Sabe-se que o problema é divido em dois até que o tamanho seja de uma unidade, ou seja, são encontrados n elementos nas folhas da árvore. Assim o custo das folhas pode ser definido por:

$$Custo(Folhas) = Custo\ Base * Número\ de\ folhas$$

$$Custo(Folhas) = 2 * n$$

Para encontrarmos o custo interno precisamos somar o custo de todos os elementos que não sejam folhas, nível a nível. Sabemos que o número de elementos do nível depende do nível que está, ou seja, $2^{nivel-1}$. Sabe-se também que o custo de execução de cada elemento é 11. Além disso, é sabido que o número de níveis internos na árvore é dado por $log_2 n - 1$. Podemos então pensar no custo interno da seguinte forma:

$$Custo\ Interno = 2^0 * 11 + 2^1 * 11 + 2^2 * 11 + \cdots + 2^{log_2 n - 1} * 11)$$

$$Custo\ Interno = \sum_{i=0}^{log_2 n - 1} 2^i * 11$$

Usando a propriedade da soma dos termos de uma série exponencial

$$\sum_{k=0}^{m} x^k = \frac{x^{m+1} - 1}{x - 1}$$

Neste cenário consideramos que $x = 2$ e $m = log_2 n - 1$, então

$$Custo\ Interno = 11 * \sum_{i=0}^{log_2 n - 1} 2^i = 11 * \frac{2^{(log_2 n - 1)+1} - 1}{2 - 1} = 11 * (n - 1) = 11n - 11$$

Assim,

$$T(n) = Custo\ (Interno) + Custo(Folhas)$$

$$T(n) = 11n - 11 + 2n$$

$$T(n) = 13n - 11$$

Obtendo, dessa forma, o custo assintótico:

$$\boldsymbol{T(n) = O(n)}$$

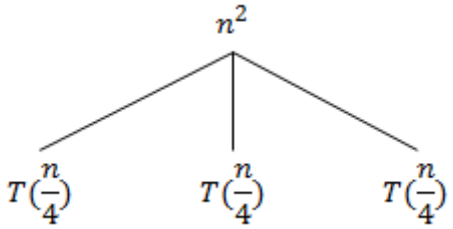
Exemplo 2

$$\begin{cases} T(n) = 1 & n = 1 \\ T(n) = 3T\left(\frac{n}{4}\right) + n^2 & n > 1 \end{cases}$$

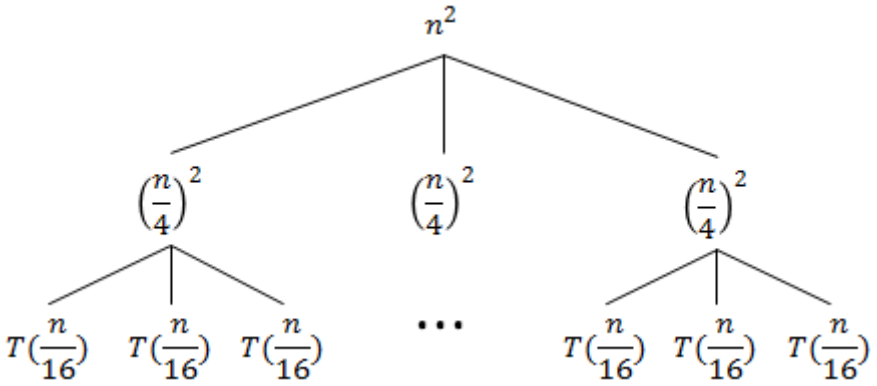
Inicialmente tem-se a primeira execução do método. Como o tamanho do problema ainda não atinge o critério de parada (n>1), é feita a chamada recursiva.

