Análise de algoritmos

Recorrências

Conteúdo

Introdução

O tempo de execução de um algoritmo recursivo pode frequentemente ser descrito por uma equação de recorrência.

Definição

Uma **recorrência** é uma equação ou desigualdade que descreve o seu valor em termos de seu valor em entradas menores.

Exemplo

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

Definição

Resolver uma equação de recorrência significa obter limites assintóticos para ela.

Simplificações

1. Ignorar que n é inteiro
2. Omissão da condição limite
3. Omissão de pisos e tetos
4. Exemplo da equação de recorrência do merge-sort

$$T(n) = \begin{cases} \Theta(1) & \text{se } n = 1, \\ T(\lceil n/2 \rceil) + T(\lfloor n/2 \rfloor) + \Theta(n) & \text{se } n > 1. \end{cases}$$

versos

$$T(n) = 2T(n/2) + \Theta(n)$$

O método de substituição

O método consiste em duas etapas:

1. Pressupor a forma da solução.
2. Usar indução matemática para encontrar as constantes e mostrar que a solução funciona.

O método de substituição

Exemplo

Vamos determinar um limite superior sobre a recorrência

$$T(n) = 2T(\lfloor n/2 \rfloor) + n$$

O método de substituição

Exemplo

Vamos determinar um limite superior sobre a recorrência

$$T(n) = 2T(\lfloor n/2 \rfloor) + n$$

Solução no quadro.

O método de substituição

- Como fazer um bom palpite?

O método de substituição

- Como fazer um bom palpite?
  - Não há nenhum modo geral
  - Experiência e criatividade
  - Utilização de árvores de recorrências
  - Semelhança com recorrências já conhecidas. Qual é um bom palpite para $T(n) = 2T(\lfloor n/2 \rfloor + 17) + n$?
  - Provar limites superiores e inferiores e então reduzir o intervalo

O método de substituição

- Sutilezas
  - O limite está correto, mas a matemática não funciona
  - Exemplo $T(n) = T(\lfloor n/2 \rfloor) + T(\lceil n/2 \rceil) + 1$
  - Vamos supor que a solução seja $O(n)$ e tentar mostrar que $T(n) \leq cn$, para alguma constante c. Substituindo a suposição na recorrência, obtemos

$$\begin{aligned} T(n) &\leq c\lfloor n/2 \rfloor + c\lceil n/2 \rceil + 1 \\ &= cn + 1 \end{aligned}$$

O método de substituição

- Sutilezas
  - O limite está correto, mas a matemática não funciona
  - Exemplo $T(n) = T(\lfloor n/2 \rfloor) + T(\lceil n/2 \rceil) + 1$
  - Vamos supor que a solução seja $O(n)$ e tentar mostrar que $T(n) \leq cn$, para alguma constante c. Substituindo a suposição na recorrência, obtemos

$$\begin{aligned} T(n) &\leq c\lfloor n/2 \rfloor + c\lceil n/2 \rceil + 1 \\ &= cn + 1 \end{aligned}$$

o que não implica que $T(n) \leq cn$.

O método de substituição

- Sutilezas
  - O limite está correto, mas a matemática não funciona
  - Exemplo $T(n) = T(\lfloor n/2 \rfloor) + T(\lceil n/2 \rceil) + 1$
  - Vamos supor que a solução seja $O(n)$ e tentar mostrar que $T(n) \leq cn$, para alguma constante c. Substituindo a suposição na recorrência, obtemos

    $$\begin{aligned} T(n) &\leq c\lfloor n/2 \rfloor + c\lceil n/2 \rceil + 1 \\ &= cn + 1 \end{aligned}$$

    o que não implica que $T(n) \leq cn$.
  - Neste caso, devemos utilizar uma hipótese indutiva mais forte.
  - Nossa nova suposição é $T(n) \leq cn - b$, onde $b \geq 0$ é constante. Obtemos

    $$\begin{aligned} T(n) &\leq (c\lfloor n/2 \rfloor - b) + (c\lceil n/2 \rceil - b) + 1 \\ &= cn - 2b + 1 \\ &\leq cn - b, \end{aligned}$$

    deste que $b \geq 1$.

O método de substituição

- Como evitar armadilhas
  - É fácil errar na utilização da notação assintótica
  - Vamos provar falsamente que $T(n) = 2T(\lfloor n/2 \rfloor) + n$ é $O(n)$
  - Supondo que $T(n) = O(n)$, obtemos

$$\begin{aligned} T(n) &\leq 2(c\lfloor n/2 \rfloor) + n \\ &\leq cn + n \\ &= O(n) \qquad \text{errado!} \end{aligned}$$

O método de substituição

- Como evitar armadilhas
  - É fácil errar na utilização da notação assintótica
  - Vamos provar falsamente que $T(n) = 2T(\lfloor n/2 \rfloor) + n$ é $O(n)$
  - Supondo que $T(n) = O(n)$, obtemos

$$\begin{aligned} T(n) &\leq 2(c\lfloor n/2 \rfloor) + n \\ &\leq cn + n \\ &= O(n) \qquad \text{errado!} \end{aligned}$$

  - O erro foi não provarmos a forma exata da hipótese indutiva, ou seja, que $T(n) \leq cn$

O método de substituição

- Como trocar variáveis
  - Qual é um bom palpite para $T(n) = 2T(\lfloor\sqrt{n}\rfloor) + \lg n$?

