Recorrência

Manoel Ribeiro Filho

Para analisar algoritmos recursivos é necessário resolver recorrências. Uma recorrência é uma fórmula que define uma função, digamos T, em termos dela mesma. Mais precisamente, a recorrência define T(n) em termos de T(n-1), T(n-2), T(n-3), etc.

Uma solução de uma recorrência é uma fórmula que exprime T(n) em termos de n apenas.

Considere o algoritmo Soma, que calcula a soma dos elementos de um vetor, na sua versão recursiva, que já vimos e mostramos novamente, na próxima transparência.

```
// O  algoritmo  Soma devolve a soma dos  elementos de  V[0..n-1].
Algoritmo Soma(V,n)
1         Se n== 0 retorne 0
2         Se não
3                   S =Soma(V,n-1)
4                   S=  S + V[n-1]
5                   retorne S
```

Digamos que T(n) é a função que dá o consumo de tempo no pior caso. Se a execução de qualquer das linhas do pseudocódigo consome uma unidade de tempo,
então T(0) = 1 (tempo de consumo da linha 1)
e T(n) satisfaz a recorrência T(n) = T(n-1) + 3
(O termo T(n-1) corresponde à linha 3 e o termo 3 é o consumo de tempo das linhas 1, 4 e 5.).
Teríamos, agora, que resolver a recorrência T(n) = T(n-1) + 3. Resolver recorrências não é uma tarefa fácil, em nosso curso vamos apresentar uma tabela com diversas soluções de recorrência, e nossos problemas sempre caíram em uma das soluções apresentadas. Para a recorrência citada acima a solução é
T(n) = 2 + 3n, logo para o pior caso o tempo de execução é $\Theta(n)$

Algumas soluções de equações de recorrências

$T(n) = T(n-1) + c$ pior caso $\Theta(n)$

$T(n) = 2T(n/2) + n$ pior caso $\Theta(n \lg n)$

$T(n) = T(n-1) + n$ pior caso $\Theta(n^2)$

$T(n) = 2T(n-1) + 1$ pior caso $\Theta(2^n)$

$T(n) = T(n/2) + 1$ pior caso $\Theta(\lg n)$

Para o fatorial recursivo

Algoritmo Fat(n)

1 Se n == 0 retorne 1

2 Se não retorne n*fat(n-1)

Claramente T(n) = T(n-1) +1, logo pior caso $\Theta(n)$

Análise do tempo de execução do algoritmo Merge Sort

Algoritmo Merge-Sort(A,p,r)
1 Se p<r
2 Então q <- (p+r)/2
3 Merge-Sorte(A,p,q)
4 Merge-Sorte(A,q+1,r)
5 Intercala(A,p,q,r)

O tempo de processamento da linha 1 é 1, ou seja T(1)=1, caso base
Agora veremos o tempo de processamento de T(n), caso recursivo. Já vimos que o tempo de execução da função intercala é n, ou seja o tempo de processamento da linha 5 é n. Para as linhas 3 e 4, a função Merge-Sort, que inicialmente tem tamanho n, é chamada recursivamente, sempre com tamanho n/2, logo
T(n) = T(n/2) + T(n/2) + n
T(n) = 2T(n/2) +n, cuja solução pela nossa tabela, para o tempo de execução de pior caso é

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