INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
SUL-RIO-GRANDENSE
Campus Passo Fundo

ALGORITMOS II

Prof. Adilso Nunes de Souza

INTRODUÇÃO

- **Recursividade ou recursão** ocorre quando uma função invoca (chama) a si mesma.
- Ao construir uma função recursiva a primeira providência é identificar um ponto de parada na recursividade, evitando que a função entre em um *loop* infinito.

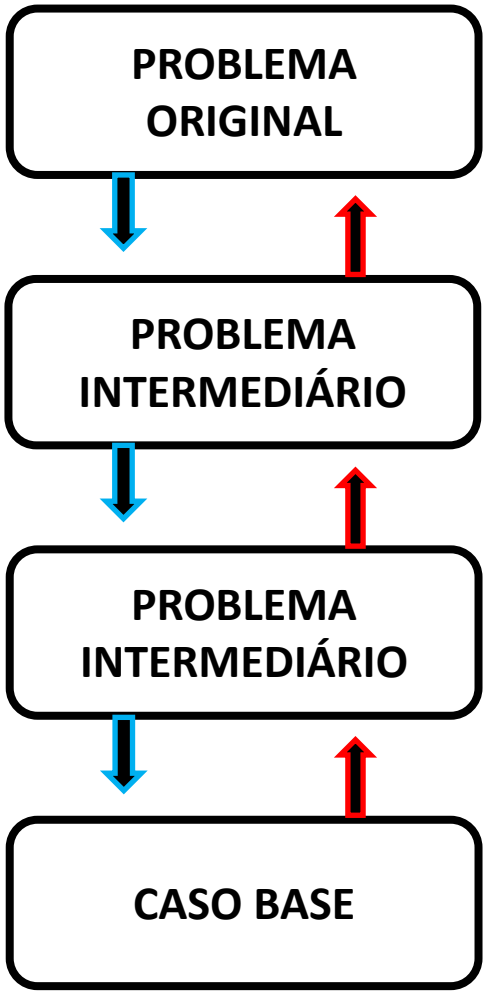
FUNCIONALIDADE

- A função começa a execução do seu primeiro comando cada vez que é chamada;
- Novas e distintas cópias dos parâmetros passados por valor e variáveis locais são criadas;
- A posição que chama a função é colocada em estado de espera, enquanto que o nível gerado recursivamente esteja executando.

FUNCIONALIDADE

- As novas chamadas da função devem ser semelhantes a chamada original, ou seja, o problema deve ter um padrão, a cada chamada o problema fica mais próximo da solução.
- O passo recursivo é executado novamente sobre uma versão mais simples do problema, até chegar no caso base, quando então o problema é resolvido.

FUNCIONALIDADE



EXEMPLO

- Imagine uma rotina que parte de uma valor N qualquer e deverá somar com seus antecessores até N = 1
- Se n=5, essa função deve retornar:
  soma(5) = 5 + 4 + 3 + 2 + 1
- Se n=4, essa função deve retornar:
  soma(4) = 4 + 3 + 2 + 1
- Se n=3, essa função deve retornar:
  soma(3) = 3 + 2 + 1
- E assim sucessivamente.

EXEMPLO

- Para fazermos uso da recursividade, temos que notar padrões.
- Note que:

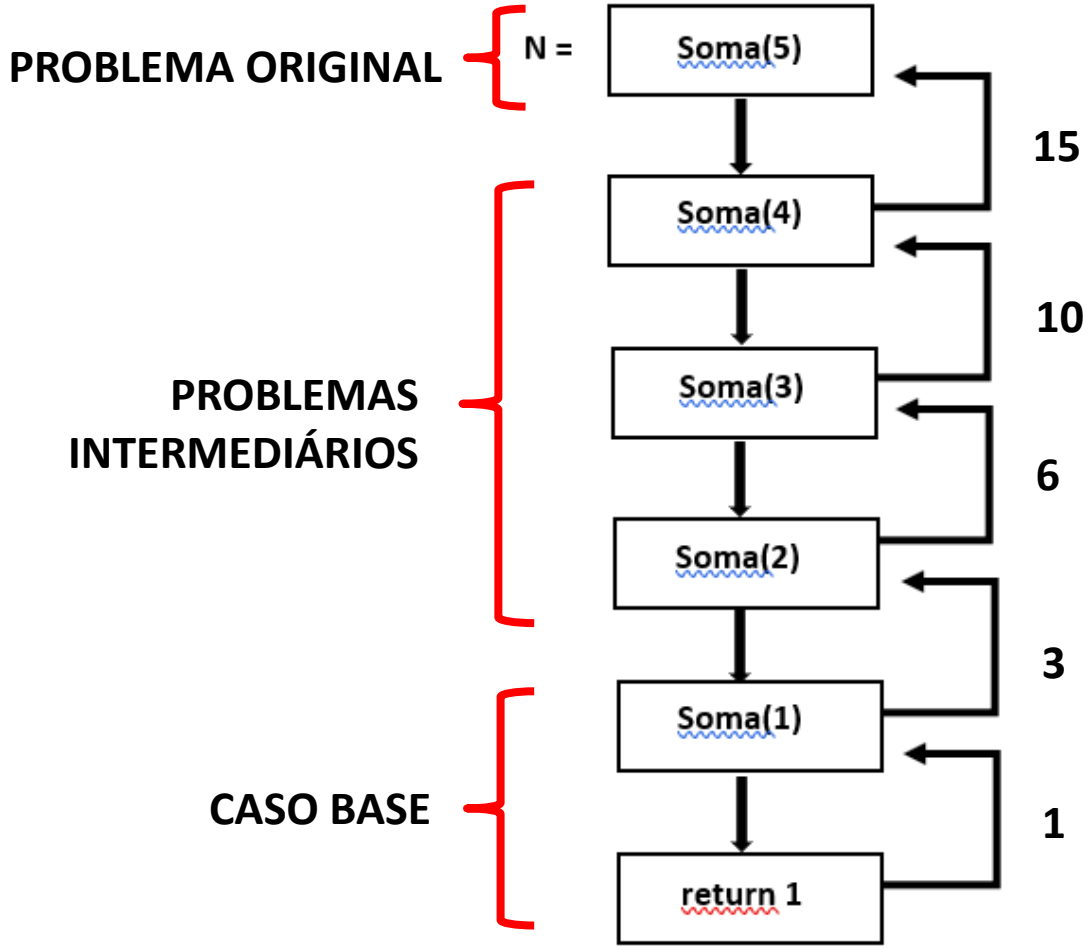
  soma(5) = 5 + 4 + 3 + 2 +1

  é igual a: 5 + soma(4)
- Da mesma forma

  soma(4) = 4 + 3 + 2 + 1

  é igual a: 4 + soma(3)
- Seguindo este padrão pode-se afirmar que as rotinas se repetem (o mesmo algoritmo é aplicado em todos os problemas) podendo resolver através da recursividade com a fórmula: soma(n) = n + soma(n-1)

EXEMPLO


PROBLEMA ORIGINAL
N =
Soma(5)
15
Soma(4)
10
PROBLEMAS INTERMEDIÁRIOS
Soma(3)
6
Soma(2)
3
Soma(1)
CASO BASE
1
return 1

EXEMPLO

```
int soma(int n);
main()
{
  int v;
  cout << "Informe um valor qualquer: ";
  cin >> v;
  cout <<  soma(v);
}

int soma(int n)
{
   if(n == 1)
      return 1;
   else
      return (n + soma(n-1));
}
```

FINAL DA RECURSÃO

- Resolvido o caso base, seu resultado é retornado a função chamadora (que estava em espera) com este resultado é possível resolver o problema nesta etapa e retornar para a etapa anterior e assim sucessivamente até chegar na chamada original.
- A função termina quando a chamada original da função receber o resultado dos passo recursivos gerados.

VANTAGENS

- **Vantagens da recursividade**
  - Diminui o número de variáveis necessárias.
  - Código mais simples e elegante, tornando-o fácil de entender e de manter.
  - Um aprimoramento considerável das capacidades de análise de problemas e construções de solução
  - Melhor entendimento de como um fluxo de execução de um programa se comporta

DESVANTAGENS

- **Desvantagens da recursividade**
  - Quando o loop recursivo é muito grande é consumida muita memória nas chamadas a diversos níveis de recursão, não sendo muito eficiente em termos de performance.

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
SUL-RIO-GRANDENSE
Campus Passo Fundo

MATERIAL DE APOIO

Série de Fibonacci:
0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, 1597, 2584, 4181, 6765, ...

https://www.youtube.com/watch?v=QaWepnGWRs8&feature=related

https://www.youtube.com/watch?v=M7c-m2xN9FQ

REFERÊNCIAS

- ARAÚJO, Jáiro – Dominando a Linguagem C. Editora Ciência Moderna.
- PEREIRA, Silvio do Lago. Estrutura de Dados Fundamentais: Conceitos e Aplicações, 12. Ed. São Paulo, Érica, 2008.
- LORENZI, Fabiana. MATTOS, Patrícia Noll de. CARVALHO, Tanisi Pereira de. Estrutura de Dados. São Paulo: Ed. Thomson Learning, 2007.
- VELOSO, Paulo. SANTOS, Celso dos. AZEVEDO, Paulo. FURTADO, Antonio. Estrutura de dados. Rio de Janeiro: Ed. Elsevier, 1983 27ª reimpressão.