Formando Relações de Recorrência

$T(n) = c$ se $n = 1$ ou $n = 2$ (1)
$T(n) = T(n-1) + T(n-2) + b$ se $n > 2$ (2)
Determinando o limite inferior de T(n):

Expandindo:
$$
\begin{aligned}
T(n) &= T(n-1) + T(n-2) + b \\
&\geq T(n-2) + T(n-2) + b \\
&= 2T(n-2) + b \\
&= 2[T(n-3) + T(n-4) + b] + b && \text{substituindo } T(n-2) \text{ em (2)} \\
&\geq 2[T(n-4) + T(n-4) + b] + b \\
&= 2^2T(n-4) + 2b + b \\
&= 2^2[T(n-5) + T(n-6) + b] + 2b + b && \text{substituindo } T(n-4) \text{ em (2)} \\
&\geq 2^3T(n-6) + (2^2 + 2^1 + 2^0)b \\
&\;\;\ldots \\
&\geq 2^kT(n-2k) + (2^{k-1} + 2^{k-2} + \ldots + 2^1 + 2^0)b \\
&= 2^kT(n-2k) + (2^k - 1)b
\end{aligned}
$$

O caso base ocorre quando $n - 2k = 2 \rightarrow k = (n-2)/2$

Logo
$$
\begin{aligned}
T(n) &\geq 2^{(n-2)/2}\,T(2) + [2^{(n-2)/2} - 1]b \\
&= (b + c)2^{(n-2)/2} - b \\
&= [(b + c)/2]*(2)^{n/2} - b && \rightarrow \text{Fibonacci recursivo é exponencial}
\end{aligned}
$$

Método da Substituição

Método Geral:

1.Imaginar a forma da solução

2.Verificar por indução

3.Resolver para as constantes

Método da Substituição

Método Geral:

1.Imaginar a forma da solução

2.Verificar por indução

3.Resolver para as constantes

Exemplo: $T(n) = 4T(n/2) + n$

- [Assume $T(1) = \Theta(1)$.]
- Imagine $O(n^3)$. (Prove O e Ω separados)
- Assume que $T(k) \leq ck^3$ para $k < n$.
- Prove $T(n) \leq cn^3$ provar por indução

Exemplo de Substituição

$$\begin{aligned} T(n) &= 4T(n/2) + n \qquad k = n/2 \quad \text{supõe } T(k) \leq ck^3 \\ &\leq 4c(n/2)^3 + n \\ &= (c/2)n^3 + n \\ &= cn^3 - ((c/2)n^3 - n) \leftarrow \text{desejado} - \text{residual} \\ &\leq cn^3 \leftarrow \text{desejado} \end{aligned}$$

Quando $(c/2)n^3 - n \geq 0,$ por exemplo,
se $c \geq 2$ e $n \geq 1.$

residual

Exemplo

- Temos de ajustar o caso base na indução
- ***Base:*** $T(n) = \Theta(1)$ para todo $n < n_0$, onde n_0 é uma constante adequada
- Para $1 \le n < n_0$, temos "$\Theta(1)$" $\le cn^3$, se pegamos um c grande o suficiente

Não é um limite estreito o suficiente

Limite superior estreito

Devemos provar $T(n) = O(n^2)$.

Limite superior estreito

Devemos provar $T(n) = O(n^2)$.

Assuma que $T(k) \leq ck^2$ para $k < n$:

$$\begin{aligned} T(n) &= 4T(n/2) + n \\ &\leq 4c(n/2)^2 + n \\ &= cn^2 + n \\ &= O(n^2) \end{aligned}$$

Limite superior estreito

Devemos provar $T(n) = O(n^2)$.

Assuma que $T(k) \leq ck^2$ para $k < n$:

$$
\begin{aligned}
T(n) &= 4T(n/2) + n \\
&\leq 4c(n/2)^2 + n \\
&= cn^2 + n \\
&= \cancel{O(n^2)}
\end{aligned}
$$

Errado! Nós devemos provar a indução

Limite superior estreito

Devemos provar $T(n) = O(n^2)$.