O método de substituição

- Como trocar variáveis
  - Qual é um bom palpite para $T(n) = 2T(\lfloor\sqrt{n}\rfloor) + \lg n$?
  - Vamos renomear $m = \lg n$, obtemos $T(2^m) = 2T(2^{m/2}) + m$

O método de substituição

- Como trocar variáveis
  - Qual é um bom palpite para $T(n) = 2T(\lfloor\sqrt{n}\rfloor) + \lg n$?
  - Vamos renomear $m = \lg n$, obtemos $T(2^m) = 2T(2^{m/2}) + m$
  - Vamos renomear $S(m) = T(2^m)$, obtemos $S(m) = 2S(m/2) + m$

O método de substituição

- Como trocar variáveis
  - Qual é um bom palpite para $T(n) = 2T(\lfloor\sqrt{n}\rfloor) + \lg n$?
  - Vamos renomear $m = \lg n$, obtemos $T(2^m) = 2T(2^{m/2}) + m$
  - Vamos renomear $S(m) = T(2^m)$, obtemos $S(m) = 2S(m/2) + m$
  - Semelhante a recorrência 4.4. A solução é $S = O(m \lg m)$

O método de substituição

- Como trocar variáveis
  - Qual é um bom palpite para $T(n) = 2T(\lfloor\sqrt{n}\rfloor) + \lg n$?
  - Vamos renomear $m = \lg n$, obtemos $T(2^m) = 2T(2^{m/2}) + m$
  - Vamos renomear $S(m) = T(2^m)$, obtemos $S(m) = 2S(m/2) + m$
  - Semelhante a recorrência 4.4. A solução é $S = O(m \lg m)$
  - Trocando de volta $S(m)$ por $T(n)$, obtemos $T(n) = T(2^m) = S(m) = O(m \lg m) = O(\lg n \lg \lg n)$

Exercícios

4.1-1 Mostre que a solução de $T(n) = T(\lceil n/2 \rceil) + 1$ é $O(\lg n)$.

4.1-2 Vimos que a solução de $T(n) = 2T(\lfloor n/2 \rfloor) + n$ é $O(n \lg n)$. Mostre que a solução desta recorrência também é $\Omega(n \lg n)$. Conclua que a solução é $\Theta(n \lg n)$.

4.1-3 Mostre que, supondo uma hipótese indutiva diferente, podemos superar a dificultade com a condição limite $T(1) = 1$ para a recorrência (4.4), sem ajustar as condições limite para a prova indutiva.

4.1-4 Mostre que $\Theta(n \lg n)$ é a solução para a recorrência "exata" (4.2) para a ordenação por intercalação.

4.1-5 Mostre que a solução para $T(n) = 2T(\lfloor n/2 \rfloor + 17) + n$ é $O(n \lg n)$.

4.1-6 Resolva a recorrência $T(n) = 2T(\sqrt{n})$, fazendo uma troca de variáveis. Sua solução deve ser assintoticamente restrita. Não se preocupe em saber se os valores são integrais.

O método da árvore de recursão

Definição

Em uma **árvore de recursão**, cada nó representa o custo de um único subproblema em algum lugar no conjunto de invocações de funções recursivas.

No método da árvore de recursão, somamos os custos dentro de cada nível da árvore para obter um conjunto de custos por nível, e então somamos todos os custos por nível para determinar o custo total de todos os níveis da recursão.

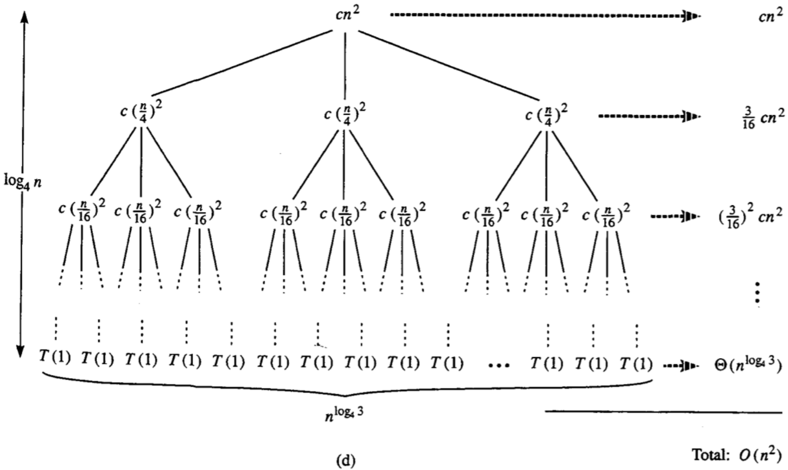
O método da árvore de recursão

- Uma árvore de recursão é usada para gerar uma boa suposição, que é então verificada pelo método de substituição
- Como estamos gerando uma suposição, podemos tolerar algumas “sujeiras”
- Vamos utilizar uma árvore de recursão para encontrar uma suposição para a recorrência

$$T(n) = 3T(\lfloor n/4 \rfloor) + \Theta(n^2)$$

- Para simplificar o processo, vamos supor que n é uma potência de 4 (sujeira)
- Também vamos desconsiderar a função piso (sujeira)

O método da árvore de recursão



O método da árvore de recursão

- A que distância da raiz o tamanho do problema alcança a condição limite?

O método da árvore de recursão

- A que distância da raiz o tamanho do problema alcança a condição limite?
  - O tamanho do subproblema para um nó na profundidade i é $n/4^i$
  - Deste modo o tamanho do subproblema chega a 1 quando $n/4^i = 1$, ou seja, $i = \log_4 n$

O método da árvore de recursão

- A que distância da raiz o tamanho do problema alcança a condição limite?
  - O tamanho do subproblema para um nó na profundidade i é $n/4^i$
  - Deste modo o tamanho do subproblema chega a 1 quando $n/4^i = 1$, ou seja, $i = \log_4 n$
- Quantos níveis tem a árvore?

