Cálculo Fatorial Usando Programação Recursiva

Índice de Conteúdo

Pseudo-Código:

Função calcularFatorial(n):

```
Função calcularFatorial(n):
    Se n for igual a 0 ou 1:
        Retornar 1 // Condição de parada
    Caso contrário:
```

```
        Retornar n * calcularFatorial(n - 1)     // Chamada recursiva
```

Estrutura principal:

```
Início
    Ler                                 número                                  n
    Chamar                                                          calcularFatorial(n)
    Exibir                              o                                   resultado
Fim
```

Explicação:

Entrada: O programa começa lendo o número `n`, que será usado para calcular o fatorial.

Condição de Parada: Se `n` é 0 ou 1, o fatorial é 1. Esta é a condição de parada da função recursiva, garantindo que a recursão não continue indefinidamente.

Recursão: Para qualquer valor de `n` maior que 1, a função chama a si mesma com o argumento `n - 1`, multiplicando `n` pelo resultado da chamada recursiva. Esse processo continua até que `n` atinja 1, acumulando o produto dos números de 1 a `n`.

Saída: O resultado final, que é o fatorial de `n`, é exibido no console.

Funcionamento do Código:

Para a entrada `n = 7`, o código realiza o cálculo da seguinte forma:

`calcularFatorial(7)` chama `calcularFatorial(6)`, e assim por diante.

O cálculo se desenvolve como: `7 * 6 * 5 * 4 * 3 * 2 * 1 = 5040`.

O resultado final, `5040`, é exibido como "Fatorial de 7 é: 5040".

Manipulação de Números Muito Grandes:

Para calcular o fatorial de números muito grandes, como 100, a classe `BigInteger` é utilizada, pois ela suporta operações com números inteiros de tamanho arbitrário.

Problema: A classe `BigInteger` não possui suporte direto para formatação em notação científica, e a conversão para `double` pode levar à perda de precisão para números extremamente grandes.

Solução Proposta com Notação Científica:

Para exibir números grandes em notação científica e em formato decimal completo:

Função formatarResultadoFatorial:

```
Função formatarResultadoFatorial(resultado):
    Contagem de Dígitos: Verifica o comprimento do número em formato decimal.
    Notação de Potência de 10: Se o número de dígitos excede o limite definido (DIGITOS_MAXIMOS), a função calcula o expoente (número total de dígitos menos 1) e cria uma string no formato 10^expoente.
    Formatação Decimal: Adiciona a versão completa do número em formato decimal abaixo da notação de potência de 10.
```

Exibição:

Mostra o número em notação de potência de 10 seguido pela versão decimal completa do número.

Observações:

Limitação de Precisão: `BigInteger` é adequado para cálculos com números grandes, mas a conversão para `double` pode não ser precisa para números extremamente grandes. Por isso, a notação científica foi calculada manualmente.

Ajuste de DIGITOS_MAXIMOS: Ajuste o valor de `DIGITOS_MAXIMOS` conforme necessário para determinar o ponto em que a notação de potência de 10 deve ser usada.

Exemplo:

Para `n = 100`:

O fatorial é um número muito grande.

A notação de potência de 10 será usada para exibir o número de forma mais compacta.

O valor decimal completo será mostrado logo abaixo.

Análise Assintótica do Algoritmo Fatorial Recursivo

Algoritmo Recursivo para Cálculo Fatorial

```
Função calcularFatorial(n):
    Se n for igual a 0 ou 1:
        Retornar 1 // Condição de parada
    Caso contrário:
        Retornar n * calcularFatorial(n - 1) // Chamada recursiva
```

Análise Assintótica

Algoritmo Recursivo para Cálculo Fatorial

Complexidade Temporal: A função recursiva `calcularFatorial(n)` faz chamadas recursivas para calcular o fatorial de `n-1`, `n-2`, até 1. Cada chamada recursiva realiza uma multiplicação e

uma chamada adicional. O número total de chamadas recursivas é `n`, resultando em uma complexidade de tempo de `O(n)`.

Complexidade Espacial: Cada chamada recursiva adiciona um novo frame na pilha de execução (stack), utilizado para armazenar o estado atual da função. O número de frames é igual ao número de chamadas recursivas, que é `n`. Portanto, a complexidade espacial do algoritmo recursivo é `O(n)`, pois cada chamada recursiva ocupa espaço na pilha.

Algoritmo Iterativo para Cálculo Fatorial

Complexidade Temporal: O loop iterativo executa `n` iterações para calcular o fatorial. A complexidade temporal do método iterativo é `O(n)`.

Complexidade Espacial: O método iterativo utiliza uma quantidade constante de memória adicional (variáveis e loop), independentemente do valor de `n`. Assim, a complexidade espacial é `O(1)`.

Comparação entre Abordagens

- **Recursiva:** `O(n)` para tempo e `O(n)` para espaço.
- **Iterativa:** `O(n)` para tempo e `O(1)` para espaço.

A abordagem iterativa é mais eficiente em termos de uso de memória, pois não requer a pilha de chamadas recursivas, enquanto a abordagem recursiva pode ser mais intuitiva para alguns problemas. O uso excessivo de recursão pode levar a um estouro de pilha (stack overflow) para valores muito grandes de `n`.

Aplicação Prática

Para valores grandes de `n`, a abordagem iterativa pode ser preferível para evitar problemas de pilha. Para o cálculo de fatorial de números muito grandes (como 100!), o uso de `BigInteger` é necessário, e a notação científica pode ser usada para exibir resultados extensos.