ESTRUTURAS DE DADOS II

MSC. DANIELE CARVALHO OLIVEIRA

DOUTORANDA EM CIÊNCIA DA COMPUTAÇÃO - USP

MESTRE EM CIÊNCIA DA COMPUTAÇÃO – UFU

BACHAREL EM CIÊNCIA DA COMPUTAÇÃO - UFJF

2

RECURSÃO

3 INDUÇÃO MATEMÁTICA

- Uma proposição P(X) pode ser provada por indução matemática (ou indução finita) do seguinte modo:
  - Base: Comprovamos que P é verdadeira para os casos básicos (X=0 ou X=1, por exemplo).
  - Hipótese indutiva: Supomos que P seja verdadeira para o caso genérico X=N.
  - Passo indutivo: Demonstramos que P também é verdadeira para o caso X=N+1.

- Ideia: como a proposição vale para o caso inicial e o passo é correto, então essa proposição também será válida para todos os casos subsequentes.

4 EXEMPLO

- Proposição: numa sequência de peças de dominó que estejam em pé, suficientemente próximas entre si, se a primeira caiu então todas caíram.

- Prova por indução:
  - **Base:** A primeira peça caiu (por definição).
  - **Hipótese indutiva:** Supomos que a n-ésima tenha caído.
  - **Passo indutivo:** Como a n-ésima peça caiu e ela está suficientemente perto da seguinte, então a (n+1)-ésima peça também terá caído.

5 UM EXEMPLO PRÁTICO

- Demonstre que, para todos os números naturais $x > 0$ e n, $x^n - 1$ é divisível por $x - 1$.

- Prova por indução (em n):
  - **Base:** Para $n = 1, x^1 - 1$ é divisível por $x - 1$.
  - **Hipótese indutiva:** Para um valor qualquer $k > 1$, supomos que $x^k - 1$ seja divisível por $x - 1$, para todo x>0 natural.
  - **Passo indutivo:**
    - Sabemos que $x^{k+1} - 1 = x^{k+1} - 1 - x + x = x(x^k - 1) + (x - 1)$
    - Pela hipótese de indução, a primeira parcela é divisível por $x - 1$.
    - Como sabemos que a segunda também é, o passo está provado.

6 DEFINIÇÕES RECURSIVAS OU INDUTIVAS

- Em uma definição recursiva, uma classe de objetos relacionados é definida em termos desses próprios objetos.

- Uma definição recursiva envolve:
  - Uma **base**, onde um ou mais objetos elementares são definidos.
  - Um **passo indutivo**, onde objetos subsequentes são definidos em termos de objetos já conhecidos.

7 EXEMPLO

- Definição recursiva dos números naturais:
  - Base: o número 0 está em N.
  - Passo indutivo: se n está em N, então n + 1 também está.

- O conjunto dos números naturais é o menor conjunto que satisfaz as condições acima.

8 OUTRO EXEMPLO

- As expressões numéricas são comumente definidas de forma recursiva:
  - **Base:** Todos os operandos atômicos (números, variáveis, etc.) são expressões numéricas.
  - **Passo indutivo:** Se E1 e E2 são expressões numéricas então (E1 + E2), (E1 – E2), (E1 * E2), (E1 / E2) e (-E1) também são.

9 RECURSIVIDADE

- Um objeto é dito recursivo se pode ser definido em termos de si próprio
- Recursão é o processo de se usar um objeto recursivo
- Uma função é dita recursiva se invoca a si mesma, direta ou indiretamente.

10 ALGORITMOS RECURSIVOS

- Recursão (ou recursividade) é um método de programação no qual um procedimento (função, método, etc.) pode chamar a si mesmo.

- Algoritmos recursivos possuem uma clara analogia com o método indutivo:
  - **Base:** Uma entrada elementar, que pode ser resolvida diretamente.
  - **Parte indutiva:** Chamadas a si mesmo, mas com entradas mais simples.