O método da árvore de recursão

- A que distância da raiz o tamanho do problema alcança a condição limite?
  - O tamanho do subproblema para um nó na profundidade i é $n/4^i$
  - Deste modo o tamanho do subproblema chega a 1 quando $n/4^i = 1$, ou seja, $i = \log_4 n$
- Quantos níveis tem a árvore?
  - A árvore tem $\log_4 n + 1$ níveis $(0, 1, 2, \ldots, \log_4 n)$

O método da árvore de recursão

- A que distância da raiz o tamanho do problema alcança a condição limite?
  - O tamanho do subproblema para um nó na profundidade i é $n/4^i$
  - Deste modo o tamanho do subproblema chega a 1 quando $n/4^i = 1$, ou seja, $i = \log_4 n$
- Quantos níveis tem a árvore?
  - A árvore tem $\log_4 n + 1$ níveis $(0, 1, 2, \ldots, \log_4 n)$
- Qual é o custo em cada nível da árvore?

O método da árvore de recursão

- A que distância da raiz o tamanho do problema alcança a condição limite?
  - O tamanho do subproblema para um nó na profundidade i é $n/4^i$
  - Deste modo o tamanho do subproblema chega a 1 quando $n/4^i = 1$, ou seja, $i = \log_4 n$
- Quantos níveis tem a árvore?
  - A árvore tem $\log_4 n + 1$ níveis $(0, 1, 2, \ldots, \log_4 n)$
- Qual é o custo em cada nível da árvore?
  - Cada nível tem três vezes mais nós do que o nível acima dele

O método da árvore de recursão

- A que distância da raiz o tamanho do problema alcança a condição limite?
  - O tamanho do subproblema para um nó na profundidade i é $n/4^i$
  - Deste modo o tamanho do subproblema chega a 1 quando $n/4^i = 1$, ou seja, $i = \log_4 n$
- Quantos níveis tem a árvore?
  - A árvore tem $\log_4 n + 1$ níveis $(0, 1, 2, \ldots, \log_4 n)$
- Qual é o custo em cada nível da árvore?
  - Cada nível tem três vezes mais nós do que o nível acima dele
  - Assim, o número de nós na profundidade i é 3^i

O método da árvore de recursão

- A que distância da raiz o tamanho do problema alcança a condição limite?
  - O tamanho do subproblema para um nó na profundidade i é $n/4^i$
  - Deste modo o tamanho do subproblema chega a 1 quando $n/4^i = 1$, ou seja, $i = \log_4 n$
- Quantos níveis tem a árvore?
  - A árvore tem $\log_4 n + 1$ níveis $(0, 1, 2, \ldots, \log_4 n)$
- Qual é o custo em cada nível da árvore?
  - Cada nível tem três vezes mais nós do que o nível acima dele
  - Assim, o número de nós na profundidade i é 3^i
  - O tamanho dos subproblemas se reduzem por um fator de 4 a cada nível, portanto, cada nó na profundidade i tem o custo $c(n/4^i)^2$

O método da árvore de recursão

- A que distância da raiz o tamanho do problema alcança a condição limite?
  - O tamanho do subproblema para um nó na profundidade i é $n/4^i$
  - Deste modo o tamanho do subproblema chega a 1 quando $n/4^i = 1$, ou seja, $i = \log_4 n$
- Quantos níveis tem a árvore?
  - A árvore tem $\log_4 n + 1$ níveis $(0, 1, 2, \ldots, \log_4 n)$
- Qual é o custo em cada nível da árvore?
  - Cada nível tem três vezes mais nós do que o nível acima dele
  - Assim, o número de nós na profundidade i é 3^i
  - O tamanho dos subproblemas se reduzem por um fator de 4 a cada nível, portanto, cada nó na profundidade i tem o custo $c(n/4^i)^2$
  - O custo de todos os nós na profundidade i, para $i = 0, 1, 2, \ldots, \log_4 n - 1$, é de $3^i c(n/4^i)^2 = (3/16)^i cn^2$

O método da árvore de recursão

- A que distância da raiz o tamanho do problema alcança a condição limite?
  - O tamanho do subproblema para um nó na profundidade i é $n/4^i$
  - Deste modo o tamanho do subproblema chega a 1 quando $n/4^i = 1$, ou seja, $i = \log_4 n$
- Quantos níveis tem a árvore?
  - A árvore tem $\log_4 n + 1$ níveis $(0, 1, 2, \ldots, \log_4 n)$
- Qual é o custo em cada nível da árvore?
  - Cada nível tem três vezes mais nós do que o nível acima dele
  - Assim, o número de nós na profundidade i é 3^i
  - O tamanho dos subproblemas se reduzem por um fator de 4 a cada nível, portanto, cada nó na profundidade i tem o custo $c(n/4^i)^2$
  - O custo de todos os nós na profundidade i, para $i = 0, 1, 2, \ldots, \log_4 n - 1$, é de $3^i c(n/4^i)^2 = (3/16)^i cn^2$
  - O último nível, na profundidade $\log_4 n$, tem $3^{log_4 n} = n^{\log_4 3}$ nós, cada um contribuindo com $T(1)$ para o custo total, portanto, o custo do último nível é $n^{\log_4 3} T(1) = \Theta(n^{\log_4 3})$