Assuma que $T(k) \leq ck^2$ para $k < n$:

$$
\begin{aligned}
T(n) &= 4T(n/2) + n \\
&\leq 4c(n/2)^2 + n \\
&= cn^2 + n \\
&= \cancel{O(n^2)} \quad \textbf{\textit{Errado! Nós devemos provar a indução}} \\
&= cn^2 - (-n) \; [\text{ desejado - residual}] \\
&\leq cn^2 \quad \text{para nenhuma escolha com } c > 0\textbf{!}
\end{aligned}
$$

Limite superior estreito

Ideia: Fortalecer a hipótese indutiva.

• ***Subtrair*** um termo de baixa ordem.

Hipótese indutiva : $T(k) \leq c_1 k^2 - c_2 k$ para $k < n$.

Limite superior estreito

Ideia: Fortalecer a hipótese indutiva.

- ***Subtrair*** um termo de baixa ordem.

Hipótese indutiva : $T(k) \le c_1 k^2 - c_2 k$ para $k < n$.

$$
\begin{aligned}
T(n) &= 4T(n/2) + n \qquad k = n/2 \\
&\le 4(c_1 (n/2)^2 - c_2 (n/2)) + n \\
&= c_1 n^2 - 2c_2 n + n \\
&= c_1 n^2 - c_2 n - (c_2 n - n) \\
&\le c_1 n^2 - c_2 n \text{ se } c_2 \ge 1.
\end{aligned}
$$

Limite superior estreito

Ideia: Fortalecer a hipótese indutiva.

• ***Subtrair*** um termo de baixa ordem.

Hipótese indutiva : $T(k) \leq c_1 k^2 - c_2 k$ para $k < n$.

$$
\begin{aligned}
T(n) &= 4T(n/2) + n \qquad k = n/2 \\
&\leq 4(c_1(n/2)^2 - c_2(n/2)) + n \\
&= c_1 n^2 - 2c_2 n + n \\
&= c_1 n^2 - c_2 n - (c_2 n - n) \\
&\leq c_1 n^2 - c_2 n \text{ se } c_2 \geq 1.
\end{aligned}
$$

Com um c_1 que seja suficiente para as condições iniciais

Método da árvore de recursão

- Uma árvore de recursão modela o custo em tempo de uma execução recursiva de um algoritmo.
- A árvore de recursão pode não ser totalmente confiável já que deixa passos indicados
- O método da árvore de recursão promove a intuição do processo
- Se torna útil para gerar tentativas de soluções para os outros métodos

Recorrência para o mergesort

$$T(n) = \begin{cases} \Theta(1) \text{ se } n = 1; \\ 2T(n/2) + \Theta(\text{n}) \text{ se } n > 1. \end{cases}$$

O caso base pode ser ignorado assumindo que ele corresponde a uma constante maior que zero

