Funções recursivas

Marco A L Barbosa

malbarbo.pro.br

Departamento de Informática

Universidade Estadual de Maringá

Introdução

- Uma técnica poderosa na resolução de problemas é a decomposição
- Por exemplo, para resolvermos o problema de encontrar as soluções de uma equação de segundo grau decompomos o problema em problemas menores
    - Calcular o delta
    - Calcular as raízes (se houverem)
- Veremos agora uma forma particular de decomposição
    - A recursão

Introdução

Recursão é um método de decomposição de problemas onde o problema é decomposto em instâncias menores do mesmo tipo de problema

Introdução

- Considere por exemplo o problema de calcular o produto de todos os números inteiros de 1 até um dado valor n
  - Se $n = 1$ resolvemos o problema de forma trivial
  - Mas e se $n > 1$? Parece mais difícil, vamos tentar decompor o problema
  - Suponha que tivéssemos o valor do produto de todos os números de 1 até $n - 1$, como poderíamos calcular o produto de todos os números de 1 até n? Multiplicando o valor por n
  - Mas como encontramos a resposta para $n - 1$? Da mesma forma que encontramos a resposta para n, decompondo o problema recursivamente!

Funções recursivas

- Uma função é **recursiva** se ela é definida em termos dela mesmo
    - Para evitar uma definição circular, toda função recursiva precisa de pelo menos um caso não recursivo, chamado de **caso base**

Funções recursivas

- O produto de todos os números inteiros de 1 até n é chamado de fatorial e é definido da seguinte maneira

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

- Observe que a função é definida com dois casos, o caso base ($n = 0$) e um caso recursivo ($n > 0$)

Exemplo 1

Defina uma função que calcule o fatorial de um dado número n.

Exemplo 1

```
def fatorial(n):
    '''
    Natural -> Natural
    Devolve o valor n * (n - 1) * ... * 1.
    Exemplos
    >>> fatorial(0)
    1
    >>> fatorial(1)
    1
    >>> fatorial(3)
    6
    >>> fatorial(5)
    120
    '''
    if n == 0:
        fat = 1
    else:
        fat = n * fatorial(n - 1)
    return fat
```

Como projetar funções recursivas

- Pense em como decompor o problema
    - Se o problema é definido sobre os número inteiros, então uma forma de decompor o problema é subtraindo um
- Suponha (por fé) que a função está pronta e que ela funciona corretamente para o subproblema, então pense em como resolver o problema atual usando a solução do subproblema
- Pense no caso base
    - Como resolver o problema quando ele não pode mais ser decomposto?

Exemplo 2

Defina uma função que verifique se um dado número é par usando apenas subtração.

Exemplo 3

```
def par(n):
    '''
    Natural -> Booleano
    Devolve True se n é par, isto é, divisível por 2,
    False caso contrário.
    Exemplos
    >>> par(8)
    True
    >>> par(5)
    False
    '''
    if n == 0:
        e_par = True
    else:
        e_par = not par(n - 1)
    return e_par
```

Exemplo 3

Defina uma função que calcule a divisão inteira de dois números usando apenas subtração.

Atividades

1. Dado dois inteiros positivos a e b, defina uma função que calcule o resto da divisão de a por b usando apenas subtração.

2. Dado dois número inteiros positivos a e b, defina uma função que calcule o valor a^b usando apenas multiplicação.