Estruturas de Dados I

Prof. Leonardo C. R. Soares - *leonardo.soares@ifsudestemg.edu.br*

Instituto Federal do Sudeste de Minas Gerais

18 de outubro de 2024

Apresentação da disciplina

Carga horária

- 60horas;
- 4 aulas por semana;
- Quinta (10:30 - 12:00) e Sexta (10:30 - 12:00).

Apresentação da disciplina

Carga horária

- 60horas;
- 4 aulas por semana;
- Quinta (10:30 - 12:00) e Sexta (10:30 - 12:00).

Ementa

- Recursividade
- Listas lineares e encadeadas;
- Pilhas e filas;
- Árvores;
- Métodos de busca e ordenação.

Apresentação da disciplina

Bibliografia básica

- NINA EDELWEISS; RENATA GALANTE. Estruturas de Dados. 36. ed. Bookman, 2009.
- PIVA JUNIOR, D. et al. Estrutura de Dados e Técnicas de programação. 1. ed. Elsevier, 2014.
- ROBERTO FERRARI; ET. AL. Estruturas de Dados com Jogos. 1. ed. Elsevier, 2014.

Apresentação da disciplina

Bibliografia complementar

- **THOMAS H. CORMEN; (ET. AL). Algoritmos: teoria e prática. 3. ed. Elsevier, 2012.**
- NIVIO ZIVIANI. Projeto de Algoritmos com Implementações em Java e C++. 1. ed. Thomson Learning, 2006.
- MARCO MEDINA; CRISTINA FERTIG. Algoritmos e Programação: teoria e prática. 2. ed. Novatec, 2006.
- CORMEN, T.H. Desmistificando Algoritmos. Rio de Janeiro: Elsevier, 2014.
- GOODRICH, Michael T.; TAMASSIA, Robert. Estruturas de Dados e Algoritmos em Java. 4. ed. Porto Alegre: Bookman, 2007.

Apresentação da disciplina

Avaliações

Nossa disciplina possuirá quatro atividades avaliativas, sendo três avaliações individuais, teóricas e sem consulta, cada uma valendo 3 pontos. E uma lista de exercícios valendo 1 ponto.

- 21/11/2024 - 2pts;
- 20/12/2023 - 2pts;
- 13/02/2025 - 2pts;
- 13/03/2024 - 3pts;
- Exercícios - 1pt[a]

[a]Caso o discente resolva corretamente mais de 90% dos exercícios, receberá 1 ponto; Caso o discente resolva corretamente entre 60% e 89% dos exercícios, receberá 0,5 pontos; Caso o discente resolva corretamente entre 50% e 59% dos exercícios, receberá 0,25 pontos; Abaixo de 50% será atribuído nota zero.

Algoritmos recursivos

Definição

Um algoritmo pode ser composto por funções, que, por sua vez, podem invocar outras funções.

Algoritmos recursivos

Definição

Um algoritmo pode ser composto por funções, que, por sua vez, podem invocar outras funções.

Quando uma função invoca a si própria, a denominamos **função recursiva**.

Algoritmos recursivos

Definição

Um algoritmo pode ser composto por funções, que, por sua vez, podem invocar outras funções.

Quando uma função invoca a si própria, a denominamos **função recursiva**.

Uma função recursiva pode ser:

- Direta: A função X invoca a própria função X;

Algoritmos recursivos

Definição

Um algoritmo pode ser composto por funções, que, por sua vez, podem invocar outras funções.

Quando uma função invoca a si própria, a denominamos **função recursiva**.

Uma função recursiva pode ser:

- Direta: A função X invoca a própria função X;
- Indireta: A função X invoca a função Y que, por sua vez, invoca a função X.

Algoritmos recursivos

Princípio

A recursividade está intimamente relacionada ao princípio de indução matemática.

Parte-se da hipótese que a solução para um problema de tamanho t pode ser obtida a partir da solução para o mesmo problema, porém, de tamanho $t - 1$.

Algoritmos recursivos

Princípio

A recursividade está intimamente relacionada ao princípio de indução matemática.

Parte-se da hipótese que a solução para um problema de tamanho t pode ser obtida a partir da solução para o mesmo problema, porém, de tamanho $t-1$.

Recursão vs. Iteração

Toda função recursiva possui uma versão iterativa construída com a utilização de estruturas de repetições.

Algoritmos recursivos

Projeto

Um algoritmo recursivo é composto, em sua forma mais simples, por uma **condição de parada** e por um **passo recursivo**.

