Recursividade e Equações de Recorrência

Prof. Luiz Chaimowicz

Algoritmos Recursivos

- Um procedimento recursivo é aquele que chama a si mesmo direta ou indiretamente
- Normalmente possui duas partes
  - Uma ou mais chamadas recursivas
  - Condição de Parada
- Vantagens:
  - implementação mais simples, normalmente direta a partir da definição do problema
- Desvantagens:
  - Custo das chamadas recursivas / pilha de ativação

Exemplo Clássico 1

- Fatorial:

$$\begin{cases} n! = n*(n-1)! & p/\ n>0 \\ 0! = 1 & p/\ n=0 \end{cases}$$

- Em C

```
int Fat (int n) {
    if (n<=0)
        return 1;
    else
        return n * Fat(n-1);
}
```

Custo:
Tempo: $\Theta(n)$
Espaço: $\Theta(n)$ ☹

Implementação iterativa gasta $\Theta(1)$ de espaço...

Exemplo Clássico 2

De quando **não** se deve usar recursividade

- **Série de Fibonnaci:**

$$\begin{cases} F_n = F_{n-1} + F_{n-2} & n > 2, \\ F_0 = F_1 = 1 \end{cases}$$

  - 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89...

```
Fib(int n) {
  if (n<2)
    return 1;
  else
    return Fib(n-1)+Fib(n-2);
}
```

Custo: $\Theta(\phi^n)$!?!

Implementação iterativa gasta $\Theta(n)$ de tempo e $\Theta(1)$ de espaço...

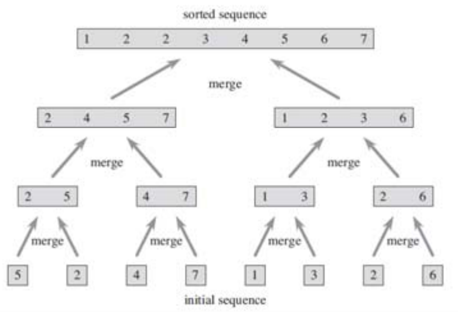
n	20	30	50	100
Recursiva	1 seg	2 min	21 dias	10^9 anos
Iterativa	1/3 mseg	1/2 mseg	3/4 mseg	1,5 mseg

Algoritmos *Dividir para Conquistar*

- Algoritmos que dividem o problema em vários subproblemas menores que são resolvidos recursivamente (detalhes com o Prof. Wagner)
- Normalmente possuem um custo equivalente aos algoritmos iterativos, que requerem o uso de estruturas auxiliares (pilhas, etc)
- Exemplo: **Mergesort:**

```
MERGE-SORT(A, p, r)
  if p < r
     q = ⌊(p + r)/2⌋
     MERGE-SORT(A, p, q)
     MERGE-SORT(A, q + 1, r)
     MERGE(A, p, q, r)
```

MergeSort



Análise de Algoritmos Recursivos

- A análise de algoritmos recursivos consiste na **representação** do seu custo através de uma **equação de recorrência** e na **resolução** desta equação
- Equações de Recorrência
  - Descrevem o custo de resolver um problema de tamanho *n* em termos da solução de problemas menores
  - Exigem um ferramental matemático em sua resolução

Análise de Algoritmos Recursivos

- Representação através da recorrência
  - Requer um entendimento dos custos envolvidos na chamada recursiva e demais operações
- Resolução da equação de recorrência
  - Requer o uso de diferentes métodos:
    - "Expansão de termos" (ou "método iterativo")
    - Teorema Mestre
    - Método da Substituição
    - Método da Árvore de Recursão

Análise do Fatorial

```
int Fat (int n) {
  if (n<=0)
    return 1;
  else
    return n * Fat(n-1);
}
```

$$T(n) = \begin{cases} T(n-1) + c & \text{p/ } n>0 \\ d & \text{p/ } n \leq 0 \end{cases}$$

