Algoritmos Recursivos

Algoritmo recursivos: Introdução

- Às vezes podemos reduzir a solução de um problema com um conjunto particular de valores de entrada para a solução do mesmo problema com valores de entrada menores.
- Quando tal redução pode ser feita, a solução para o problema original pode ser encontrada via uma sequência de reduções, até que o problema tenha sido reduzido a algum caso inicial para o qual a solução é conhecida.
- Veremos que algoritmos que reduzem sucessivamente um problema ao mesmo problema entradas menores são usadas para resolver uma grande variedade de problemas.

  Tais algoritmos são chamados de <u>recursivos</u>.
- Um **algoritmo recursivo** é um algoritmo que resolve um problema reduzindo este problema a uma instância do mesmo problema com entradas menores.
- A seguir vamos ver vários exemplos de algoritmos recursivos.

Exemplos de algoritmos recursivos

- $\boxed{\text{Exemplo 18}}$ Dê um algoritmo recursivo para computar $n!$, onde n é um número inteiro não-negativo.

  Solução. Para construir um algoritmo recursivo que encontre $n!$, onde n é um inteiro não-negativo, podemos nos basear na definição recursiva de $n!$:

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

  Esta definição diz que para encontrar $n!$ para um inteiro particular n, podemos usar a etapa recursiva repetidamente vezes, em cada vez substituindo um valor da função fatorial pelo valor da função fatorial no próximo inteiro menor.

  Fazemos isso até atingir o passo base, em que o inteiro a ser computado é 0, e podemos inserir o valor conhecido de $0! = 1$.

- Exemplo 18 (Continuação)

Um pseudo-código para um algoritmo recursivo para a função fatorial é o seguinte.

procedure *factorial*(n: nonnegative integer)
if $n = 0$ **then return** 1
else return $n \cdot$ *factorial*$(n - 1)$
$\{$output is $n!\}$

Exemplo de execução do algoritmo para o valor de entrada $n = 4$:

Função	**Chamada recursiva**	**Valor de retorno**
`factorial(4)`	$4 \cdot$ `factorial(3)`	24
`factorial(3)`	$3 \cdot$ `factorial(2)`	6
`factorial(2)`	$2 \cdot$ `factorial(1)`	2
`factorial(1)`	$1 \cdot$ `factorial(0)`	1
`factorial(0)`	—	1

Exemplos de algoritmos recursivos

- $\boxed{\text{Exemplo 19}}$ Dê um algoritmo recursivo para calcular a^n, onde a é um número real diferente de zero e n é um inteiro não-negativo.

  Solução. Para construir um algoritmo recursivo que encontre a^n, onde a é um real diferente de zero e n é um inteiro não-negativo, podemos nos basear na definição recursiva de a^n:

$$an = \begin{cases} a \cdot a^{n-1}, & \text{se } n > 0 \\ 1, & \text{se } n = 0 \end{cases}$$

  Esta definição diz que para encontrar a^n para valores particulares de a e n, podemos usar a etapa recursiva repetidamente vezes, em cada vez substituindo um valor da função de exponenciação pelo valor da função de exponenciação com um expoente sendo o próximo inteiro menor.

  Fazemos isso até atingir o passo base, em que o valor a ser computado é a^0, e podemos inserir o valor conhecido de $a^0 = 1$.

Exemplos de algoritmos recursivos

- Exemplo 19 (Continuação)

Um pseudo-código para um algoritmo recursivo para a função de exponenciação é o seguinte.

procedure *power*(a: nonzero real number, n: nonnegative integer)
if $n = 0$ **then return** 1
else return $a \cdot power(a, n-1)$
{output is a^n}

Exemplo de execução do algoritmo para o valor de entrada $a = 2$, $n = 4$:

Função	Chamada recursiva	Valor de retorno
`power(2,4)`	2 · `power(2,3)`	16
`power(2,3)`	2 · `power(2,2)`	8
`power(2,2)`	2 · `power(2,1)`	4
`power(2,1)`	2 · `power(2,0)`	2
`power(2,0)`	—	1

Exemplos de algoritmos recursivos

- $\boxed{\text{Exemplo 20}}$ Dada uma sequência $L = a_1, a_2, \ldots, a_n$ de elementos e um elemento arbitrário x, uma **pesquisa linear** do elemento x em L retorna:
  - o menor valor de i tal que $a_i = x$, caso o valor x esteja presente na lista L; ou
  - o valor 0, caso o valor x não esteja na lista L.

