Conceitos Fundamentais

Estrutura de Dados 1

Algoritmos

Estrutura de Dados 1

Um pouco de história...

- Por volta do ano de 825 d.C., um matemático persa escreveu uma série de textos matemáticos
- Utilizava o sistema numérico decimal, criado na Índia cerca de 200 anos antes
- Detalhou métodos para adição, multiplicação, radiciação entre outros
- Seu nome era Buchafar Al-Kharismi

Um pouco de história...

- Os métodos descritos por Al-Kharismi eram
  - **Precisos** Não dependiam da “interpretação pessoal” do leitor
  - **Inequívocos** Sem as ambiguidades comuns da língua escrita
  - **Mecânicos** Podiam ser reproduzidos de maneira “automatizada”
  - **Eficientes** Eram a melhor solução possível para determinado problema
  - **Corretos** Você sempre chegará à mesma solução

- Eram, em essência, algoritmos

Definição

- De um modo geral (e até informal):

> *Um algoritmo é um conjunto de regras que definem precisamente uma sequência de operações*

Stone, Harold S. *Introduction to Computer Organization and Data Structures*. McGraw-Hill, New York, 1971, 312p.

Mais história

- Al-Kharismi não inventou os algoritmos
  - Ele sistematizou diversos métodos já conhecidos
  - Alguns anos antes, ele mesmo havia descrito algoritmos para resolver sistemas lineares e equações quadráticas

- Antes dele, já havia importantes métodos numéricos:
  - A multiplicação de dois números é atribuída aos egípcios (c. 2000 a.C.)
  - A fatoração e a raiz quadrada têm origem babilônica (c. 1600 a.C.)
  - O Algoritmo de Euclides data de cerca 300 a.C.
  - O Crivo de Erastóstenes foi descrito por volta de 200 a.C.

Para saber mais

- *Timeline of algorithms*

https://en.wikipedia.org/wiki/Timeline_of_algorithms

O Algoritmo de Euclides

- Sejam m e n dois números inteiros
- Dizemos que n divide m se m % n == 0
- Note que: m % n == 0 se, e somente se, m == dn para algum inteiro d
- O máximo divisor comum de dois números inteiros m e n é o maior número inteiro que divide ambos
  - É denotado por mdc(m, n)

O Algoritmo de Euclides

- Para calcular o mdc(m, n) para 0 <= n < m, o algoritmo de Euclides usa a seguinte recorrência:

  mdc(m,0) == m;
  mdc(m,n) == mdc(n, m % n), para n > 0.

- Por exemplo,

  mdc(12,18) == mdc(18,12) == mdc(12,6) == mdc(6,0) == 6

O Algoritmo de Euclides

- O Algoritmo de Euclides tem uma importante característica: ele é recursivo

- Matematicamente, a recursão é o ato de definir um objeto (função), em termos do próprio objeto

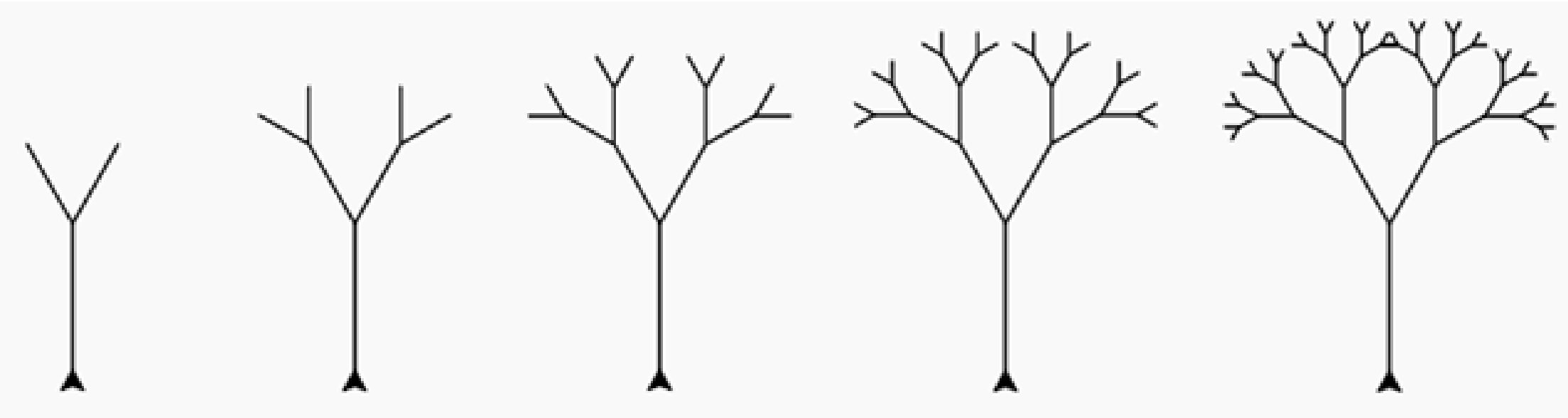
- Outro exemplo clássico: fatorial de n

  0! == 1

  n! == n*(n-1)!