O método da árvore de recursão

- Agora somamos os custos sobre todos os níveis para determinar o custo correspondente à árvore inteira

$$
\begin{aligned}
T(n) &= cn^2 + \frac{3}{16}cn^2 + \left(\frac{3}{16}\right)^2 cn^2 + \cdots + \left(\frac{3}{16}\right)^{\log_4 n - 1} cn^2 \\
&\quad + \Theta(n^{\log_4 3}) \\
&= \sum_{i=0}^{\log_4 n - 1} \left(\frac{3}{16}\right)^i cn^2 + \Theta(n^{\log_4 3}) \\
&= \frac{(3/16)^{\log_4 n} - 1}{(3/16) - 1} cn^2 + \Theta(n^{\log_4 3})
\end{aligned}
$$

O método da árvore de recursão

- Agora somamos os custos sobre todos os níveis para determinar o custo correspondente à árvore inteira

$$
\begin{aligned}
T(n) &= cn^2 + \frac{3}{16}cn^2 + \left(\frac{3}{16}\right)^2 cn^2 + \cdots + \left(\frac{3}{16}\right)^{\log_4 n-1} cn^2 \\
&\quad + \Theta(n^{\log_4 3}) \\
&= \sum_{i=0}^{\log_4 n-1} \left(\frac{3}{16}\right)^i cn^2 + \Theta(n^{\log_4 3}) \\
&= \frac{(3/16)^{\log_4 n} - 1}{(3/16) - 1} cn^2 + \Theta(n^{\log_4 3})
\end{aligned}
$$

- Esta fórmula é um pouco confusa. O que podemos fazer?

O método da árvore de recursão

- Vamos considerar outra sujeira, e trocar o limite $\log_4 n - 1$ da série para ∞.

$$
\begin{aligned}
T(n) &= \sum_{i=0}^{\log_4 n-1} \left(\frac{3}{16}\right)^i cn^2 + \Theta(n^{\log_4 3}) \\
&< \sum_{i=0}^{\infty} \left(\frac{3}{16}\right)^i cn^2 + \Theta(n^{\log_4 3}) \\
&= \frac{1}{1-(3/16)} cn^2 + \Theta(n^{\log_4 3}) \\
&= \frac{16}{13} cn^2 + \Theta(n^{\log_4 3}) \\
&= O(n^2)
\end{aligned}
$$

O método da árvore de recursão

- Vamos considerar outra sujeira, e trocar o limite $\log_4 n - 1$ da série para ∞.

$$
\begin{aligned}
T(n) &= \sum_{i=0}^{\log_4 n - 1} \left(\frac{3}{16}\right)^i cn^2 + \Theta(n^{\log_4 3}) \\
&< \sum_{i=0}^{\infty} \left(\frac{3}{16}\right)^i cn^2 + \Theta(n^{\log_4 3}) \\
&= \frac{1}{1 - (3/16)} cn^2 + \Theta(n^{\log_4 3}) \\
&= \frac{16}{13} cn^2 + \Theta(n^{\log_4 3}) \\
&= O(n^2)
\end{aligned}
$$

- Com este processo, derivamos a suposição de que $T(n) = O(n^2)$

O método da árvore de recursão

- Que nó contribui mais para para o custo total da árvore?

O método da árvore de recursão

- Que nó contribui mais para para o custo total da árvore? O nó raiz, que custa cn^2

O método da árvore de recursão

- Que nó contribui mais para para o custo total da árvore? O nó raiz, que custa cn^2
- O limite superior $O(n^2)$ deve ser um limite restrito. Por quê?

O método da árvore de recursão

- Agora vamos usar o método de substituição para verificar a suposição, isto é, $T(n) = O(n^2)$ é um limite superior para $T(n) = 3T(\lfloor n/4 \rfloor) + \Theta(n^2)$

O método da árvore de recursão

- Agora vamos usar o método de substituição para verificar a suposição, isto é, $T(n) = O(n^2)$ é um limite superior para $T(n) = 3T(\lfloor n/4 \rfloor) + \Theta(n^2)$
- Queremos mostrar que $T(n) \leq dn^2$ para alguma constante $d > 0$. Usando a mesma constante $c > 0$ de antes temos

O método da árvore de recursão

- Agora vamos usar o método de substituição para verificar a suposição, isto é, $T(n) = O(n^2)$ é um limite superior para $T(n) = 3T(\lfloor n/4 \rfloor) + \Theta(n^2)$
- Queremos mostrar que $T(n) \leq dn^2$ para alguma constante $d > 0$. Usando a mesma constante $c > 0$ de antes temos

$$\begin{aligned}
T(n) &\leq 3T(\lfloor n/4 \rfloor) + cn^2 \\
&\leq 3d\lfloor n/4 \rfloor^2 + cn^2 \\
&\leq 3d(n/4)^2 + cn^2 \\
&= \frac{3}{16}dn^2 + cn^2 \\
&\leq dn^2 \qquad \text{desde que } d \geq (16/13)c
\end{aligned}$$

O método da árvore de recursão

- Agora vamos usar o método de substituição para verificar a suposição, isto é, $T(n) = O(n^2)$ é um limite superior para $T(n) = 3T(\lfloor n/4 \rfloor) + \Theta(n^2)$
- Queremos mostrar que $T(n) \leq dn^2$ para alguma constante $d > 0$. Usando a mesma constante $c > 0$ de antes temos

$$\begin{aligned}
T(n) &\leq 3T(\lfloor n/4 \rfloor) + cn^2 \\
&\leq 3d\lfloor n/4 \rfloor^2 + cn^2 \\
&\leq 3d(n/4)^2 + cn^2 \\
&= \frac{3}{16}dn^2 + cn^2 \\
&\leq dn^2 \qquad \text{desde que } d \geq (16/13)c
\end{aligned}$$

- Portanto, confirmamos que a suposição $T(n) = O(n^2)$, encontrada pelo método da árvore de recursão, é verdadeira