$$T(n)$$

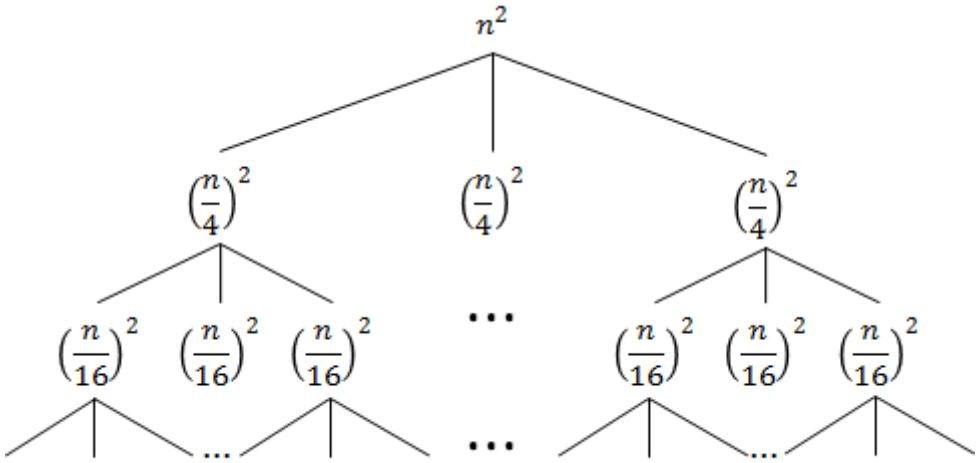
No momento desta chamada é pago o custo da primeira iteração (n^2) e são disparadas 3 recursões onde cada uma terá ¼ do tamanho do problema na iteração anterior.



Uma vez que nas três novas chamadas recursivas o tamanho do problema ainda não atinge o critério de parada, o processo recursivo é executado novamente. Neste caso, cada uma das chamadas dispara novas três recursões. No momento em que ocorrem as novas chamadas, cada uma deve pagar o custo a ela atrelado $\left(\frac{n}{4}\right)^2$.

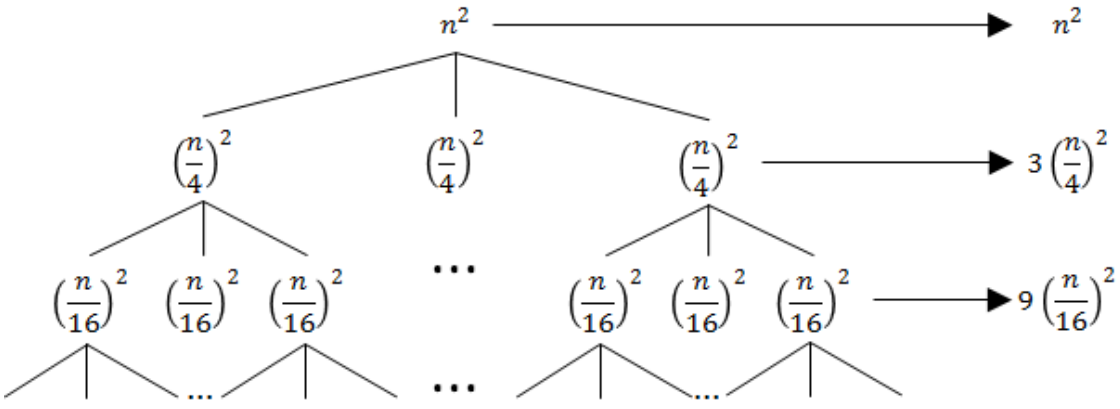


O processo é executado recursivamente até que o critério de parada seja atingido.



Visto que para se determinar o custo dispendido pela recursão é necessário efetuar a soma dos elementos que compõe a árvore, faz-se necessário determinar o custo envolvido na execução de cada nível da árvore. Analisando-se o comportamento da árvore pode-se observar um padrão no crescimento dos novos níveis:

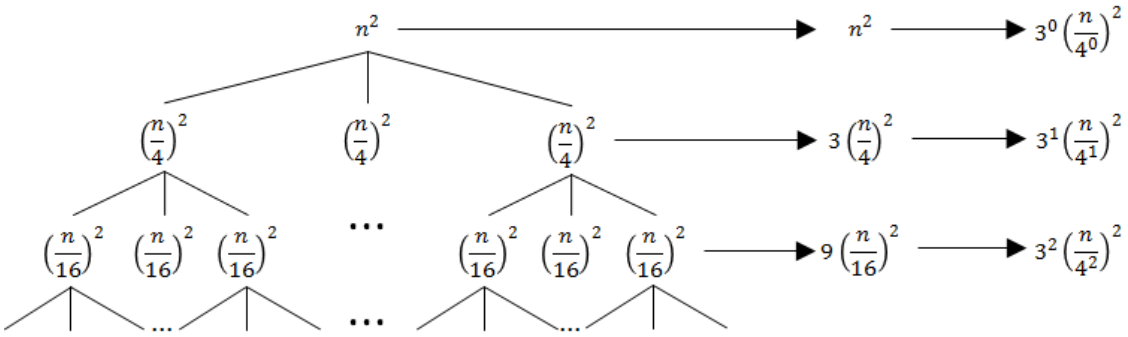
a) Cada novo nível é composto por 3x mais problemas que o nível anterior;
b) Cada novo subproblema tem ¼ do tamanho dos problemas do nível acima;