Recursão

Estrutura de Dados 1


1st Recursion
2nd Recursion
3rd Recursion
4th Recursion
5th Recursion

Definições

- Em computação, a recursão ocorre quando um dos passos de um determinado algoritmo envolve a repetição desse mesmo algoritmo

- Um procedimento que se utiliza da recursão é dito recursivo

- Também é dito recursivo qualquer objeto que seja resultado de um procedimento recursivo

Definições

- É possível, por meio de recursão, obter um objeto ou sequências infinitas a partir de um componente finito

- O conjunto dos números naturais, por exemplo, pode ser definido formalmente por:
  - Seja 0 um número natural
  - Cada número natural n tem um sucessor (n + 1) que é também um número natural

Definições

- **Caso base**
  - Parte não recursiva, também chamada de âncora, ocorre quando a resposta para o problema é trivial.
- **Passo indutivo**
  - Parte da definição que especifica como cada elemento (solução) é gerado a partir do precedente
- Voltando ao fatorial:

  0! == 1 (caso base)

  n! == n*(n-1)! (passo indutivo)

Trabalhando com recursão

- Se a instância é pequena, resolva-a diretamente (caso base)

senão

1. Reduza-a a uma instância menor do mesmo problema
2. Aplique o método à instância menor
3. Volte à instância original

Trabalhando com recursão

Principal regra:

- Qualquer função recursiva deve verificar se o caso base foi atingido antes da nova chamada recursiva

Além disso:

- Deve-se pensar em como quebrar um problemas em subproblemas que possam ser resolvidos instantaneamente

Voltando ao fatorial

```
// Para n >= 0
int fatorial(int n)
{
    if (n == 0)      // caso base
        return 1;
    else             // passo indutivo
        return (n * fatorial(n-1));
}
```

Trabalhando com recursão

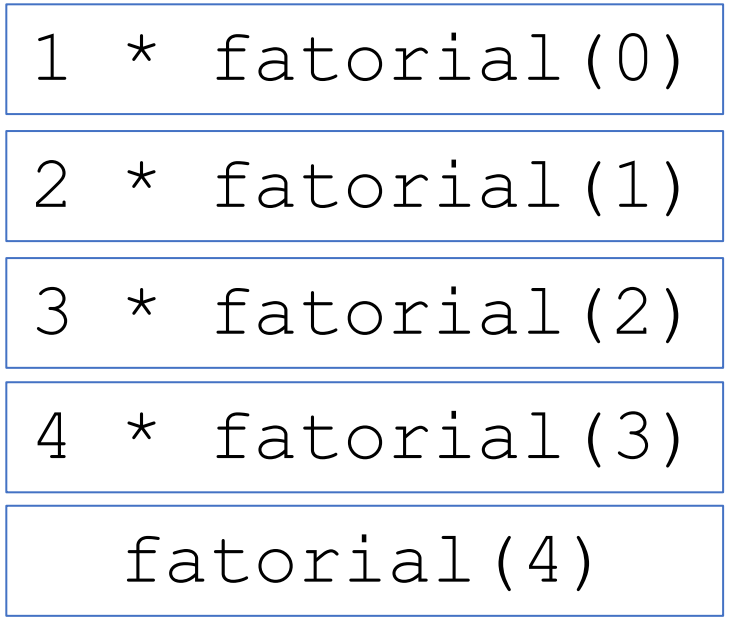
- O compilador implementa um procedimento recursivo é por meio de uma pilha

- Nessa pilha, são armazenados os dados usados em cada chamada de uma função que ainda não terminou de ser processar

Trabalhando com recursão

```
fatorial(4)
[ 4 * fatorial(3) ]
[ 4 * [ 3 * fatorial(2) ] ]
[ 4 * [ 3 * [ 2 * fatorial(1) ] ] ]
[ 4 * [ 3 * [ 2 * [ 1 * fatorial(0) ] ] ] ]
[ 4 * [ 3 * [ 2 * [ 1 * 1] ] ] ]
[ 4 * [ 3 * [ 2 * 1 ] ] ]
[ 4 * [ 3 * 2 ] ]
[ 4 * 6 ]
24
```

Visualmente...