Algoritmos recursivos

Projeto

Um algoritmo recursivo é composto, em sua forma mais simples, por uma **condição de parada** e por um **passo recursivo**.

Condição de parada

Garante que a recursividade seja finita. Geralmente é definida sobre um **caso base**. Por exemplo, a condição $n > 0$ garante que o algoritmo só será executado enquanto n for positivo.

Algoritmos recursivos

Projeto

Um algoritmo recursivo é composto, em sua forma mais simples, por uma **condição de parada** e por um **passo recursivo**.

Condição de parada

Garante que a recursividade seja finita. Geralmente é definida sobre um **caso base**. Por exemplo, a condição $n > 0$ garante que o algoritmo só será executado enquanto n for positivo.

Passo recursivo

Processa os diferentes valores de retorno e faz as chamadas recursivas.

Algoritmos recursivos

Exemplo 1

Um algoritmo que imprima, recursivamente, os cinco primeiros números inteiros.

```java
public static void recursiveFunction(int n){
    if (n<5){
        System.out.println(n);
        recursiveFunction(n+1);
    }
}
```

Exemplo 1

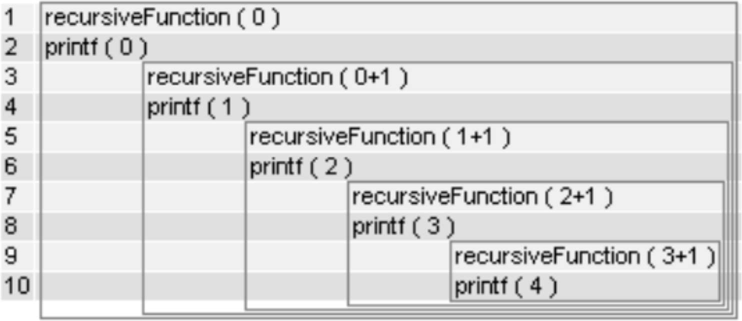

Figura: Ilustração da execução do exemplo

Algoritmos recursivos

Exemplo 2

Um algoritmo que imprima, recursivamente, os cinco primeiros números inteiros mas em ordem decrescente.

```
public static void recursiveFunction(int n){
    if (n<5){
        recursiveFunction(n+1);
        System.out.println(n);
    }
}
```

Exemplo 2

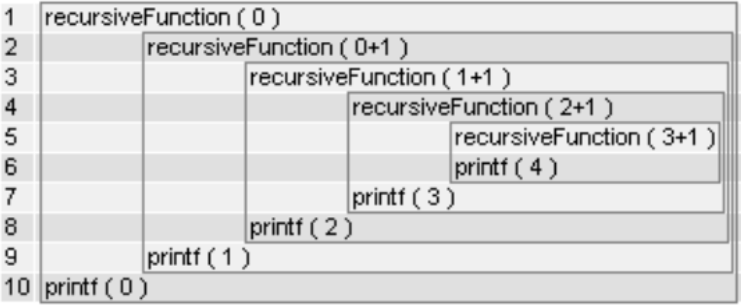

Figura: Ilustração da execução do exemplo

Algoritmos recursivos

Exemplo 3

Um algoritmo que some, recursivamente, todos os números inteiros entre m e n, inclusive.

```java
public static int soma(int m, int n){
    if (m == n)
        return m;
    else
        return m+soma(m+1,n);
}
```

Exemplo 3

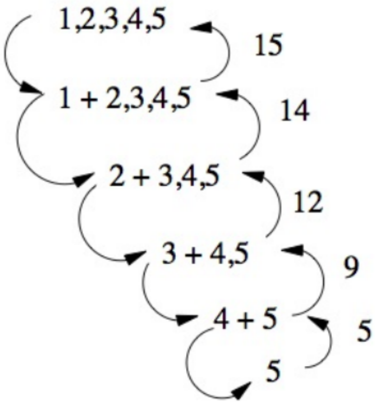

Figura: Árvore de recursão para $m = 1$ e $n = 5$

Algoritmos recursivos

Recursão em cauda vs. Recursão crescente

Na **Recursão Crescente** (ou funções crescentemente recursivas), depois de encerrada a chamada recursiva outras operações ainda são realizadas.

Algoritmos recursivos

Recursão em cauda vs. Recursão crescente

Na **Recursão Crescente** (ou funções crescentemente recursivas), depois de encerrada a chamada recursiva outras operações ainda são realizadas.

Na **Recursão em Cauda**, não existe processamento a ser realizado depois de encerrada a chamada recursiva, ou seja, a chamada recursiva é a última instrução a ser executada.