Expansão de termos:

T(n) = T(n-1) + c
T(n-1) = T(n-2) + c
…
T(2) = T(1) + c
T(1) = T(0) + c
T(0) = d

T(n) = c+c+…+c+d
T(n) = nc + d
T(n) = Θ(n)

Análise do Mergesort

```
MERGE-SORT(A, p, r)
  if p < r
      q = ⌊(p + r)/2⌋
      MERGE-SORT(A, p, q)
      MERGE-SORT(A, q + 1, r)
      MERGE(A, p, q, r)
```

$$\begin{cases} T(n) = 2*T(n/2) + n & \text{p/ } n>1 \\ T(n) = 1 & \text{p/ } n=1 \end{cases}$$

Alguns Detalhes:

- N potência de 2
- Algumas vezes caso base é omitido

Resolvendo por Expansão

$T(n) = 2T(n/2) + n$
$T(1) = 1$
Expandindo a equação:
$T(n) = 2T(n/2) + n$
$2T(n/2) = 4T(n/4) + n$
$4T(n/4) = 8T(n/8) + n$
$\vdots$
$2^{i-1}T(n/2^{i-1}) =$
$= 2^{i}T(n/2^{i}) + n$

Substituindo os termos:
$T(n) = 2^{i}T(n/2^{i}) + i.n$
Caso base:
$T(n/2^{i}) \rightarrow T(1)$
$n/2^{i} = 1 \rightarrow i = \log_2 n$
Logo:
$T(n) = 2^{i}T(n/2^{i}) + i.n$
$= 2^{\log_2 n}.1 + (\log_2 n).n$
$= n + n.\log_2 n = \Theta(n.\log_2 n)$

Teorema Mestre

Teorema Mestre: Sejam $a \geq 1$ e $b > 1$ constantes, $f(n)$ uma função assintoticamente positiva e $T(n)$ uma medida de complexidade definida sobre os inteiros. A solução da equação de recorrência:

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

para b uma potência de n é:

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

Teorema Mestre

Intuição: a função ***f(n)*** é comparada com $n^{\log_b a}$ e a maior das duas funções é a solução da recorrência. No caso das duas funções serem equivalentes, a solução é $n^{\log_b a}$ multiplicada por um fator logarítmico

Detalhes: nos casos 1 e 3, a função *f(n)* deve ser polinomialmente menor / maior do que $n^{\log_b a}$. Além disso, a função deve satisfazer uma condição de regularidade.

Teorema Mestre - Exemplos

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

onde $a = 9$, $b = 3$, $f(n) = n$ e $n^{\log_b a} = n^{\log_3 9} = \Theta(n^2)$.

O caso 1 se aplica porque $f(n) = O(n^{\log_b a - \epsilon}) = O(n)$, onde $\epsilon = 1$, e a solução é $T(n) = \Theta(n^2)$.

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

onde $a = 3$, $b = 4$, $f(n) = n \log n$ e $n^{\log_b a} = n^{\log_3 4} = n^{0.793}$.

O caso 3 se aplica porque $f(n) = \Omega(n^{\log_3 4 + \epsilon})$, onde $\epsilon \approx 0.207$ e $af(n/b) = 3(n/4)\log(n/4) \leq cf(n) = (3/4)n \log n$, para $c = 3/4$ e n suficientemente grande.

Teorema Mestre - Exemplos

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

onde $a = 2$, $b = 2$, $f(n) = n - 1$ e $n^{\log_b a} = n^{\log_2 2} = \Theta(n)$.

O caso 2 se aplica porque $f(n) = \Theta(n^{\log_b a}) = \Theta(n)$, e a solução é $T(n) = \Theta(n \log n)$.

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

onde $a = 3$, $b = 3$, $f(n) = n \log n$ e $n^{\log_b a} = n^{\log_3 3} = n$.

