Fazendo Tetração

Mário Leite

...

Existem muitas operações matemáticas que podem parecer bem estranhas para o iniciante em programação; a exponenciação pode ser um exemplo. Entretanto, existe uma operação que excede as nossas capacidades de pensar sobre o assunto: a **Tetração**, que é uma operação de *superexponencial*. Então, se exponenciação já é um tanto esquisita, embora tenha uma aplicação muito prática para substituir repetições de multiplicações. Por outro lado, a Tetração é bem esquisita, pois é uma operação matemática que cresce muito rapidamente e é mais teórica do que prática na maioria das aplicações; no entanto, tem algumas aplicações e implicações em várias áreas das ciências e na matemática avançada. Aqui estão algumas áreas onde essa operação tem relevâncias:

1. Teoria dos Números

Na teoria dos números a Tetração pode ser usada para estudar funções que crescem rapidamente. Ela ajuda a entender e descrever o comportamento de certas funções e sequências matemáticas que têm crescimento superexponencial.

2. Teoria Computacional

Em ciência da computação, especialmente na teoria da complexidade pode ser usada para descrever funções que crescem muito rapidamente, além do que exponenciais podem descrever. Por exemplo, a Função de Ackermann, que cresce muito rapidamente, pode ser expressa em termos de Tetração.

3. Matemática Discreta e Lógica

Em matemática discreta, especialmente em problemas envolvendo combinatória e lógica matemática pode ser usada para explorar conceitos de crescimento e limites. Embora não seja usada diretamente em problemas práticos, fornece uma base para entender padrões de crescimento extremo.

4. Teorias de Funções e Sistemas Dinâmicos

Na teoria de funções e sistemas dinâmicos pode ser usada para estudar o comportamento de sistemas que possuem um crescimento muito rápido. Isso pode ser relevante em modelos matemáticos que envolvem crescimento exponencial iterado.

5. Computação e Algoritmos

Em algumas áreas da computação, especialmente na análise de algoritmos, pode ser usada para descrever a complexidade de algoritmos que têm um crescimento superexponencial. Isso pode ser relevante em algoritmos para problemas difíceis ou em teoria da computação.

6. Física Teórica e Cosmologia

Embora não seja uma aplicação direta, conceitos de crescimento extremo como a Tetração podem ser usados em modelos teóricos de física e cosmologia. Por exemplo, em modelos que tentam descrever o crescimento do universo ou a complexidade de certos sistemas físicos.

Embora a Tetração não tenha muitas aplicações práticas diretas no sentido convencional, sua importância reside na teoria matemática e na compreensão de funções de crescimento rápido. É uma ferramenta útil para explorar e descrever conceitos matemáticos avançados.

O programa **"Tetração"**, codificado em C# é uma solução simples, porém mostra como essa operação pode ser feita. A **figura 1** mostra um exemplo de saída do programa com um valor-base **2** e profundidade **3**; pode ser até interpretado como o $\mathbf{2*2^3}$, mas é bem mais complexo do que isto.

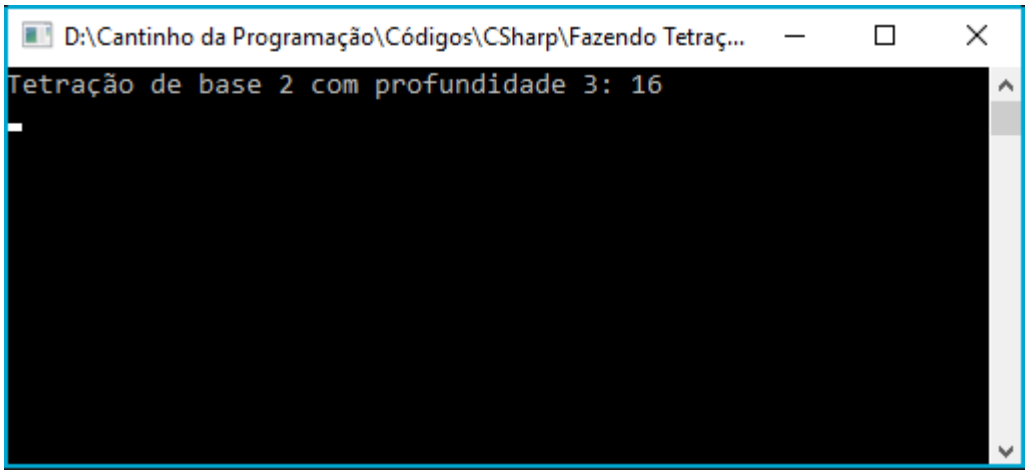

Figura 1 - Um exemplo da operação de Tetração

==========

```
using System;
using System.Threading.Tasks;
using System.Numerics;

 namespace Tetração
 {
     class Program
     {

         // Função para calcular tetração
          static BigInteger Tetracao(BigInteger valorBase, int profundidade)
          {
            if (profundidade <= 0)
                 throw new ArgumentException("Profundidade deve ser inteiro
                 positivo!");

              if (valorBase == 0)
                  return 0;

              if (valorBase == 1 || profundidade == 1)
                  return valorBase;

              BigInteger resultado = valorBase;

              Fo r (int i = 1; i < profundidade; i++)
              {
                 resultado = Power(valorBase, resultado);
                 if (resultado <= 0)
                      throw new OverflowException("Resultado é grande demais!");
              }

              return resultado;
          }
```

```
        // Função auxiliar para calcular a potência de BigInteger
        static BigInteger Power(BigInteger valorBase, BigInteger exponente)
        {
            BigInteger resultado = 1;
            for (BigInteger i = 0; i < exponente; i++)
            {
                try  {
                    resultado = BigInteger.Multiply(resultado, valorBase);
                }
                catch (OverflowException)
                {
                    throw new OverflowException("Resultado é muito grande!");
                }
            }
            return resultado;
        }

        static void Main()
        {
            BigInteger valorBase = 2; // Base da tetração
            int profundidade=3; //Profundidade da tetração (para evitar estouro)

            try
            {
                BigInteger resultado = Tetracao(valorBase, profundidade);
                Console.WriteLine($"Tetração de base {valorBase} com
                profundidade {profundidade}: {resultado}");
            }
            catch (Exception ex)
            {
                Console.WriteLine($"Erro: {ex.Message}");
            }
            Console.ReadKey();
        }

    }
} // fim do programa “Tetração”
```