UNIVERSIDADE FEDERAL DE MINAS GERAIS

DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO

**Trabalho Prático: Fractais**

Documentação

Júlio Guerra Domingues

2022431280

Belo Horizonte – MG

Junho/2023

Sumário

[Introdução 3](#_Toc137253266)

[Especificação, projeto e implementação 3](#_Toc137253267)

[Especificação 3](#_Toc137253268)

[Estratégias para implementação dos fractais 3](#_Toc137253269)

[Projeto e Implementação 4](#_Toc137253270)

[Equação de recorrência 4](#_Toc137253271)

[Complexidade 5](#_Toc137253272)

[Desenho dos fractais 6](#_Toc137253273)

[Anexo 1 7](#_Toc137253274)

[Fractal (i): Floco de neve de onda quadrada de von Koch 7](#_Toc137253275)

[Fractal (ii): Preenchimento de espaço de Hilbert 9](#_Toc137253276)

[Fractal (iii): Fractal original 11](#_Toc137253277)

## Introdução

Fractais são estruturas geométricas complexas cujas propriedades, em geral, repetem-se em qualquer escala. O presente trabalho prático define alguns fractais que devem ser construídos. Trata-se de requisito parcial da disciplina Matemática Discreta, do 2º período do curso de Ciência da Computação da Universidade Federal de Minas Gerais.

## Especificação, projeto e implementação

### Especificação

De acordo com as instruções fornecidas, foram definidos os seguintes fractais:

1. Floco de neve de onda quadrada de von Koch
   1. Soma dos algarismos do número de matrícula (=24) módulo 4 = 0
   2. Regras:
      1. Axioma: F
      2. Ângulo: 90 graus
      3. F = F-F+F+FF-F-F+F
2. Preenchimento de espaço de Hilbert
   1. Número de matrícula par
   2. Regras:
      1. Axioma: X
      2. Ângulo: 90 graus
      3. X = -YF+XFX+FY-
      4. Y = +XF-YFY-FX+
3. Fractal definido pelo aluno, que gera cadeia de polígonos simples com pelo menos duas regras
   1. Foi criado um fractal, original (pelo que foi pesquisado), a partir de variações nas regras do fractal de preenchimento de espaço de Hilbert
   2. Foram utilizadas as seguintes regras:
      1. Axioma: X
      2. Ângulo: 90 graus
      3. X = YF+XFYFX+FY
      4. Y = XF-YFXFY-FX

O código que gera as instruções para construção dos fractais deve ser escrito em linguagem de programação C. Apesar de a orientação inicial ser a de fazer programas individuais para cada fractal, o arquivo de entradas para o TP (fornecido pela monitora) indica que o código que realiza os três fractais deve estar contido em apenas um programa. Para o desenho de tais fractais, podem ser utilizados softwares já existentes ou, ainda, códigos em outras linguagens.

O prazo para realização desse trabalho prático é 10/06/2023.

### Estratégias para implementação dos fractais

Para a implementação dos fractais, tal qual em diversos outros problemas computacionais, mais de uma abordagem por ser utilizada. Basicamente, as principais abordagens são as iterativas e as recursivas.

Uma forma seria utilizando iteratividade, em que os caracteres de um estágio intermediário são gravados (na memória ou em um arquivo) e, após, lidos e processados, para gerar o estágio subsequente. Tal processo se repete até que se alcance o estágio final. Algoritmos iterativos podem ser mais eficientes em termos de uso de memória e velocidade de execução, apresentando menores riscos de ocasionar problemas como o Stack overflow. Por outro lado, em alguns casos, como ocorre na construção de fractais, as soluções iterativas são mais complexas e com pior legibilidade, sendo mais difíceis de entender e mais susceptíveis a erros de lógica.

Algoritmos recursivos, por sua vez, especialmente em problemas de natureza recursiva – que podem ser divididos em subproblemas menores, como os fractais – são mais facilmente compreensíveis, simples e intuitivos. Tal abordagem usualmente consome mais memória e processamento quando comparada a alternativas iterativas.