O método da árvore de recursão

- Vamos usar uma árvore de recursão para encontrar um limite superior para

$$T(n) = T(n/3) + T(2n/3) + O(n)$$

O método da árvore de recursão

- Vamos usar uma árvore de recursão para encontrar um limite superior para

$$T(n) = T(n/3) + T(2n/3) + O(n)$$

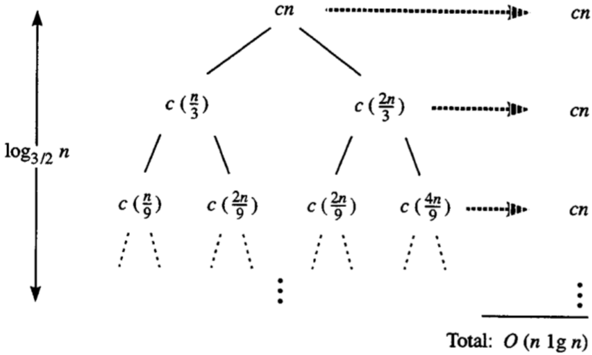
- Vamos desenha a árvore de recursão

O método da árvore de recursão

- Vamos usar uma árvore de recursão para encontrar um limite superior para

$$T(n) = T(n/3) + T(2n/3) + O(n)$$

- Vamos desenha a árvore de recursão



O método da árvore de recursão

- Qual é o custo de cada nível?

O método da árvore de recursão

- Qual é o custo de cada nível? cn

O método da árvore de recursão

- Qual é o custo de cada nível? cn
- Qual é a altura da árvore?

O método da árvore de recursão

- Qual é o custo de cada nível? cn
- Qual é a altura da árvore? O caminho mais longo da raiz até uma folha é $n \to (2/3)n \to (2/3)^2 n \to \cdots \to 1$. Tendo em vista que $(2/3)^i n = 1$ quando $i = \log_{3/2} n$, a altura da árvore é $\log_{3/2} n$.

O método da árvore de recursão

- Qual é o custo de cada nível? cn
- Qual é a altura da árvore? O caminho mais longo da raiz até uma folha é $n \to (2/3)n \to (2/3)^2 n \to \cdots \to 1$. Tendo em vista que $(2/3)^i n = 1$ quando $i = \log_{3/2} n$, a altura da árvore é $\log_{3/2} n$.
- É possível extrair um limite com estas informações?

O método da árvore de recursão

- Qual é o custo de cada nível? cn
- Qual é a altura da árvore? O caminho mais longo da raiz até uma folha é $n \to (2/3)n \to (2/3)^2 n \to \cdots \to 1$. Tendo em vista que $(2/3)^i n = 1$ quando $i = \log_{3/2} n$, a altura da árvore é $\log_{3/2} n$.
- É possível extrair um limite com estas informações? Intuitivamente, esperamos que a solução para recorrência seja no máximo o número de níveis vezes o custo de cada nível, ou $O(cn \log_{3/2} n) = O(n \lg n)$

O método da árvore de recursão

- Qual é o custo de cada nível? cn
- Qual é a altura da árvore? O caminho mais longo da raiz até uma folha é $n \to (2/3)n \to (2/3)^2 n \to \cdots \to 1$. Tendo em vista que $(2/3)^i n = 1$ quando $i = \log_{3/2} n$, a altura da árvore é $\log_{3/2} n$.
- É possível extrair um limite com estas informações? Intuitivamente, esperamos que a solução para recorrência seja no máximo o número de níveis vezes o custo de cada nível, ou $O(cn \log_{3/2} n) = O(n \lg n)$
- Existe algum problema com este limite?

O método da árvore de recursão

- Qual é o custo de cada nível? cn
- Qual é a altura da árvore? O caminho mais longo da raiz até uma folha é $n \to (2/3)n \to (2/3)^2 n \to \cdots \to 1$. Tendo em vista que $(2/3)^i n = 1$ quando $i = \log_{3/2} n$, a altura da árvore é $\log_{3/2} n$.
- É possível extrair um limite com estas informações? Intuitivamente, esperamos que a solução para recorrência seja no máximo o número de níveis vezes o custo de cada nível, ou $O(cn \log_{3/2} n) = O(n \lg n)$
- Existe algum problema com este limite? Faltou considerar os nós folhas!

O método da árvore de recursão

- Qual é o custo de cada nível? cn
- Qual é a altura da árvore? O caminho mais longo da raiz até uma folha é $n \to (2/3)n \to (2/3)^2 n \to \cdots \to 1$. Tendo em vista que $(2/3)^i n = 1$ quando $i = \log_{3/2} n$, a altura da árvore é $\log_{3/2} n$.
- É possível extrair um limite com estas informações? Intuitivamente, esperamos que a solução para recorrência seja no máximo o número de níveis vezes o custo de cada nível, ou $O(cn \log_{3/2} n) = O(n \lg n)$
- Existe algum problema com este limite? Faltou considerar os nós folhas!
- Discussão em sala. Veja a página 58.