Árvore de Recorrência

Resolver $T(n) = 2T(n/2) + dn$, com $d > 0$ constante

Árvore de Recorrência

Resolver $T(n) = 2T(n/2) + dn$, com $d > 0$ constante

$$T(n)$$

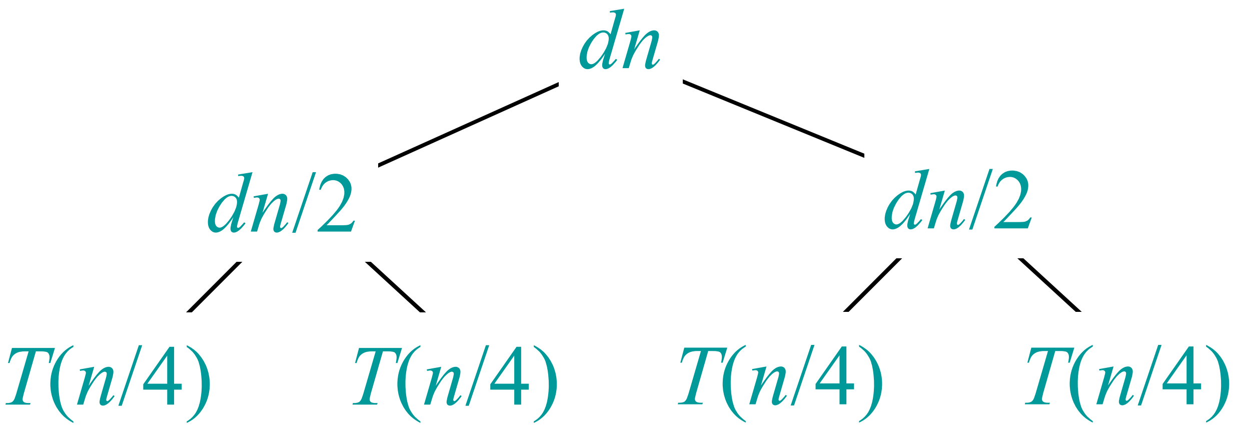
Árvore de Recorrência

Resolver $T(n) = 2T(n/2) + dn$, com $d > 0$ constante

$$\begin{array}{ccc} & dn & \\ T(n/2) & & T(n/2) \end{array}$$

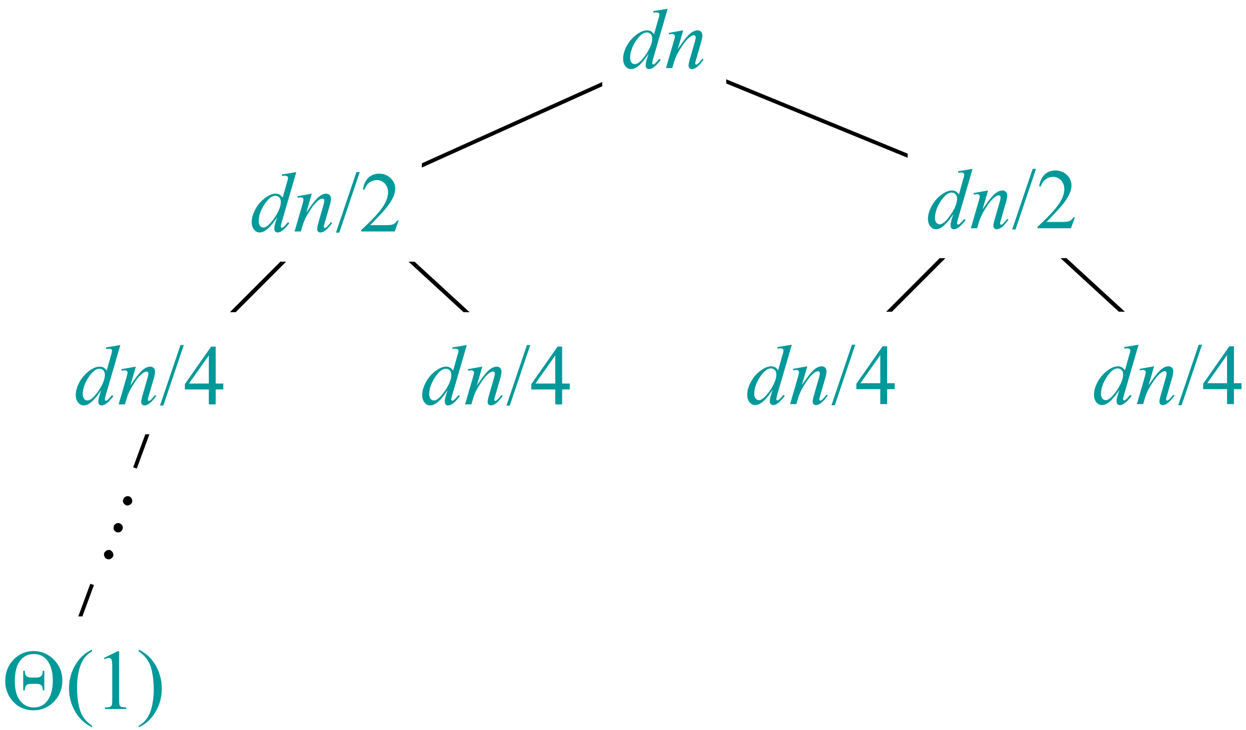