Em resumo, podemos notar um *trade-off* entre legibilidade e eficiência entre abordagens recursivas e iterativas. Especificamente para o problema dos fractais proposto, a abordagem recursiva parece mais natural, sendo de fato a mais utilizada nas fontes de referência utilizadas. Os fractais são matematicamente descritos de forma recursiva, o que faz com que, ao utilizarmos a mesma abordagem, o código fique mais compreensível, sendo a sua implementação mais direta e com menor risco de introdução de erros lógicos.

### Projeto e Implementação

O trabalho foi desenvolvido em linguagem C, utilizando o editor de código Visual Studio Code em ambiente Windows, e compilador MSVC 19.36. Optou-se pela implementação por meio de abordagem recursiva, como explicado acima. Conforme orientações, foi criado um único código que recebe a entrada em formato pré-estabelecido e gera um arquivo de texto (em formato .txt) que armazena as instruções para o desenho do fractal. Foram realizados testes até o quarto nível de recursão.

O TP foi realizado com base nas instruções fornecidas no arquivo inicial disponibilizado pelo professor via Moodle, nas orientações sobre entradas e saídas disponibilizadas pela monitora e nas discussões do fórum. Para dirimir eventuais dúvidas, o arquivo “leiame.txt” apresenta informações detalhadas sobre o ambiente computacional utilizado e instruções para a execução do programa. Caso haja problemas para rodar os testes, uma versão online do programa pode ser acessada em: <https://onlinegdb.com/1l4ifV_3y> (gcc 11.3.0 -std=gnu99).

## Equação de recorrência

A partir do arquivo “fractal\_count.txt”, gerado ao rodar os fractais, foram verificadas as seguintes quantidades para cada um dos símbolos impressos, em cada fractal:

(i)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 |
| F | 8 | 64 | 512 | 4096 |
| X | 0 | 0 | 0 | 0 |
| Y | 0 | 0 | 0 | 0 |
| + | 3 | 27 | 219 | 1755 |
| - | 3 | 27 | 219 | 1755 |
| Total | 14 | 118 | 950 | 7606 |

Para o caso base, a equação de recorrência é: T(0) = 1

Para os passos recursivos, a equação é: T(n) = 8\*T(n-1)+6

(ii)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 |
| F | 3 | 15 | 63 | 255 |
| X | 2 | 8 | 32 | 128 |
| Y | 2 | 8 | 32 | 128 |
| + | 2 | 10 | 42 | 170 |
| - | 2 | 10 | 42 | 170 |
| Total | 11 | 51 | 211 | 851 |

Para o caso base, a equação de recorrência é: T(0) = 1

Para os passos recursivos, a equação de recorrência é: T(n) = 4\*T(n-1) + 7

Desconsiderando o F, a equação de recorrência passa a ser: T(n) = 4\*T(n-1) + 4

(iii)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 |
| F | 12 | 72 | 372 | 1872 |
| X | 2 | 13 | 62 | 313 |
| Y | 3 | 12 | 63 | 312 |
| + | 2 | 6 | 32 | 156 |
| - | 0 | 6 | 30 | 156 |
| Total | 19 | 109 | 559 | 2809 |

Para o caso base, a equação de recorrência é: T(0) = 1

Para os passos recursivos, a equação de recorrência é: T(n) = 5\*T(n-1) + 14

Desconsiderando o F, a equação de recorrência passa a ser: T(n) = 5\*T(n-1) + 2

## Complexidade

De forma simplificada, podemos estimar a complexidade dos algoritmos a partir das equações de recorrência calculadas no item anterior. Cabe ressaltar que essa é uma estimativa limitada, que desconsidera custos de diversas partes do programa e de eventuais bibliotecas utilizadas.

Fractal (i):

Podemos expandir a equação de recorrência:

T(n) = 8 \* T(n-1) + 6

= 8 \* (8 \* T(n-2) + 6) + 6 = 8^2 \* T(n-2) + 8 \* 6 + 6

= 8^2 \* (8 \* T(n-3) + 6) + 8 \* 6 + 6 = 8^3 \* T(n-3) + 8^2 \* 6 + 8 \* 6 + 6

...

