Estrutura de Dados

Recursão

Profa. Ana Cristina dos Santos

email: ana.csantos@sp.senac.br

Tópicos da Aula

- Conceitos básicos de recursividade
- Aplicação da recursividade no contexto da computação
- Algoritmos recursivos *versus* algoritmos iterativos

Recursividade

- Um algoritmo recursivo (ou método recursivo) tem a característica de realizar chamadas **a si próprio**
  - Executando esse procedimento algumas vezes até achar a solução de um problema
- O problema geral é dividido em pequenos subproblemas
  - É mais fácil solucionar pequenos problemas e unir as suas soluções posteriormente

Recursividade

- Veja abaixo o pseudocódigo para montar um método recursivo:

```
se a instância em questão é pequena (menor possível),
    resolva-a diretamente
senão
    reduza-a a uma instância menor do mesmo problema,
    aplique o método à instância menor
    e volte a instância original.
fimse
```

Recursividade

- As **vantagens** do conceito de recursividade são:
  - Redução significativa do tamanho do código-fonte
  - Permite programar algoritmos de forma clara e concisa
- Já as suas **desvantagens** estão relacionadas:
  - Ao baixo desempenho em algumas situações devido ao gerenciamento das chamadas dos métodos
  - A dificuldade na depuração do programa recursivo aumenta de acordo com a profundidade da recursão.

Recursividade

- Definição de um **método fatorial iterativo**:
  - n! = 1, , para n = 0
  - n! = 1 x 2 x 3 x ... x n, para n >= 1

- Definição de um **método fatorial recursivo**:
  - n! = 1, , para n = 0;
  - n! = n x (n – 1)!, para n >= 1
- Vamos implementar esses dois métodos

Recursividade

- O funcionamento da função recursiva utiliza uma **pilha** para armazenar os dados utilizados em cada chamada do método
  - Todas as **variáveis locais** são armazenadas nessa pilha, informando o resultado atual do processo
  - Quando a função do **topo** da pilha **finalizar**, todas as demais funções serão finalizadas e **desempilhadas** da pilha

Recursividade

- Embora a implementação seja simples, o custo de uma execução recursiva pode ser alto.
- Cada chamada recursiva irá empilhar dados na pilha de execução do programa (Stack) e, caso esta pilha não dimensionada corretamente, podemos ter um **“estouro de pilha”**.
- Além disto, o processo de empilhamento pode ter um impacto significativo no desempenho do programa.

Recursividade

Observações importantes:

1. Todo algoritmo **recursivo** tem uma versão **iterativa**, mas nem todo algoritmo **iterativo** tem uma versão **recursiva**
2. A versão iterativa de uma algoritmo é sempre mais rápida que sua versão recursiva.
3. A complexidade de um algoritmo iterativo é a mesma que sua versão recursiva.

Busca binária

Considere o algoritmo de busca binária iterativa:

```
public int BuscaBinaria(v[], int x){
            ini = 0;          //inicio do vetor
            fim = v.lenght-1; //fim do vetor
            while (ini <= fim) {
                  meio = (ini + fim)/2; //divisao
inteira
                  if (v[meio] == x) {
                        return meio;
                  } else {
                  if (v[meio] < x) ini = meio + 1;
                        else  fim = meio - 1;
                  } return -1;
}
```

Busca binária recursiva

Para implementarmos a busca binária recursiva será necessário **generalizar** ligeiramente o problema, trocando v[0..n-1] por v[ini..fim].

Assim teríamos que a função recebe um número x e um vetor em ordem crescente v[ini..fim]. Ele devolve um índice m tal que v[m] == x ou devolve -1 se tal m não existe.

A declaração da função ficaria:

```
public int BuscaBinaria(int v[],int x,int ini,int fim );
```

Busca binária recursiva

```
public int BuscaBinaria( int v[], int x, int ini,int fim ){
    //base da recursao
              if( ini > fim )
        return -1;

    int meio = (ini + fim)/2;
    if (x == v[meio])
        return meio;

    if (x < v[meio])
        return BuscaBinaria(v, x, ini, meio-1);
    else
        return BuscaBinaria(v, x, meio+1, fim);
}
```

Exercícios

01) Considere a seguinte função abaixo:

```
public int result( int n ){
   if (n == 1)
      return 2;
   else
      return 2 * result(n - 1);
}
```

Qual será o valor retornado com a execução de result(5)?

Exercícios

02) Qual a "**profundidade da recursão**" da função que realiza a busca binária recursiva ?

Ou seja, quantas vezes **buscaBinaria()** chama a si mesma?

Exercícios

03) Os números de ***Fibonacci*** correspondem à seguinte sequência de números naturais:

0 1 1 2 3 5 8 13 ...

A regra de formação desta sequência é muito simples: exceto para os dois números iniciais (0 e 1), todos os outros números são a soma dos números imediatamente anteriores na sequência.

a) Proponha uma regra recursiva para definir o n-ésimo número de *Fibonacci*.

b) Implemente uma função recursiva que calcule o n-ésimo número de *Fibonacci* e a teste.

c) Faça um diagrama de execução das chamadas recursivas de sua implementação para calcular o 5-ésimo número de *Fibonacci*.

Exercícios

04) Escreva uma função recursiva que calcula o **produto de *a* * *b***, em que *a* e *b* são inteiros maiores que zero. considere que o produto pode ser definido como *a* somado a si mesmo *b* vezes, usando uma definição recursiva temos

$a * b = a$ se $b = 1$

$a * b = a * (b - 1) + a$ se $b > 1$

Exercícios

05) Implemente uma função recursiva para calcular a **potência** a^n, supondo que tanto a quanto n sejam números inteiros positivos.

06) Implemente uma função recursiva para calcular a **soma dos dígitos** de um número inteiro e positivo.

07) Implemente uma função recursiva para verificar se um determinado número natural, maior ou igual a 2, é **primo**.

Exercícios

08) Implemente uma função recursiva para calcular o **resto da divisão** inteira de X por Y, supondo X e Y números inteiros.

09) Implemente uma função recursiva para calcular o **quociente da divisão** inteira de X por Y, supondo X e Y números inteiros.

10) Implemente uma função iterativa e outra função recursiva que receba um número inteiro positivo na base decimal e o converta para a **base binária.**

11) Implemente uma função iterativa e outra função recursiva que receba um número inteiro positivo na base binária e o **converta para a base decimal.**

Exercícios

12) Dada uma **sequência de números inteiros positivos**, descreva uma função recursiva para encontrar:

a) Busca linear de um elemento da sequência;

b) Menor elemento da sequência;

c) A soma dos elementos da sequência

c) A média aritmética dos elementos da sequência

Considere que as sequência informada para as funções não é vazia, ou seja, se n >= 1.

Exercícios

13) Qual o valor de retorno da função a seguir, caso n=27 ?

```
public int recursao( n ){
    if (n <= 10)
            return n * 2;
    else
            return recursao(recursao(n/3));
}
```