Projeto de algoritmos: Divisão e Conquista
ACH2002 - Introdução à Ciência da Computação II

Delano M. Beder

Escola de Artes, Ciências e Humanidades (EACH)
Universidade de São Paulo
dbeder@usp.br

09/2008

Material baseado em slides do professores Cid de Souza e Cândida Nunes

Divisão e Conquista

- Construção incremental
  - Consiste em, inicialmente, resolver o problema para um sub-conjunto dos elementos da entrada e, então adicionar os demais elementos um a um. Em muitos casos, se os elementos forem adicionados em uma ordem ruim, o algoritmo não será eficiente.
- Divisão e Conquista
  - Para resolver um problema eles chamam a si mesmos para resolver *subproblemas* menores e combinam as *subrespostas*.
  - É mais um paradigma de projeto de algoritmos baseado no princípio da indução.
  - Informalmente, podemos dizer que o paradigma *incremental* representa o projeto de algoritmos por **indução fraca**, enquanto o paradigma de *divisão e conquista* representa o projeto por **indução forte**.

Divisão e Conquista

- **Dividir** o problema em determinado número de subproblemas.
- **Conquistar** os subproblemas, resolvendo-os recursivamente.
  - Se o tamanho do subproblema for pequeno o bastante, então a solução é direta.
- **Combinar** as soluções fornecidas pelos subproblemas, a fim de produzir a solução para o problema original.
- A busca binária recursiva utiliza essa técnica?
  - Sim. Mas a etapa de *combinar* tem custo zero, pois o resultado do subproblema já é o resultado do problema maior.

Exemplo - Exponenciação - Solução 1

Problema:
Calcular a^n, para todo real a e inteiro $n \geq 0$.

Primeira solução por indução fraca:

- **Caso base:** $n = 0$; $a^0 = 1$.
- **Hipótese de indução:** Suponha que, para qualquer inteiro $n > 0$ e real a, sei calcular a^{n-1}.
- **Passo da indução:** Queremos provar que conseguimos calcular a^n, para $n > 0$. Por hipótese de indução, sei calcular a^{n-1}. Então, calculo a^n multiplicando a^{n-1} por a.

Solução 1 - Algoritmo

```
/* Entrada: A base a e o expoente n.
   Saída: O valor de a elevado a n. */
double exponenciacao(double a, int n) {
  double an;
  if (n == 0) {
     return 1; /* caso base */
  } else {
     return a * exponenciacao(a, n - 1);
  }
}
```


Solução 1 - Complexidade

Seja $T(n)$ o número de operações executadas pelo algoritmo para calcular a^n.

Então a relação de recorrência deste algoritmo é:

$$T(n) = \begin{cases} c1 & \text{se} \quad n = 0 \\ T(n-1) + c2 & \text{se} \quad n > 0 \end{cases}$$

onde $c1$ e $c2$ representam, respectivamente, o tempo (constante) executado na atribuição da base e multiplicação do passo.

Nesse caso, não é difícil ver que

$$T(n) = c1 + \sum_{i=1}^{n} c2 = c1 + nc2 = \Theta(n)$$

Exemplo - Exponenciação - Solução 2

Vamos agora projetar um algoritmo para o problema usando indução forte de forma a obter um algoritmo de divisão e conquista.

Segunda solução por indução forte:

- **Caso base:** $n = 0$; $a^0 = 1$.
- **Hipótese de indução:** Suponha que, para qualquer inteiro $n > 0$ e real a, sei calcular a^k, para todo $k < n$.
- **Passo da indução:** Queremos provar que conseguimos calcular a^n, para $n > 0$. Por hipótese de indução, sei calcular $a^{\lfloor \frac{n}{2} \rfloor}$. Então, calculo a^n da seguinte forma:

$$a^n = \begin{cases} {a^{\lfloor \frac{n}{2} \rfloor}}^2 & \text{se} \quad n \mod 2 = 0 \\ a \times {a^{\lfloor \frac{n}{2} \rfloor}}^2 & \text{se} \quad n \mod 2 = 1 \end{cases}$$

Solução 2 - Algoritmo

```
/* Entrada: A base a e o expoente n.
   Saída: O valor de a elevado a n. */
double exponenciacao(double a, int n) {
  double an;
  if (n == 0) {
     return 1; /* caso base */
  } else {
     double aux = exponenciacao(a, n div 2);
     an = aux * aux;
     if (n mod 2 == 1) {
       an = an * a;
     }
     return an;
  }
}
```

Solução 2 - Complexidade

Seja $T(n)$ o número de operações executadas pelo algoritmo para calcular a^n.

Então a relação de recorrência deste algoritmo é:

$$T(n) = \begin{cases} c1 & \text{se} \quad n = 0 \\ T(\lfloor \frac{n}{2} \rfloor) + c2 & \text{se} \quad n > 0 \end{cases}$$

onde $c1$ e $c2$ representam, respectivamente, o tempo (constante) executado na atribuição da base e multiplicação do passo.

Nesse caso, não é difícil ver que $T(n) \in \Theta(\log \text{n})$.

Exercício - Máximo e Mínimo

Problema:
Dado um conjunto S de $n \geq 2$ números reais, determinar o maior e o menor elemento de S.

