MC102 - Algoritmos e Progração de Computador

Prof. Alexandre Xavier Falcão

18° Aula: Recursão

1 Recursão

A solução de um problema é dita **recursiva** quando ela é escrita em função de si própria para instâncias menores do problema. Considere, por exemplo, o problema de calcular a soma dos números inteiros no intervalo $[m, n]$, onde $m, n \in \mathbb{Z}$ e $m \leq n$.

A solução **iterativa** é $m + (m+1) + (m+2) + \ldots + n$, que pode ser facilmente implementada com a função abaixo:

```
int Soma(int m, int n)
{
  int soma=0, i;

  for (i=m; i <= n; i++)
    soma += i;

  return(soma);
}
```

A solução recursiva está diretamente ligada ao conceito de indução matemática. A **indução fraca** usa como hipótese que a solução de um problema de tamanho t pode ser obtida a partir de sua solução de tamanho $t-1$. Por exemplo, se soubermos o resultado da soma de m até $n-1$, basta somar n a este resultado: $Soma(m, n) = Soma(m, n-1) + n$. Neste caso, dizemos que a recursão é **decrescente**. Podemos também obter o mesmo resultado somando m com a soma de $m+1$ até n: $Soma(m, n) = m + Soma(m+1, n)$. Neste caso, dizemos que a recursão é **crescente**. A solução recursiva, portanto, pode ser implementada com funções do tipo:

```
tipo função(<parâmetros>)
{
  <variáveis locais>

  if (<condição de parada>)
  {
    <comandos finais>
    return (valor);
  } else {
    <comandos iniciais>
    <chamada recursiva>
```

```
    <comandos finais>
    return (valor);
  }
}
```

No caso do exemplo acima temos:

- Solução crescente

```
int Soma(int m, int n)
{
  if (m==n){
    return(n);
  } else {
    return(m+Soma(m+1,n));
  }
}
```

- Solução decrescente

```
int Soma(int m, int n)
{
  if (m==n){
    return(m);
  } else {
    return(Soma(m,n-1)+n);
  }
}
```

Observe que nestes exemplos não precisamos de variáveis locais, nem de comandos iniciais e finais, portanto podemos colocar a chamada recursiva no comando de retorno. As **árvores de recursão** mostradas na Figura 1 ajudam a visualizar o comportamento dessas funções.

A **indução forte** usa outra hipótese: a solução de um problema de tamanho t depende de suas soluções de tamanhos t', para todo $t' < t$. Esta estratégia também é denominada **divisão e conquista**. Por exemplo, podemos dizer que $Soma(m, n) = Soma(m, \frac{m+n}{2}) + Soma(\frac{m+n}{2} + 1, n)$. Ou seja, o problema é dividido em subproblemas similares (divisão), os subproblemas são resolvidos recursivamente (i.e. quando o problema chega a seu tamanho mínimo, ele é resolvido de forma direta), e as soluções dos subproblemas são combinadas para gerar a solução do problema de tamanho maior (conquista). Isto requer, porém, ao menos duas chamadas recursivas:

```
tipo função(<parâmetros>)
{
  <variáveis locais>

  if (<condição de parada>)
  {
    <comandos finais>
```

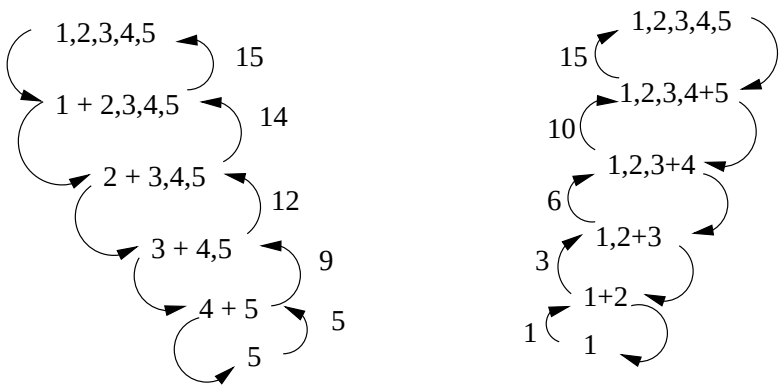

Figura 1: Árvores de recursão para $m = 1$ e $n = 5$.

```
    return (valor);
  } else {
    <comandos iniciais>
    <primeira chamada recursiva>
    <segunda  chamada recursiva>
    <comandos finais>
    return (valor);
  }
}
```

No caso do exemplo acima temos:

```
int Soma(int m, int n)
{
  if (m==n){
    return(m);
  } else {
    return(Soma(m,(m+n)/2)+Soma((m+n)/2+1,n));
  }
}
```

Mais uma vez, pelas mesmas razões acima, as chamadas recursivas podem ser feitas no comando de retorno. A árvore de recursão é apresentada na Figura 2.

Observe que em todos os exemplos, as chamadas recursivas são para a própria função. Dizemos então que a **recursão é direta**. Em alguns problemas (e.g. análise sintática do compilador), porém, uma função A pode chamar uma função B que chama C, que chama ..., que chama A. Neste caso dizemos que a **recursão é indireta**.

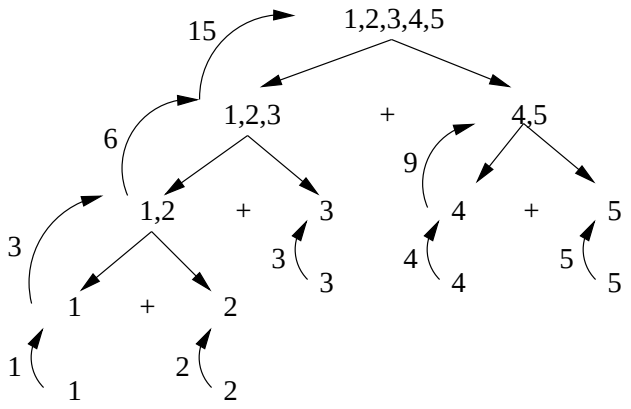

Figura 2: Árvore de recursão para $m = 1$ e $n = 5$.

2 Exercícios

1. Escreva uma função recursiva para calcular o fatorial de um número inteiro n. Note que o fatorial de n pode ser definido de forma recursiva como:

$$fat(n) \begin{cases} 1 & \text{se } n = 0 \\ n \times fat(n-1) & \text{se } n > 0 \end{cases} \quad (1)$$

2. Os termos de uma seqüência de Fibonacci (e.g. $\langle 0, 1, 1, 2, 3, 5, 8, \ldots \rangle$ são definidos de forma recursiva como:

$$fib(n) = \begin{cases} n-1 & \text{se } 1 \leq n \leq 2 \\ fib(n-1) + fib(n-2) & \text{se } n > 2 \end{cases} \quad (2)$$

   Escreva uma função recursiva para retornar o n-ésimo termo desta seqüência.