Árvore de Recorrência

Resolver $T(n) = 2T(n/2) + dn$, com $d > 0$ constante



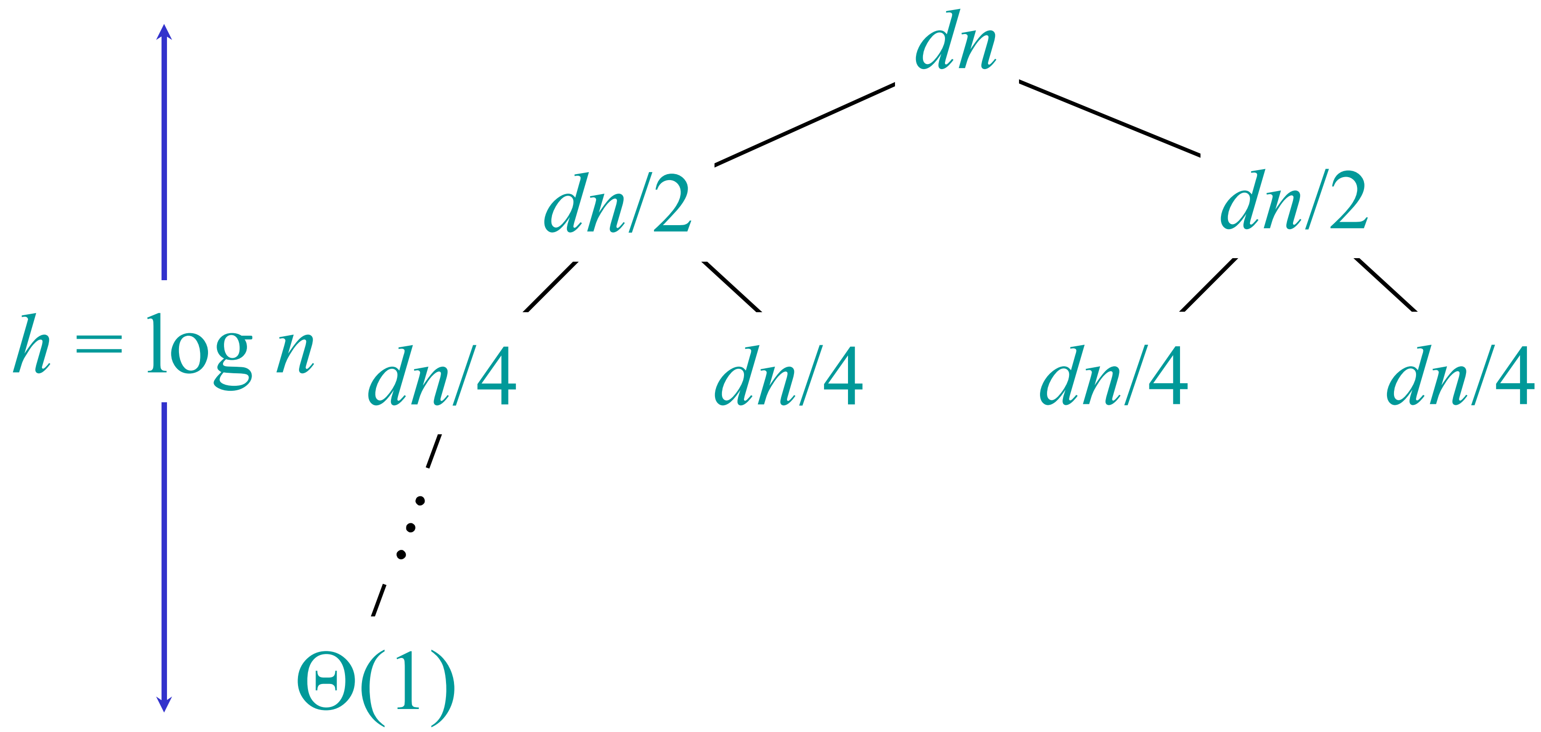
Árvore de Recorrência

Resolver $T(n) = 2T(n/2) + dn$, com $d > 0$ constante



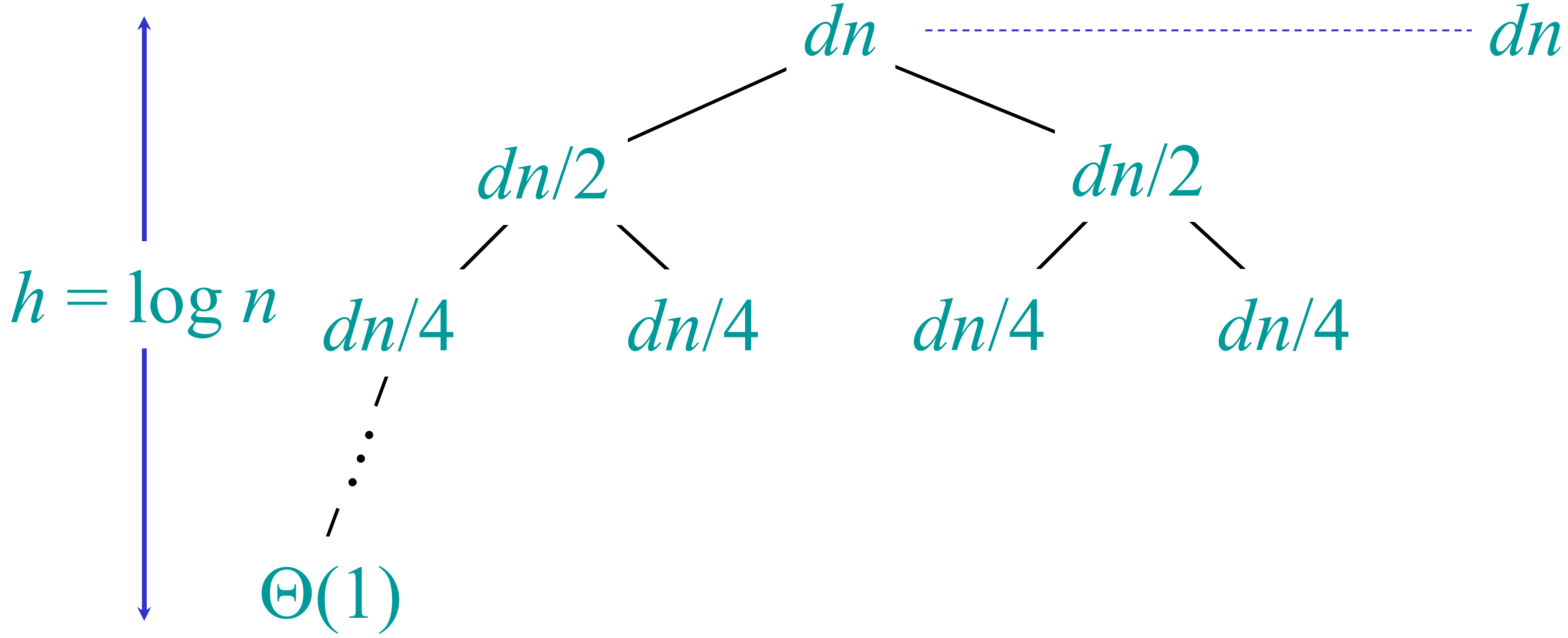
Árvore de Recorrência

Resolver $T(n) = 2T(n/2) + dn$, com $d > 0$ constante



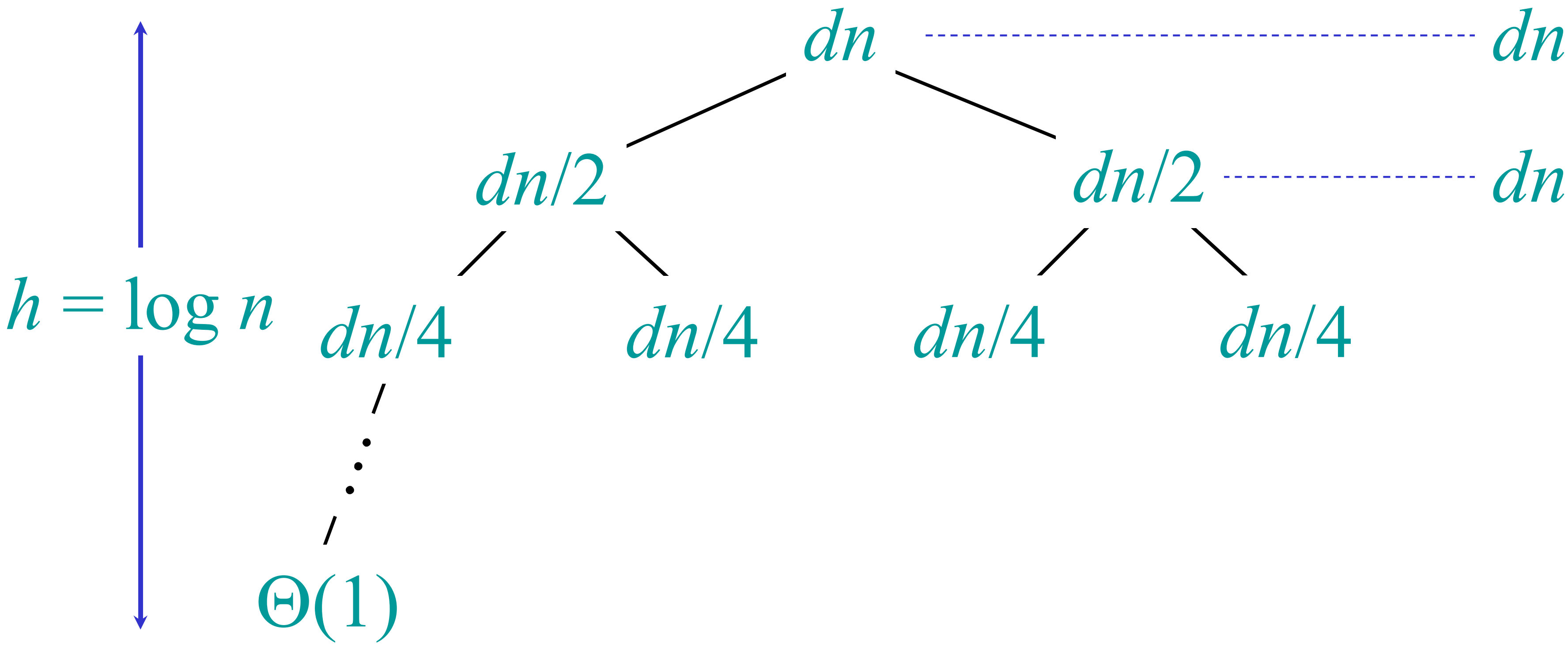