  Expresse a pesquisa linear como um algoritmo recursivo.

  Solução. Vamos chamar de $search(i, j, x)$ o procedimento que procura a primeira ocorrência de x na sub-sequência $a_i, a_{i+1}, \ldots, a_j$ de L iniciada em a_i e terminada em a_j.

  A entrada para o algoritmo recursivo da pesquisa linear consiste na tripla $(1, n, x)$, pois queremos que o elemento x seja procurado na lista L toda, ou seja, entre a_1 e a_n.

- $\boxed{\text{Exemplo 20}}$ (Continuação)

O algoritmo funciona assim:

1. Se o primeiro termo da sub-sequência a ser pesquisada for o próprio x, o algoritmo retorna o índice i deste termo.
2. Se a sub-sequência a ser pesquisada só tem um elemento e este elemento não é x, o algoritmo retorna 0.
3. Se o primeiro termo da sub-sequência a ser pesquisada não é x, mas a sub-sequência tem termos adicionais, o mesmo algoritmo é repetido na sub-sequência obtida retirando-se o primeiro termo da sub-sequência atual.

Exemplos de algoritmos recursivos

- $\boxed{\text{Exemplo 20}}$ (Continuação)

  Um pseudo-código para um algoritmo recursivo para a função de pesquisa linear é o seguinte.

procedure *search*(i, j, x: integers, $1 \leq i \leq j \leq n$)
if $a_i = x$ **then**
    return i
else if $i = j$ **then**
    return 0
else
    return *search*($i + 1, j, x$)

- $\boxed{\text{Exemplo 20}}$ (Continuação)

Exemplo de execução do algoritmo de pesquisa linear para os valores de entrada

$$i = 1, \qquad j = 5, \qquad x = 17,$$

na lista

$$a_1 = 3, \quad a_2 = 12, \quad a_3 = 20, \quad a_4 = 17, \quad a_5 = 5:$$

Função	Chamada recursiva	Valor de retorno
`search(1,5,17)`	`search(2,5,17)`	4
`search(2,5,17)`	`search(3,5,17)`	4
`search(3,5,17)`	`search(4,5,17)`	4
`search(4,5,17)`	`-----`	4

Exemplos de algoritmos recursivos

- $\boxed{\text{Exemplo 20}}$ (Continuação)

Exemplo de execução do algoritmo de pesquisa linear para os valores de entrada

$$i = 1, \qquad j = 5, \qquad x = 10,$$

na lista

$$a_1 = 3, \quad a_2 = 12, \quad a_3 = 20, \quad a_4 = 17, \quad a_5 = 5 :$$

Função	**Chamada recursiva**	**Valor de retorno**
`search(1,5,10)`	`search(2,5,10)`	0
`search(2,5,10)`	`search(3,5,10)`	0
`search(3,5,10)`	`search(4,5,10)`	0
`search(4,5,10)`	`search(5,5,10)`	0
`search(5,5,10)`	—	0

- $\boxed{\text{Exemplo 21}}$ Dada uma sequência $L = a_1, a_2, \ldots, a_n$ de de inteiros positivos em ordem crescente e um elemento arbitrário x, uma **pesquisa binária** do elemento x em L funciona assim:

  1. Compare o elemento x a ser pesquisado com o termo do meio da sequência, ou seja, o termo $a_{\lfloor (n+1)/2 \rfloor}$.
  2. Se x for igual a este termo, retorne a localização deste termo no sequência.
  3. Caso contrário:
     a) faça uma nova pesquisa binária na primeira metade da sequência original se x for menor que o termo do meio; ou
     b) faça uma nova pesquisa binária na segunda metade da sequência original se x for maior que o termo do meio; ou
     c) retorne 0 se não há mais elementos a serem pesquisados.