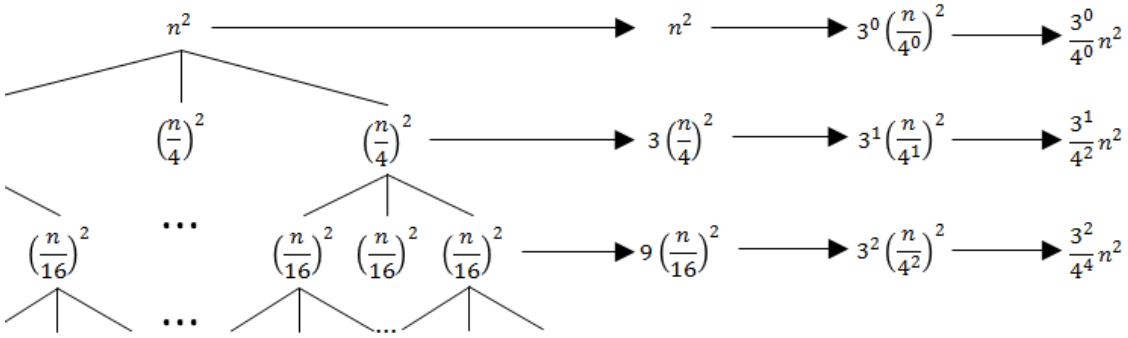
Verifica-se então que o número de subproblemas cresce de acordo com o aumento do nível dos elementos em uma potência de base três, ou seja, $3^{\text{nível}}$.

O tamanho dos subproblemas também está relacionado ao nível em que está posicionado: quanto mais profundo estiver o elemento, menor será o tamanho do

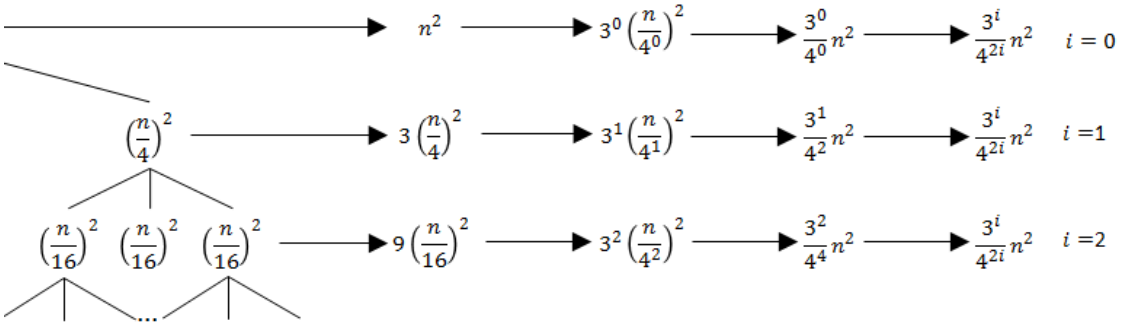
problema a ser resolvido. A variação no tamanho dos subproblemas se dá em uma potência de base quatro, $4^{\text{nível}}$.



Separando n do crescimento dos elementos fixos 3 e 4:



O comportamento dos elementos fixos em relação à profundidade i então:



Dessa forma, observou-se que o custo de cada nível é definido pelo termo $\frac{3^i}{4^{2i}}n^2$ em que i corresponde ao índice do nível.

Descoberto o custo de cada nível o passo seguinte seria somar todos os níveis presentes na árvore. No entanto, é necessário determinar quantos níveis compõe a estrutura.

Para calcular a altura da árvore é usado o caso base como critério, uma vez que o problema é dividido até que o tamanho do subproblema atinja o valor conhecido, neste caso, n=1.

O desenvolvimento da recursão mostra que o tamanho dos subproblemas varia de acordo com o termo $\frac{n}{4^i}$ em que i corresponde ao nível da árvore:

$$T(n) \rightarrow T\left(\frac{n}{4}\right) \rightarrow T\left(\frac{n}{16}\right) = T\left(\frac{n}{4^i}\right)$$

O processo recursivo é executado até que o problema possua tamanho 1, quando alcança o caso base, ou seja, até que $\frac{n}{4^i} = 1$. Assim pode-se descobrir qual é altura das folhas da árvore e consequentemente a sua altura.

$$\frac{n}{4^i} = 1$$

$$n = 4^i$$

Usando a propriedade logarítmica $a^x = b \rightarrow \quad log_a b = x$

$$i = log_4 n$$

Assim, a altura das folhas (e da árvore) será $log_4 n$.