- A ideia é aproveitar a solução de um ou mais subproblemas para resolver todo o problema.

11 EXEMPLO

Algoritmo recursivo para o cálculo de fatorial:

0! = 1! = 1

n! = n.(n-1)!

```
int fat(int n) {
  if (n <= 1) return 1;
  return n*fat(n-1);
}
```

Execução de `fat(4)`:

```
call ↓
  fat(4)
  call ↓
    fat(3)
    call ↓
      fat(2)
      call ↓
        fat(1)
```

```
return 24 ↑
   fat(4)
return 6 ↑
   fat(3)
return 2 ↑
   fat(2)
return 1 ↑
```

ANÁLISE DA COMPLEXIDADE DE TEMPO

- Seja T(n) o tempo de execução de fat(n):
  - Base: $T(1) = a$.
  - Parte indutiva: $T(n) = T(n-1) + b, n>1$.

- Cálculos:
  - $T(2) = T(1) + b = a + b$
  - $T(3) = T(2) + b = a + 2b$
  - $T(4) = T(3) + b = a + 3b$
  - Generalizando: $T(n) = a + (n-1)b$
  - Portanto: $T(n) = O(n)$

13 MECANISMO DA RECURSÃO

- Durante a execução de um programa, quando um procedimento é chamado, é preciso guardar o contexto atual de processamento (valores de parâmetros e variáveis locais, endereço de retorno, etc.) para que depois seja possível recomeçar de onde se parou

- Deseja-se que o último procedimento interrompido seja o primeiro a recomeçar a sua execução. Por isso, o sistema operacional utiliza uma pilha de execução, alocada na memória.

- Portanto, os algoritmos recursivos poderiam se escritos de forma iterativa: basta utilizar uma pilha explícita, que simule o gerenciamento realizado pelo sistema operacional.

14 ALGORITMO ITERATIVO EQUIVALENTE

- Costuma-se calcular o fatorial de um número natural n da seguinte maneira:

```
int fat(int n) {
    int f = 1;
    while (n > 0)
        f *= n--;
    return f;
}
```

- É fácil constatar que esse algoritmo iterativo também gasta tempo O(n), ou seja, tem a mesma complexidade que a sua versão recursiva.

- No entanto, é mais rápido… Por quê?
- E com relação às complexidades de espaço?

15 VANTAGENS/DESVANTAGENS

- A Recursão deve ser utilizada com critério: não há regras gerais
- Usualmente, é menos eficiente que o seu equivalente iterativo (devido ao overhead da pilha de execução), mas essa diferença nem sempre é decisiva.
- A sua transformação em uma versão iterativa nem sempre é trivial
- Muitas vezes, é vantajosa em clareza, legibilidade e simplicidade de código.

Trabalho

- Resolva com algoritmos recursivos:
  - Dado um número natural, imprimir recursivamente a sua representação binária.
  - (Busca binária) Dado um vetor ordenado de tamanho n, verificar se um determinado elemento está ou não presente.

EXTRA

- URI
  - 1028; 1029; 1030; 1031; 1032; 1191; 1660

CONSIDERAÇÕES

- Nós vimos apenas uma base da Análise de Algoritmos.
- É um assunto extremamente relevante que necessita de maior aprofundamento.
  - Análise de Algoritmos
    - https://pt.coursera.org/course/aofa
  - Algoritmos, Parte I e II
    - https://pt.coursera.org/course/algs4partI
    - https://pt.coursera.org/course/algs4partII
  - Algoritmos: Design e Análise, Parte 1 e II
    - https://pt.coursera.org/course/algo
    - https://pt.coursera.org/course/algo2
  - Análise de Algoritmos – Skiena
    - https://www.youtube.com/playlist?list=PLOtl7M3yp-DV69F32zdK7YJcNXpTunF2b
  - Programming Challenges
    - https://www.youtube.com/playlist?list=PL07B3F10B48592010

FIM DA AULA 8

Próxima aula:

Hash