O método da árvore de recursão

- Vamos mostrar (usando o método de substituição) que $T(n) \leq dn \lg n$, para alguma constante d

$$
\begin{aligned}
T(n) &\leq T(n/3) + T(2n/3) + cn \\
&\leq d(n/3)\lg(n/3) + d(2n/3)\lg(2n/3) + cn \\
&= d(n/3)\lg n - d(n/3)\lg 3 + d(2n/3)\lg n \\
&\qquad - d(2n/3)\lg(3/2) + cn \\
&= dn\lg n - d((n/3)\lg 3 + (2n/3)\lg(3/2)) + cn \\
&= dn\lg n - d((n/3)\lg 3 + (2n/3)\lg 3 - (2n/3)\lg 2) + cn \\
&= dn\lg n - dn(\lg 3 - 2/3) + cn \\
&\leq dn\lg n \qquad\qquad \text{deste que } d \geq c/(\lg 3 - (2/3))
\end{aligned}
$$

O método da árvore de recursão

- Vamos mostrar (usando o método de substituição) que $T(n) \leq dn \lg n$, para alguma constante d

$$
\begin{aligned}
T(n) &\leq T(n/3) + T(2n/3) + cn \\
&\leq d(n/3)\lg(n/3) + d(2n/3)\lg(2n/3) + cn \\
&= d(n/3)\lg n - d(n/3)\lg 3 + d(2n/3)\lg n \\
&\qquad - d(2n/3)\lg(3/2) + cn \\
&= dn \lg n - d((n/3)\lg 3 + (2n/3)\lg(3/2)) + cn \\
&= dn \lg n - d((n/3)\lg 3 + (2n/3)\lg 3 - (2n/3)\lg 2) + cn \\
&= dn \lg n - dn(\lg 3 - 2/3) + cn \\
&\leq dn \lg n \qquad\qquad \text{deste que } d \geq c/(\lg 3 - (2/3))
\end{aligned}
$$

- Conclusão: não foi necessário fazer um contabilidade precisa do custo para obtermos um limite válido.

Exercícios

- 4.2-1 Use uma árvore de recursão para determinar um bom limite superior assintótico na recorrência $T(n) = 3T(\lfloor n/2 \rfloor) + n$. Use o método de substituição para verificar sua resposta.
- 4.2-2 Demonstre que a solução para a recorrência $T(n) = T(n/3) + T(2n/3) + cn$, onde c é uma constante, é $\Omega(n \lg n)$, apelando para uma árvore de recursão.
- 4.2-3 Trace a árvore de recursão para $T(n) = 4T(\lfloor n/2 \rfloor) + cn$, onde c é uma constante, e forneça um limite assintótico restrito sobre sua solução. Verifique o limite pelo método de substituição.
- 4.2-4 Use uma árvore de recursão com o objetivo de fornecer uma solução assintoticamente restrita para a recorrência $T(n) = T(n-a) + T(a) + cn$, onde $a \geq 1$ e $c > 0$ são constantes.
- 4.2-5 Use uma árvore de recursão para fornecer uma solução assintoticamente restrita para a recorrência $T(n) = T(\alpha n) + T((1-\alpha)n) + cn$, onde α é uma constante no intervalo $0 < \alpha < 1$ e $c > 0$ também é uma constante.

O método mestre

O método mestre fornece um processo de “livro de receitas” para resolver recorrências da forma

$$T(n) = aT(n/b) + f(n) \text{ onde } a \geq 1 \text{ e } b > 1$$

Sendo que

- n é o tamanho do problema
- a é o número de subproblemas na recursão
- n/b é o tamanho de cada subproblema
- $f(n)$ é uma função assintoticamente positiva que representa o custo de dividir o problema e combinar os resultados

Teorema mestre

Sejam $a \geq 1$ e $b > 1$ constantes, seja $f(n)$ uma função e seja $T(n)$ definida sobre os inteiros não negativos pela recorrência

$$T(n) = aT(n/b) + f(n)$$

onde interpretamos n/b com significado de $\lceil n/b \rceil$ ou $\lfloor n/b \rfloor$.
Então, $T(n)$ pode ser limitado assintoticamente como a seguir:

Teorema mestre

Sejam $a \geq 1$ e $b > 1$ constantes, seja $f(n)$ uma função e seja $T(n)$ definida sobre os inteiros não negativos pela recorrência

$$T(n) = aT(n/b) + f(n)$$

onde interpretamos n/b com significado de $\lceil n/b \rceil$ ou $\lfloor n/b \rfloor$.
Então, $T(n)$ pode ser limitado assintoticamente como a seguir:

1. Se $f(n) = O(n^{\log_b a - \epsilon})$ para alguma constante $\epsilon > 0$, então $T(n) = \Theta(n^{\log_b a})$.

Teorema mestre

Sejam $a \geq 1$ e $b > 1$ constantes, seja $f(n)$ uma função e seja $T(n)$ definida sobre os inteiros não negativos pela recorrência

$$T(n) = aT(n/b) + f(n)$$

onde interpretamos n/b com significado de $\lceil n/b \rceil$ ou $\lfloor n/b \rfloor$.
Então, $T(n)$ pode ser limitado assintoticamente como a seguir:

1. Se $f(n) = O(n^{\log_b a - \epsilon})$ para alguma constante $\epsilon > 0$, então $T(n) = \Theta(n^{\log_b a})$.
2. Se $f(n) = \Theta(n^{\log_b a})$, então $T(n) = \Theta(n^{\log_b a} \lg n)$.

Teorema mestre

Sejam $a \geq 1$ e $b > 1$ constantes, seja $f(n)$ uma função e seja $T(n)$ definida sobre os inteiros não negativos pela recorrência

$$T(n) = aT(n/b) + f(n)$$

onde interpretamos n/b com significado de $\lceil n/b \rceil$ ou $\lfloor n/b \rfloor$.
Então, $T(n)$ pode ser limitado assintoticamente como a seguir:

1. Se $f(n) = O(n^{\log_b a - \epsilon})$ para alguma constante $\epsilon > 0$, então $T(n) = \Theta(n^{\log_b a})$.
2. Se $f(n) = \Theta(n^{\log_b a})$, então $T(n) = \Theta(n^{\log_b a} \lg n)$.
3. Se $f(n) = \Omega(n^{\log_b a + \epsilon})$ para alguma constante $\epsilon > 0$, e se $af(n/b) \leq cf(n)$ (condição de regularidade) para alguma constante $c < 1$ e para todo n suficientemente grande, então $T(n) = \Theta(f(n))$.