Árvore de Recorrência

Resolver $T(n) = 2T(n/2) + dn$, com $d > 0$ constante



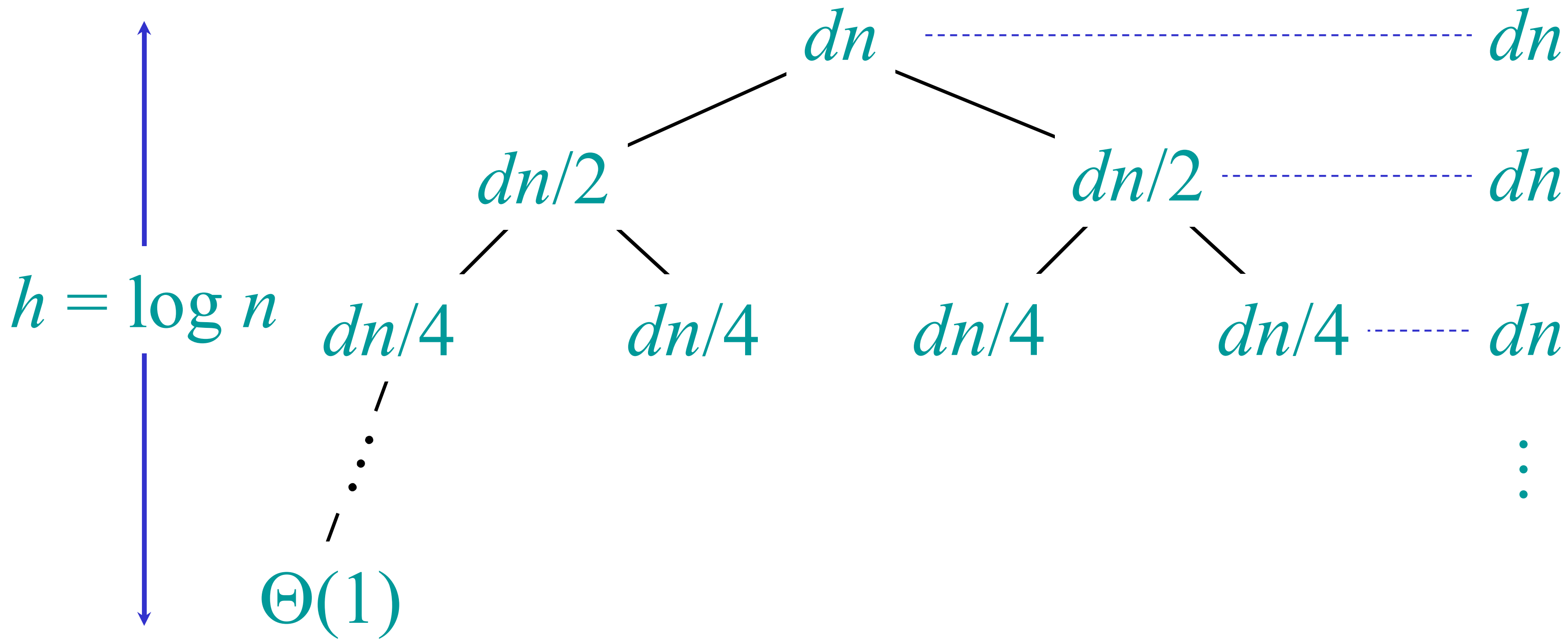
Árvore de Recorrência

Resolver $T(n) = 2T(n/2) + dn$, com $d > 0$ constante



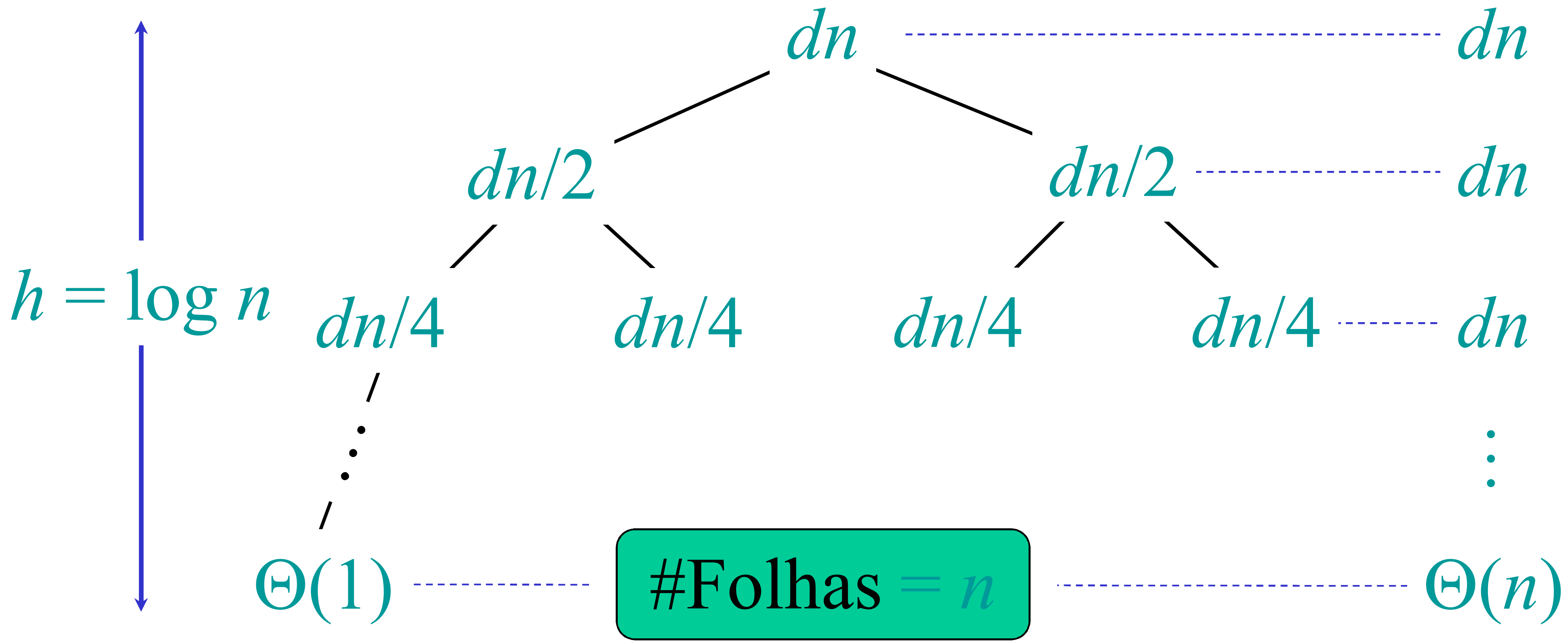
Árvore de Recorrência

Resolver $T(n) = 2T(n/2) + dn$, com $d > 0$ constante



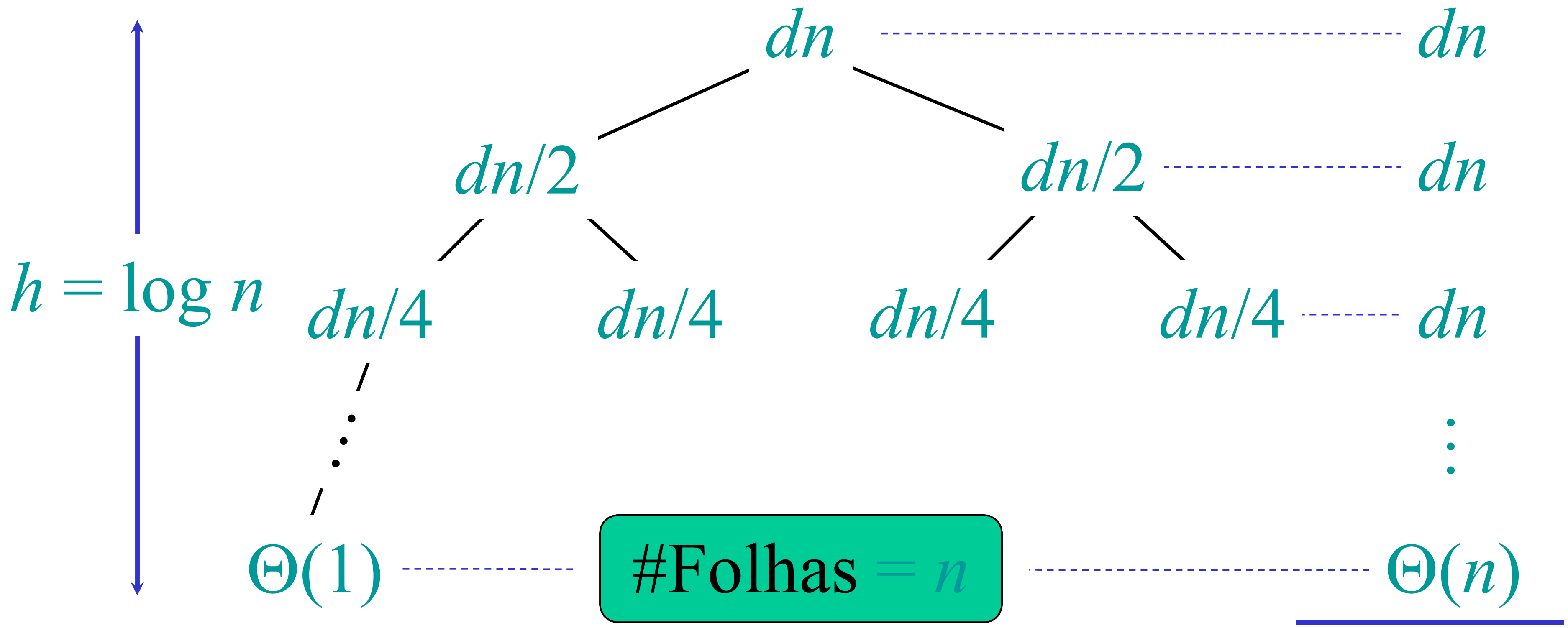