  Expresse a pesquisa binária como um algoritmo recursivo.

- $\boxed{\text{Exemplo 21}}$ (Continuação)

Solução. Um pseudo-código para um algoritmo recursivo para a função de pesquisa binária é o seguinte.

procedure *binary search*(i, j, x: integers, $1 \leq i \leq j \leq n$)
$m := \lfloor (i + j)/2 \rfloor$
if $x = a_m$ **then**
    return m
else if ($x < a_m$ and $i < m$) **then**
    return *binary search*($i, m - 1, x$)
else if ($x > a_m$ and $j > m$) **then**
    return *binary search*($m + 1, j, x$)
else return 0
{output is location of x in $a_1, a_2, \ldots, a_n$ if it appears; otherwise it is 0}

- Exemplo 21 (Continuação)

Exemplo de execução do algoritmo de pesquisa binária para os valores de entrada

$$i = 1, \qquad j = 12, \qquad x = 20,$$

na lista ordenada

$$\begin{array}{cccccc} a_1 = 1, & a_2 = 5, & a_3 = 8, & a_4 = 10, & a_5 = 12, & a_6 = 15, \\ a_7 = 17, & a_8 = 20, & a_9 = 23, & a_{10} = 25, & a_{11} = 28, & a_{12} = 30: \end{array}$$

Função	**Chamada recursiva**	**Valor de retorno**
`bin_search(1,12,20)`	`bin_search(7,12,20)`	8
`bin_search(7,12,20)`	`bin_search(7,8,20)`	8
`bin_search(7,8,20)`	`bin_search(8,8,20)`	8
`bin_search(8,8,20)`	——	8

- $\boxed{\text{Exemplo 21}}$ (Continuação)

  Exemplo de execução do algoritmo de pesquisa binária para os valores de entrada

$$i = 1, \qquad j = 12, \qquad x = 7,$$

na <u>lista ordenada</u>

$$\begin{array}{cccccc} a_1 = 1, & a_2 = 5, & a_3 = 8, & a_4 = 10, & a_5 = 12, & a_6 = 15, \\ a_7 = 17, & a_8 = 20, & a_9 = 23, & a_{10} = 25, & a_{11} = 28, & a_{12} = 30: \end{array}$$

Função	**Chamada recursiva**	**Valor de retorno**
`bin_search(1,12,7)`	`bin_search(1,5,7)`	0
`bin_search(1,5,7)`	`bin_search(1,2,7)`	0
`bin_search(1,2,7)`	`bin_search(2,2,7)`	0
`bin_search(2,2,7)`	——	0

Demonstrando a correção de algoritmos recursivos

- Podemos usar a indução matemática para demonstre a correção de algoritmos recursivos.

- $\boxed{\text{Exemplo 22}}$ Demonstre que o algoritmo recursivo que provemos em um exemplo anterior para calcular a exponenciação de um número real com expoente inteiro não-negativo está correto.

  Solução. Vamos usar indução matemática no expoente n.

  Passo base: Se $n = 0$ nosso algoritmo recursivo nos diz que $power(a, 0) = 1$, o que está correto porque $a^0 = 1$ para qualquer número real a.

Demonstrando a correção de algoritmos recursivos

- $\boxed{\text{Exemplo 22}}$ (Continuação)

Passo indutivo: A hipótese de indução é que o algoritmo recursivo computa o valor correto da potência para um inteiro arbitrário, ou seja, que $power(a, k) = a^k$ para qualquer valor real $a \neq 0$ e um inteiro não-negativo arbitrário k.

Temos que mostrar que se a hipótese de indução é verdadeira, então o algoritmo computa a resposta correta para $k + 1$, ou seja, teremos $power(a, k + 1) = a^{k+1}$.

Note que como k é um inteiro positivo, o algoritmo faz $power(a, k + 1) = a \cdot power(a, k)$.

Como pela hipótese de indução temos $power(a, k) = a^k$, concluímos que $power(a, k + 1) = a \cdot a^k = a^{k+1}$, e o algoritmo também está correto para $k + 1$.