O caso 3 não se aplica porque, embora $f(n) = n \log n$ seja assintoticamente maior do que $n^{\log_b a} = n$, a função $f(n)$ não é polinomialmente maior: a razão $f(n)/n^{\log_b a} = (n \log n)/n = \log n$ é assintoticamente menor do que n^ϵ para qualquer constante ϵ positiva.

Método da Substituição

- Dois passos:
  - Estimar ("chutar") uma solução
  - Usar indução matemática para mostrar que o "chute" é válido
- Pode ser usada para estabelecer limites superiores e inferiores para a recorência
- Possíveis problemas
  - Estimação da solução
  - Cuidados na aplicação da indução

Método da Substituição - Exemplo

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

Chute: $T(n) = O(n \lg n)$

Devemos provar portanto que: $T(n) \leq c.n \lg n$

Vamos começar com o passo indutivo, mostrando que: se a a solução é verdadeira para todo *m* < *n* especialmente para *n / 2* ela é válida para *T(n).* Isso é feito **substituindo-se** *T(n/2)* na equação de recorrência:

$$T(n/2) \leq cn/2 \lg(n/2)$$
$$T(n) \leq 2(cn/2 \lg(n/2)) + n$$

$$\begin{aligned} T(n) &\leq cn \lg(n/2) + n \\ &= cn \lg n - cn \lg 2 + n \\ &= cn \lg n - cn + n \\ T(n) &\leq cn \lg n \text{ para } c \geq 1 \end{aligned}$$

Método da Substituição - Exemplo

Agora é necessário mostrar que a solução é valida para o caso base *T(1) = 1.*

O problema é que **ela não é válida!**

$T(n) \leq c.n \lg n \rightarrow T(1) \leq c.1 \lg 1 = 0$

Para resolver esse problema, basta lembrar que a notação assintótica exige que: $T(n) \leq c.n \lg n \ \forall n \geq n_o$

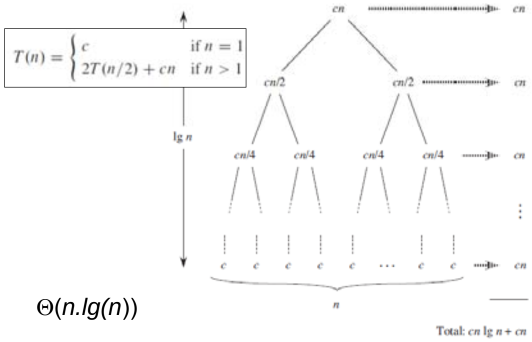
Portanto, basta escolher outro caso base, como *T(2)*. Pela equação de recorrência, se *T(1) = 1* então *T(2) = 4.*

Logo podemos mostrar que $T(2) \leq c.2 \lg 2$ é válida para c ≥ 2. Portando, a solução *T(n) = O(n.lg(n)*) funciona

Método da Árvore de Recursão

- Aplicando-se a recursão, monta-se uma árvore onde cada nó representa o custo de um subproblema.
- Expandindo-se a árvore, pode-se obter a soma de todos os custos de cada nível e depois do problema como um todo.
- De certa forma funciona como uma visualização do método da “expansão de termos” apresentado inicialmente.

Árvore de Recursão - Exemplo



$\Theta(n.lg(n))$

Comentários Gerais sobre os Métodos

- Segundo Cormen, os métodos da substituição e do Teorema Mestre são preferíveis.
  - O método iterativo deve ser usado com cuidado para evitar erros na expansão de simplificação de termos
  - O método da árvore de recursão não é um método matemático formal, portanto deve ser usado em conjunto com outros métodos

“Para Casa”

- Ler capítulo 4 do Cormen (ênfase nos métodos de resolução de equações de recorrência). Leia também o capítulo 2 do Nívio com a mesma ênfase.

- Resolver os seguintes exercícios:

  Escolha algumas (todas?) equações de recorrência do livro e as resolva (exercícios: 4.5-1, 4-1, 4-3)