Outro exemplo: Sequência de Fibonacci

- Sequência numérica obtida de forma recursiva

  f(0) = 1

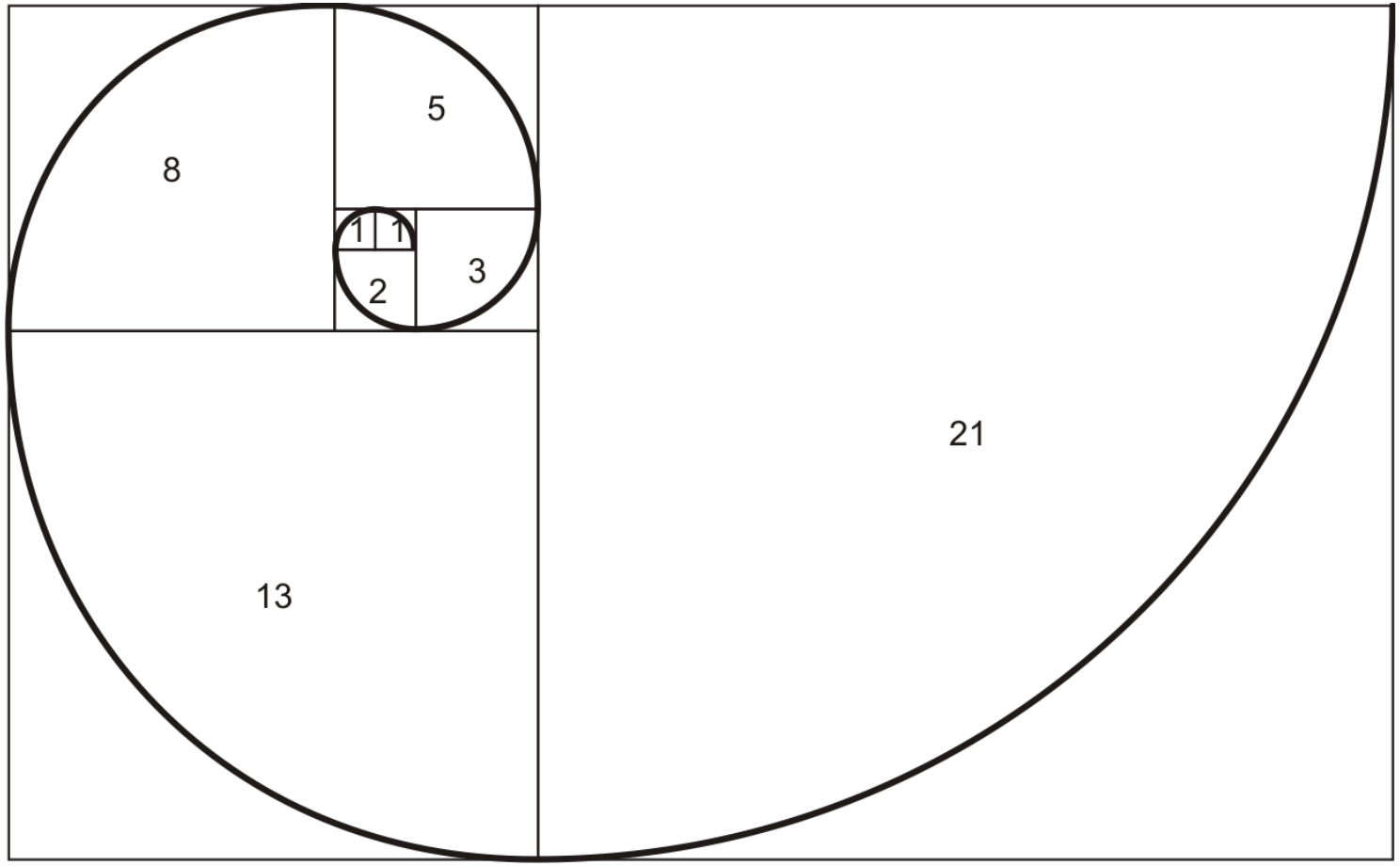
  f(1) = 1

  f(n) = f(n-1) + f(n-2)

- Modelo observado em muitos fenômenos biológicos
- Os primeiros dez termos da sequência são:

  0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55

Sequência de Fibonacci



Exercício

- Quantas chamadas recursivas são necessárias para calcular a sequência de Fibonacci para n=10?

- Implemente, em C, uma função recursiva que imprima a sequência de Fibonacci
- Avalie o seu programa com diferentes valores de n, começando em 0 e incrementando de 5 em 5 (por exemplo)

Agradecimentos

- Esses slides foram adaptados dos materiais disponibilizados pelos professores:
  - Moacir Ponti Jr. (ICMC/USP)
  - José Coelho de Pina (IME/USP)