O cálculo do custo da árvore é dividido em dois segmentos: o primeiro segue o comportamento especificado anteriormente ($\frac{3^i}{4^{2i}} n^2$) desde a raiz até o penúltimo nível da árvore. O segundo segmento envolve apenas o último nível da árvore. Essa divisão ocorre, pois, nas folhas o custo já é conhecido, conforme o caso base da recursão. Assim:

$$T(n) = Custo\left(Raiz \rightarrow (Altura - 1)\right) + Custo(Folhas)$$

Sabe-se que cada altura da árvore possui 3^i elementos, em que i corresponde ao nível em análise. Para sabermos a quantidade de folhas da árvore, substituímos o índice do nível pela altura ($log_4 n$) em que as folhas estão:

$$3^{log_4 n}$$

Usando a consequência logarítmica $c^{log_a b} \quad \rightarrow \quad b^{log_a c}$

Assim, a quantidade de folhas é $n^{log_4 3} \rightarrow \ n^{0,7925}$

Como, segundo o caso base, o custo de cada subproblema é de uma unidade quando o tamanho do problema é igual a 1, então o custo para se executar o último nível da árvore (folhas) será $1 * n^{0,7925} = n^{0,7925}$.

O custo total dos níveis pode então ser definido da seguinte forma:

$$T(n) = Custo\left(Raiz \rightarrow (Altura - 1)\right) + Custo(Folhas)$$

$$T(n) = Custo\left(Raiz \rightarrow (Altura - 1)\right) + n^{0,7925})$$

$$T(n) = \left(\frac{3}{4^2}\right)^0 n^2 + \left(\frac{3}{4^2}\right)^1 n^2 + \left(\frac{3}{4^2}\right)^2 n^2 + \cdots + \left(\frac{3}{4^2}\right)^{log_4 n - 1} n^2 + n^{0,7925})$$

$$T(n) = \sum_{i=0}^{log_4 n - 1} \left(\frac{3}{4^2}\right)^i n^2 + n^{0,7925}$$

Usando a propriedade da soma dos termos de uma série exponencial

$$\sum_{k=0}^{m} x^k = \frac{x^{m+1} - 1}{x - 1}$$

Neste cenário consideramos que $x = \left(\frac{3}{4^2}\right)$ e $m = log_4 n - 1$, então

$$T(n) = \frac{\left(\frac{3}{4^2}\right)^{log_4 n - 1 + 1} - 1}{\left(\frac{3}{4^2}\right) - 1} n^2 + n^{0,7925}$$

$$T(n) = \frac{\left(\frac{3}{4^2}\right)^{log_4 n - 1 + 1} - 1}{-0,8125} n^2 + n^{0,7925}$$

Pela regra da potência de uma potência $(a^k)^m = a^{km}$

$$T(n) = \frac{\frac{3^{log_4 n}}{(4^2)^{log_4 n}} - 1}{-0,8125} n^2 + n^{0,7925}$$

$$T(n) = \frac{\frac{n^{log_4 3}}{(4^{log_4 n})^2} - 1}{-0,8125} n^2 + n^{0,7925}$$

$$T(n) = \frac{\frac{n^{0,7925}}{(n^{log_4 4})^2} - 1}{-0,8125} n^2 + n^{0,7925}$$

$$T(n) = \frac{\frac{n^{0,7925}}{(n^1)^2} - 1}{-0,8125} n^2 + n^{0,7925}$$

$$T(n) = \frac{\frac{n^{0,7925}}{n^2} - 1}{-0,8125} n^2 + n^{0,7925}$$

$$T(n) = \frac{n^{0,7925-2} - 1}{-0,8125} n^2 + n^{0,7925}$$

$$T(n) = \frac{n^{-1,2075} - 1}{-0,8125} n^2 + n^{0,7925}$$

$$T(n) = \frac{n^{2-1,2075}}{-0,8125} + \frac{-n^2}{-08125} + n^{0,7925}$$

$$T(n) = \frac{n^2}{08125} - \frac{n^{0,7925}}{0,8125} + n^{0,7925}$$

$$T(n) = 1,231n^2 - 1,231n^{0,7925} + n^{0,7925}$$

$$\boldsymbol{T(n) = 1,231n^2 - 0,231n^{0,7925}}$$

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