O termo 8^k \* T(n-k) se repete em cada expressão, sendo adicionado um termo constante a cada passo até o caso base

Generalizando a equação: T(n) = 8^k \* T(n-k) + 6 \* (8^0 + 8^1 + ... + 8^(k-1))

Assim a soma da série geométrica 8^0 + 8^1 + ... + 8^(k-1) pode ser calculada como: 8^0 + 8^1 + ... + 8^(k-1) = (8^k - 1) / (8 - 1) 🡪 8^0 + 8^1 + ... + 8^(k-1) = (8^k - 1) / 7

Substituindo de volta na equação, temos T(n) = 8^k \* T(n-k) + 6 \* ((8^k - 1) / 7)

Para o caso base T(0) 🡪 n – k = 0 🡪 k = n

T(n) = 8^n \* T(0) + 6 \* ((8^n - 1) / 7) = 8^n + 6 \* ((8^n - 1) / 7)

Analisando a complexidade da equação de recorrência do fractal (i), temos que se trata um padrão exponencial, mais especificamente Θ(8^n).

Fractal (ii):

De forma semelhante à que fizemos com o primeiro fractal, podemos estimar a complexidade em Θ(4^n), para ambas as equações de recorrência estimadas (com e sem os símbolos F).

Fractal (iii):

De forma semelhante aos casos anteriores, podemos estimar a complexidade em Θ(5^n), para ambas as equações de recorrência estimadas (com e sem os símbolos F).

## Desenho dos fractais

Existem inúmeras opções viáveis para o desenho dos fractais a partir dos arquivos .txt gerados. Em termos gerais, é necessário que os símbolos ‘F’ sejam lidos como “linhas” e os símbolos ‘+’ e ‘-‘ como mudanças nas angulações. Como os fractais são diferentes dependendo do ângulo definido no axioma, tal informação deve ser também fornecida ao programa que fará o desenho do fractal. A seguir foram elencadas algumas opções para o desenho dos fractais a partir de arquivos texto, todas aplicáveis em Python mas também com opções para outras linguagens de programação.

O desenho pode ser dar de forma dinâmica, o que pode ser mais interessante em exibições ao vivo. Para tal, a biblioteca *turtle* é uma das mais recomendadas. Ela permite a definição do tamanho, espessura e cor da linha, replicando dinamicamente os comandos de um arquivo texto.

Existem outras bibliotecas utilizadas para a exibição de gráficos bidimensionais estáticos, como SDL2 e matplotlib. Apesar de não exibir em tempo real o desenho da imagem, elas também oferecem oferece uma ampla gama de opções de personalização, incluindo controle sobre cores, tamanhos e estilos de linha, permitindo criar imagens estáticas de fractais. Para o presente trabalho, optou-se pela utilização da biblioteca *matplotlib.*

As imagens resultantes das recursões de nível 1 a 4 dos fractais (i), (ii) e (iii) estão, respectivamente, no anexo 1 ao fim desse documento.

## Anexo 1

### Fractal (i): Floco de neve de onda quadrada de von Koch

#### Nível de recursão: 1

Forma

Descrição gerada automaticamente

#### Nível de recursão: 2

Uma imagem contendo Diagrama

Descrição gerada automaticamente

#### Nível de recursão: 3

Texto

Descrição gerada automaticamente

#### Nível de recursão: 4

Código QR

Descrição gerada automaticamente

### Fractal (ii): Preenchimento de espaço de Hilbert

#### Nível de recursão: 1

Forma, Quadrado

Descrição gerada automaticamente

#### Nível de recursão: 2

Forma

Descrição gerada automaticamente

#### Nível de recursão: 3

Código QR

Descrição gerada automaticamente

#### Nível de recursão: 4

Desenho em preto e branco

Descrição gerada automaticamente com confiança baixa

### Fractal (iii): Fractal original

#### Nível de recursão: 1

Forma, Quadrado

Descrição gerada automaticamente

#### Nível de recursão: 2

Forma, Seta

Descrição gerada automaticamente

#### Nível de recursão: 3

Diagrama

Descrição gerada automaticamente

#### Nível de recursão: 4

Texto, Esquemático

Descrição gerada automaticamente