Algoritmos recursivos

Recursão em cauda vs. Recursão crescente

Na **Recursão Crescente** (ou funções crescentemente recursivas), depois de encerrada a chamada recursiva outras operações ainda são realizadas.

Na **Recursão em Cauda**, não existe processamento a ser realizado depois de encerrada a chamada recursiva, ou seja, a chamada recursiva é a última instrução a ser executada.

A recursão em cauda geralmente é mais rápida do que recursão crescente, uma vez que não é necessário armazenar todo o contexto na pilha de execução do programa.

Recursão crescente vs. Recursão em cauda

```java
public static long somaCrescente(long x){
    if (x == 1)
        return x;
    else
        return x+somaCrescente(x-1);
}
```

Recursão crescente vs. Recursão em cauda

```java
public static long somaCrescente(long x){
    if (x == 1)
        return x;
    else
        return x+somaCrescente(x-1);
}
```

```java
public static long somaCauda(long x, long total)
{
    if (x == 0)
        return total;
    else
        return somaCauda(x-1, total+x);
}
```

Exercícios rápidos

Desenvolva um algoritmo recursivo para calcular o fatorial de um número qualquer digitado pelo usuário, sabendo que $fatorial(n) = n \times fatorial(n-1) \forall n \geq 1, fatorial(0) = 1.$

Exercícios rápidos

Desenvolva um algoritmo recursivo para calcular o fatorial de um número qualquer digitado pelo usuário, sabendo que $fatorial(n) = n \times fatorial(n-1) \forall n \geq 1, fatorial(0) = 1$.

```
public static long fatorial(int n){
    if (n == 0)
        return 1;
    else
        return n * fatorial(n-1);
}
```

Exercícios rápidos

Desenvolva um algoritmo recursivo que retorne o enésimo número da sequência fibonacci, sabendo que:

- $fibo(0) = 0;$
- $fibo(1) = 1;$
- $fibo(n) = fibo(n-1) + fibo(n-2) \forall n \geq 2.$

Utilize este método para desenvolver um programa que imprima os enésimos primeiros termos desta sequência.

Exercícios rápidos

Desenvolva um algoritmo recursivo que retorne o enésimo número da sequência fibonacci, sabendo que:

- $fibo(0) = 0;$
- $fibo(1) = 1;$
- $fibo(n) = fibo(n-1) + fibo(n-2) \forall n \geq 2.$

Utilize este método para desenvolver um programa que imprima os enésimos primeiros termos desta sequência.

```java
public static long fibo(int n){
    if ((n == 0) || (n == 1))
        return n;
    else
        return fibo(n-1)+fibo(n-2);
}
```

O programa inteiro pode ser baixado como exemplo aqui.

Exercícios

1. Implemente uma versão iterativa para os algoritmos recursivos apresentados para o cálculo do fatorial e do fibonacci;
2. Implemente um algoritmo recursivo para calcular 2^n;
3. Implemente um algoritmo recursivo para se um número é primo ou não;
4. Implemente um algoritmo recursivo que faça a multiplicação de dois números naturais através de somas sucessivas (ex.: 5*4 = 5+5+5+5);
5. Implemente um algoritmo recursivo que calcule o máximo divisor comum entre dois números, sabendo que:
   - $mdc(x, y) = y$, se $x \geq y$ e $x \ mod \ y = 0$
   - $mdc(x, y) = mdc(y, x)$, se $x < y$
   - $mdc(x, y) = mdc(y, x \ mod \ y)$, caso contrário.

*


thank you
danke
謝謝
ngiyabonga
teşekkür ederim
рахмат
Баярлалаа
спасибо
gracias
obrigado
dziękuję
bedankt
go raibh maith agat
mochchakkeram
sukriya
kop khun krap
merci
terima kasih
grazie
arigatô
dakujem
dank je
takk
sagolun
hvala
mauruuru
감사합니다
ありがとう
мерси
ευχαριστώ
tapadh leat
mersi
barka
welalin
tack
spas
faafetai lava
vinaka
blagodaram
misaotra
matondo
paldies
grazzi
mahalo
kiitos
dankie
dhanyavad
хвала
asante
manana
obrigada
tenki
enkosi
chnorakaloutioun
grafias ago
gracies
sulpay
djiere dieuf
mamnun
дякую
trugarez
sobodi
děkuji
mèsi
najis tuke
tanemirt
rahmet
diolch
dhanyavadagalu
shukriya
merce
kam sah hamnida
rahmat
xiexie
taiku
akun
dankon
köszönöm
gracie
nanni
nandri
bayarlalaa
didi madloba
chokrane
murakoze