Árvore de Recorrência

Resolver $T(n) = 2T(n/2) + dn$, com $d > 0$ constante



Árvore de Recorrência

Resolver $T(n) = 2T(n/2) + dn$, com $d > 0$ constante



O d pode ser ignorado em certos casos

Total $\Theta(n \log n)$

Método da Iteração

- $T(n) = T(n/2) + 1$

$$= T(n/4) + 1 + 1$$

$$= T(n/8) + 1 + 1 + 1$$

$$= T(1) + \underbrace{1 + 1 + \dots + 1}_{\log(n)}$$

$$= \Theta\,(\log(n))$$

Método da Iteração

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

$$= T(n-2) + 1 + 1$$

$$= T(n-3) + 1 + 1 + 1$$

$$= T(1) + \underbrace{1 + 1 + \ldots + 1}_{n\,-\,1}$$

$$= \Theta\,(n)$$

Método da Iteração

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

  $= T(n-2) + (n-1) + n$

  $= T(n-3) + (n-2) + (n-1) + n$

  $= T(1) + 2 + 3 + \ldots + n$

  $= \Theta(n^2)$

Exemplo 2 de árvore de recursão

Resolver $T(n) = T(n/4) + T(n/2) + n^2$:

Exemplo 2 de árvore de recursão

Resolver $T(n) = T(n/4) + T(n/2) + n^2$:

$$T(n)$$

Exemplo 2 de árvore de recursão

Resolver $T(n) = T(n/4) + T(n/2) + n^2$:

$$
\begin{array}{ccc}
 & n^2 & \\
T(n/4) & & T(n/2)
\end{array}
$$

Exemplo 2 de árvore de recursão