Entendendo o método mestre

- Nos três casos a função $f(n)$ é comparada com $n^{\log_b a}$
- Intuitivamente, a solução é determinada pela maior das duas funções

Entendendo o método mestre

- Nos três casos a função $f(n)$ é comparada com $n^{\log_b a}$
- Intuitivamente, a solução é determinada pela maior das duas funções
- No caso 1, $n^{\log_b a}$ é polinomialmente maior que $f(n)$

Entendendo o método mestre

- Nos três casos a função $f(n)$ é comparada com $n^{\log_b a}$
- Intuitivamente, a solução é determinada pela maior das duas funções
- No caso 1, $n^{\log_b a}$ é polinomialmente maior que $f(n)$
- No caso 2, as funções tem a mesma taxa de crescimento e o termo $\lg n$ aparece multiplicado a solução

Entendendo o método mestre

- Nos três casos a função $f(n)$ é comparada com $n^{\log_b a}$
- Intuitivamente, a solução é determinada pela maior das duas funções
- No caso 1, $n^{\log_b a}$ é polinomialmente maior que $f(n)$
- No caso 2, as funções tem a mesma taxa de crescimento e o termo $\lg n$ aparece multiplicado a solução
- No caso 3, a função $f(n)$ é polinomialmente maior que $n^{\log_b a}$ e a condição de regularidade $af(n/b) \leq cf(n)$ é válida

Entendendo o método mestre

- Nos três casos a função $f(n)$ é comparada com $n^{\log_b a}$
- Intuitivamente, a solução é determinada pela maior das duas funções
- No caso 1, $n^{\log_b a}$ é polinomialmente maior que $f(n)$
- No caso 2, as funções tem a mesma taxa de crescimento e o termo $\lg n$ aparece multiplicado a solução
- No caso 3, a função $f(n)$ é polinomialmente maior que $n^{\log_b a}$ e a condição de regularidade $af(n/b) \leq cf(n)$ é válida
- Existe uma lacuna entre os casos 1 e 2, e os casos 2 e 3 (a maior função não é polinomialmente maior ou, no caso 3, a condição de regularidade não é verdadeira)

O método mestre - exemplos

- $T(n) = 9T(n/3) + n$

O método mestre - exemplos

- $T(n) = 9T(n/3) + n$
  - $a = 9$
  - $b = 3$
  - $f(n) = n$
  - $n^{\log_b a} = n^{\log_3 9} = n^2$

O método mestre - exemplos

- $T(n) = 9T(n/3) + n$
  - $a = 9$
  - $b = 3$
  - $f(n) = n$
  - $n^{\log_b a} = n^{\log_3 9} = n^2$
  - Como $f(n) = O(n^{\log_3 9 - \epsilon})$, onde $\epsilon = 1$, aplicamos o caso 1 do teorema mestre e concluímos que a solução é

O método mestre - exemplos

- $T(n) = 9T(n/3) + n$
  - $a = 9$
  - $b = 3$
  - $f(n) = n$
  - $n^{\log_b a} = n^{\log_3 9} = n^2$
  - Como $f(n) = O(n^{\log_3 9 - \epsilon})$, onde $\epsilon = 1$, aplicamos o caso 1 do teorema mestre e concluímos que a solução é
  - $T(n) = \Theta(n^2)$

O método mestre - exemplos

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

O método mestre - exemplos

- $T(n) = T(2n/3) + 1$
  - $a = 1$
  - $b = 3/2$
  - $f(n) = 1$
  - $n^{\log_b a} = n^{\log_{3/2} 1} = n^0 = 1$

O método mestre - exemplos

- $T(n) = T(2n/3) + 1$
  - $a = 1$
  - $b = 3/2$
  - $f(n) = 1$
  - $n^{\log_b a} = n^{\log_{3/2} 1} = n^0 = 1$
  - Como $f(n) = \Theta(n^{\log_{3/2} 1})$, aplicamos o caso 2 do teorema mestre e concluímos que a solução é

O método mestre - exemplos

- $T(n) = T(2n/3) + 1$
  - $a = 1$
  - $b = 3/2$
  - $f(n) = 1$
  - $n^{\log_b a} = n^{\log_{3/2} 1} = n^0 = 1$
  - Como $f(n) = \Theta(n^{\log_{3/2} 1})$, aplicamos o caso 2 do teorema mestre e concluímos que a solução é
  - $T(n) = \Theta(\lg n)$

O método mestre - exemplos

- $T(n) = 3T(n/4) + n \lg n$

O método mestre - exemplos

- $T(n) = 3T(n/4) + n \lg n$
  - $a = 3$
  - $b = 4$
  - $f(n) = n \lg n$
  - $n^{\log_b a} = n^{\log_4 3} = O(n^{0{,}793})$

O método mestre - exemplos

- $T(n) = 3T(n/4) + n \lg n$
  - $a = 3$
  - $b = 4$
  - $f(n) = n \lg n$
  - $n^{\log_b a} = n^{\log_4 3} = O(n^{0,793})$
  - Como $f(n) = \Omega(n^{\log_4 3+\epsilon})$, onde $\epsilon \approx 0,2$, podemos aplicar o caso 3, se a condição de regularidade for verdadeira

O método mestre - exemplos

- $T(n) = 3T(n/4) + n \lg n$
  - $a = 3$
  - $b = 4$
  - $f(n) = n \lg n$
  - $n^{\log_b a} = n^{\log_4 3} = O(n^{0,793})$
  - Como $f(n) = \Omega(n^{\log_4 3+\epsilon})$, onde $\epsilon \approx 0,2$, podemos aplicar o caso 3, se a condição de regularidade for verdadeira
  - Para n suficientemente grande, $af(n/b) = 3(n/4)\lg(n/4) \leq (3/4)n \lg n = cf(n)$ para $c = 3/4$.