- Um algoritmo incremental para esse problema faz $2n - 3$ comparações: fazemos 1 comparação no caso base e 2 no passo.
- *Será que um algoritmo de divisão e conquista seria melhor ?*

Exercício:
Projete um algoritmo de divisão e conquista para esse problema que faz um número menor de comparações.

Resolução de recorrências de divisão e conquista

- A complexidade de tempo de algoritmos *divisão e conquista* para um entrada de tamanho n é:
  - T(n) = Dividir(n) + Conquistar(n) + Combinar(n).
- Para entradas pequenas, isto é, para $n \leq c$, podemos assumir que $T(n) = O(1)$.
- Vamos supor que o problema seja dividido em a subproblemas, cada um com $1/b$ do tamanho original.
- Se levamos $D(n)$ para dividir o problema em subproblemas e $C(n)$ para combinar as soluções dados aos subproblemas, então tem-se a recorrência $T(n)$ tal que:

$$T(n) = \begin{cases} O(1) & \text{se} \quad n \leq c \\ aT(n/b) + D(n) + C(n) & \text{caso contrário} \end{cases}$$

Resolução de recorrências de divisão e conquista

- Expressão geral de recorrência de um algoritmo de divisão e conquista:

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

  - Onde a representa o número e n/b o tamanho dos subproblemas obtidos na divisão, e
  - $f(n)$ é a função que dá a complexidade das etapas de divisão e conquista.

Resolução de recorrências de divisão e conquista

Teorema Mestre (CLRS)

Sejam $a \geq 1$ e $b \geq 2$ constantes, seja $f(n)$ uma função e seja $T(n)$ definida para os inteiros não-negativos pela relação de recorrência

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

Então $T(n)$ pode ser limitada assintoticamente da seguinte maneira:

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

Teorema Mestre - Exemplo de Recorrências

Exemplo 1

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

Para esta recorrência, temos que $a = 9$, $b = 3$ e $f(n) = n$.

Desta forma $n^{\log_b a} = n^{\log_3 9} = n^2$.

Desde que $f(n) \in O(n^{\log_3 9 - \epsilon})$, onde $\epsilon = 1$, nós podemos aplicar o caso 1 do Teorema Mestre.

$T(n) = \Theta(n^{\log_b a}) = \Theta(n^{\log_3 9}) = \Theta(n^2)$.

Logo, a solução é $T(n) \in \Theta(n^2)$.

Teorema Mestre - Exemplo de Recorrências

Exemplo 2

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

Para esta recorrência, temos que $a = 1$, $b = 3/2$ e $f(n) = 1$.

Desta forma $n^{\log_b a} = n^{\log_{3/2} 1} = n^0 = 1$.

Desde que $f(n) \in \Theta(n^{\log_{3/2} 1}) = \Theta(1)$, nós podemos aplicar o caso 2 do Teorema Mestre.

$T(n) = \Theta(n^{\log_b a} \log n) = \Theta(n^{\log_{3/2} 1} \log n) = \Theta(\log n)$.

Logo, a solução é $T(n) \in \Theta(\log n)$.

Teorema Mestre - Exemplo de Recorrências

Exemplo 3

$$T(n) = 3T(n/4) + n \log n$$

Para esta recorrência, temos que $a = 3$, $b = 4$ e $f(n) = n \log n$.

Desta forma $n^{\log_b a} = n^{\log_4 3} = n^{0.793}$.

Desde que $f(n) \in \Omega(n^{\log_4 3 + \epsilon})$, onde $\epsilon \approx 0.2$, nós podemos aplicar o caso 3 do Teorema Mestre, se conseguirmos mostrar que $af(n/b) \leq cf(n)$, para alguma constante $c < 1$ e para n suficientemente grande.

$af(n/b) = 3(n/4) \log(n/4) \leq (3/4) n \log n = cf(n)$ para $c = 3/4$.

Logo, a solução é $T(n) \in \Theta(f(n)) = \Theta(n \log n)$.

Teorema Mestre - Mais exemplos de recorrências

Exemplos onde o Teorema Mestre se aplica

- (Caso 1) $T(n) = 4T(n/2) + n \log n$, $T(1) = 1$.
- (Caso 2) $T(n) = 2T(n/2) + n$, $T(1) = 1$.
- (Caso 3) $T(n) = T(n/2) + n \log n$, $T(1) = 1$.

Exemplos onde o Teorema Mestre não se aplica

- $T(n) = T(n-1) + n \log n$, $T(1) = 1$.
- $T(n) = T(n-a) + T(a) + n$, $T(b) = 1$.
  (para inteiros $a \geq 1$, $b \leq a$, a e b inteiros)
- $T(n) = T(\alpha n) + T((1-\alpha)n) + n$, $T(1) = 1$.
  (para $0 < \alpha < 1$)
- $T(n) = T(n-1) + \log n$, $T(1) = 1$.
- $T(n) = 2T(\frac{n}{2}) + n \log n$, $T(1) = 1$.

Exercícios

Use o Teorema Mestre para encontrar solução para as seguintes recorrências:

1. $T(n) = 4T(n/2) + n \log n$.
2. $T(n) = 2T(n/2) + n$.
3. $T(n) = T(n/2) + n \log n$.
4. $T(n) = T(n/2) + \Theta(1)$.
   [Recorrência da exponenciacão - divisão e conquista]
5. $T(n) = 4T(n/2) + n$.
6. $T(n) = 4T(n/2) + n^2$.
7. $T(n) = 4T(n/2) + n^3$.