O método mestre - exemplos

- $T(n) = 3T(n/4) + n \lg n$
  - $a = 3$
  - $b = 4$
  - $f(n) = n \lg n$
  - $n^{\log_b a} = n^{\log_4 3} = O(n^{0,793})$
  - Como $f(n) = \Omega(n^{\log_4 3+\epsilon})$, onde $\epsilon \approx 0,2$, podemos aplicar o caso 3, se a condição de regularidade for verdadeira
  - Para n suficientemente grande, $af(n/b) = 3(n/4)\lg(n/4) \leq (3/4)n \lg n = cf(n)$ para $c = 3/4$.
  - Portanto, de acordo com o caso 3, a solução é

O método mestre - exemplos

- $T(n) = 3T(n/4) + n \lg n$
  - $a = 3$
  - $b = 4$
  - $f(n) = n \lg n$
  - $n^{\log_b a} = n^{\log_4 3} = O(n^{0,793})$
  - Como $f(n) = \Omega(n^{\log_4 3+\epsilon})$, onde $\epsilon \approx 0,2$, podemos aplicar o caso 3, se a condição de regularidade for verdadeira
  - Para n suficientemente grande, $af(n/b) = 3(n/4)\lg(n/4) \leq (3/4)n \lg n = cf(n)$ para $c = 3/4$.
  - Portanto, de acordo com o caso 3, a solução é
  - $T(n) = \Theta(n \lg n)$

O método mestre - exemplos

- $T(n) = 2T(n/2) + n \lg n$

O método mestre - exemplos

- $T(n) = 2T(n/2) + n \lg n$
  - $a = 2$
  - $b = 2$
  - $f(n) = n \lg n$
  - $n^{\log_b a} = n^{\log_2 2} = n$

O método mestre - exemplos

- $T(n) = 2T(n/2) + n \lg n$
  - $a = 2$
  - $b = 2$
  - $f(n) = n \lg n$
  - $n^{\log_b a} = n^{\log_2 2} = n$
  - Parece que o caso 3 pode ser aplicado pois $f(n)$ é assintoticamente maior que $n^{\log_b a} = n^{\log_2 2} = n$

O método mestre - exemplos

- $T(n) = 2T(n/2) + n \lg n$
  - $a = 2$
  - $b = 2$
  - $f(n) = n \lg n$
  - $n^{\log_b a} = n^{\log_2 2} = n$
  - Parece que o caso 3 pode ser aplicado pois $f(n)$ é assintoticamente maior que $n^{\log_b a} = n^{\log_2 2} = n$
  - No entanto, não é polinomialmente maior

O método mestre - exemplos

- $T(n) = 2T(n/2) + n \lg n$
  - $a = 2$
  - $b = 2$
  - $f(n) = n \lg n$
  - $n^{\log_b a} = n^{\log_2 2} = n$
  - Parece que o caso 3 pode ser aplicado pois $f(n)$ é assintoticamente maior que $n^{\log_b a} = n^{\log_2 2} = n$
  - No entanto, não é polinomialmente maior
  - A razão $f(n)/n^{\log_b a} = (n \lg n)/n = \lg n$ é assintoticamente menor que n^{ϵ} para qualquer constante positiva ϵ

O método mestre - exemplos

- $T(n) = 2T(n/2) + n \lg n$
  - $a = 2$
  - $b = 2$
  - $f(n) = n \lg n$
  - $n^{\log_b a} = n^{\log_2 2} = n$
  - Parece que o caso 3 pode ser aplicado pois $f(n)$ é assintoticamente maior que $n^{\log_b a} = n^{\log_2 2} = n$
  - No entanto, não é polinomialmente maior
  - A razão $f(n)/n^{\log_b a} = (n \lg n)/n = \lg n$ é assintoticamente menor que n^{ϵ} para qualquer constante positiva ϵ
  - Ou seja, esta situação está na lacuna entre os casos 2 e 3

Exercícios

4.3-1 Use o método mestre para fornecer limites assintóticos restritos para as recorrências a seguir.

a $T(n) = 4T(n/2) + n$

b $T(n) = 4T(n/2) + n^2$

c $T(n) = 4T(n/2) + n^3$

4.3-2 O tempo de execução de um algoritmo A é descrito pela recorrência $T(n) = 7T(n/2) + n^2$. Um algoritmo concorrente A' tem um tempo de execução $T'(n) = aT'(n/4) + n^2$. Qual o maior valor inteiro para a tal que A' seja assintoticamente mais rápido de A?

4.3-3 Use o método mestre para mostrar que a solução para a recorrência de pesquisa binária $T(n) = T(n/2) + \Theta(1)$ é $T(n) = \Theta(\lg n)$. (Veja no exercício 2.3-5 uma descrição da pesquisa binária.)

4.3-4 O método mestre pode ser aplicado à recorrência $T(n) = 4T(n/2) + n^2 \lg n$? Por que ou por que não? Forneça um limite assintótico para essa recorrência.

Exercícios

- Exercícios do capítulo 4: 4.1, 4.4

Referências

- Thomas H. Cormen et al. Introdução a Algoritmos. 2ª edição em português. Capítulo 4.