Matemática Discreta - TW
Universidade Federal de Minas Gerais - UFMG
Belo Horizonte - 31/05/2023

Trabalho Prático - Fractal

Guilherme Mota Bromonschenkel Lima - 2019027571

Introdução

Um fractal é uma forma ou padrão geométrico com semelhança em diferentes escalas ou ampliações. Ou seja, quando um fractal é ampliado, é possível ver os mesmos padrões repetidos em escalas cada vez menores, isso mostra a característica recursiva da estrutura de um fractal. Vale ressaltar que os fractais são encontrados em muitos objetos naturais e artificiais *(como flocos de neve)* e também têm sido usados em diversas áreas, como matemática, física, etc.

Especificação

Esse projeto consiste na implementação de algoritmos que construam 3 tipos de fractais. Esses tipos são definidos a partir das regras propostas no enunciado do trabalho prático.

I. Floco de neve onda senoidal 2 de von Koch

Nº de matrícula = 2019027571 / Soma dos algarismos = 2+0+1+9+0+2+7+5+7+1 = 34
Resto da divisão por 4 = 34 % 4 = 2

Axioma:	F
	$\theta = \frac{\pi}{3}$
Regras:	F → F-F+F+FF-F-F+F

II. Preenchimento de espaço de Peano

Nº de matrícula = 2019027571
Par ou Ímpar = Ímpar

Axioma:	X
	$\theta = \frac{\pi}{2}$
Regras:	X → XFYFX+F+YFXFY-F-XFYFX
	Y → YFXFY-F-XFYFX+F+YFXFY

III. Fractal de criação própria

Um fractal que gere uma cadeia de polígonos simples que tenha pelo menos duas regras como as curvas de preenchimento de espaço de Peano e Hilbert.

Axioma:	X
	$\theta = \frac{\pi}{2}$
Regras:	X → F+FY+X
	Y → -F+FY

Projeto

Nesse projeto foi utilizada a linguagem de programação C para criar o algoritmo que fizesse a construção de fractais usando um L-sistema *(axioma, valor de θ, regras de produção e estágios)*.

A aplicação consiste de alguns módulos: app *(ponto de entrada para a execução da aplicação como um todo)*; fractal *(módulo responsável pelas regras de negócio de expansão da sequência do fractal)*; fractal-file *(módulo responsável pela organização dos nomes dos arquivos usados na geração dos fractais)*; shared-utils *(módulo com ferramentas reutilizáveis pelos outros módulos da aplicação)*.

Para que o programa funcione, é necessário especificar a quantidade de estágios, o axioma e por fim quais regras serão utilizadas. Durante a execução da aplicação, os fractais resultantes são criados na pasta output.

Estratégias de implementação

As estratégias de implementação comuns para o algoritmo de fractais envolvem tanto uma abordagem iterativa quanto uma abordagem recursiva.

Vale ressaltar que, quando estamos falando de algoritmos de forma geral, é normal que a abordagem recursiva tenha um custo computacional de espaço mais elevado, visto que ela guarda toda a pilha de execução em memória até que a recursão seja resolvida. Entretanto, nem todas as abordagens recursivas podem ser transformadas em abordagem iterativas que sejam simples, legíveis e escaláveis.

Ou seja, o ideal é realizar uma análise do problema que precisa ser resolvido, antes mesmo de resolvê-lo, para entender qual a melhor abordagem que podemos usar para ter o algoritmo mais legível, performático e escalável possível.

Para o caso das fractais, tanto a abordagem iterativa quanto a recursiva são legíveis. No entanto, no quesito de performance, a abordagem iterativa ganha destaque. Por fim, vale ressaltar que as duas abordagens devem ser combinadas com estratégias “inteligentes” para garantir a escala do algoritmo.

Portanto, essa implementação foi feita usando a abordagem iterativa junto de uma estratégia baseada na leitura/escrita de arquivos para o processamento sob demanda de cada estágio *(com isso, temos um algoritmo legível, performático e escalável)*.

Ou seja, durante o processamento de cada estágio, fazemos a leitura do estágio passado para gerar então o resultado do estágio seguinte. Com isso, somos capazes de processar as informações sob demanda e em pedaços pequenos, gerando um baixo consumo de memória para a nossa aplicação.

No fim, é possível notar os seguintes benefícios nessa abordagem/estratégia:

- Processamento escalável de fractais grandes *(já que processamos os dados sob demanda, evitamos um uso de memória elevado)*;
- Código mais simples e estável *(já que os dados são trabalhados em arquivos, não é necessário fazer muitas operações de gestão de memória para alocação de variáveis)*.

Equações de Recorrência

Pelo fato das funções geradoras das fractais terem uma característica recursiva, podemos usar equações de recorrência para determinar a quantidade de segmentos e símbolos presentes em cada função.

Para encontrar as equações de recorrência, foram feitas análises através dos 3 primeiros estágios de cada fractal. Com isso, foi possível perceber um comportamento padrão na quantidade de símbolos/segmentos em cada estágio de forma recursiva.

Portanto, podemos considerar:

- ➔ **Quantidade de segmentos:** $F(n)$
- ➔ **Quantidade de símbolos:** $S(n)$

(I) Floco de neve onda senoidal 2 de von Koch

Quantidade de símbolos -, +

$C(1) = 6$

$C(n) = C(n-1) + 6 * 8^{n-1}$

Quantidade de símbolos F

$F(n) = 8^n$

$F(n) = 8^n$

$S(n) = C(n) + F(n)$

$S(n) = C(n-1) + 6 * 8^{n-1} + 8^n$

(II) Preenchimento de espaço de Peano

Quantidade de símbolos -, +

$C(1) = 4$

$C(n) = C(n-1) * 9 + 4$

Quantidade de símbolos F

$F(1) = 8$

$F(n) = F(n-1) * 9 + 8$

Quantidade de símbolos Y

$Y(1) = 4$

$Y(n) = Y(n-1) * 9 + 4$

Quantidade de símbolos X

$X(1) = 5$

$X(n) = X(n-1) * 9 - 4$

$F(n) = F(n-1) * 9 + 8$

Para o caso da equação de recorrência de quantidade de símbolos, podemos ter os seguintes casos:

➔ Todos os símbolos (símbolos F, -, +, X e Y):

$S(n) = C(n) + F(n) + Y(n) + X(n)$

$S(n) = C(n-1) * 9 + 4 + F(n-1) * 9 + 8 + Y(n-1) * 9 + 4 + X(n-1) * 9 - 4$

$S(n) = C(n-1) * 9 + F(n-1) * 9 + Y(n-1) * 9 + X(n-1) * 9 + 12$

➔ Sem os símbolos F (símbolos -, +, X e Y):

$S(n) = C(n) + Y(n) + X(n)$

$S(n) = C(n-1) * 9 + 4 + Y(n-1) * 9 + 4 + X(n-1) * 9 - 4$

$S(n) = C(n-1) * 9 + Y(n-1) * 9 + X(n-1) * 9 + 4$

(III) Fractal de criação própria

Quantidade de símbolos -, +

$C(1) = 2$

$C(n) = C(n-1) + 2 * (n-1) + 2$

Quantidade de símbolos F

$F(1) = 2$

$F(n) = F(n-1) + 2 * (n-1) + 2$

Quantidade de símbolos Y

$Y(1) = 1$

$Y(n) = (n-1) + 1$

Quantidade de símbolos X

$X(n) = 1$

$F(n) = F(n-1) + 2 * (n-1) + 2$

Para o caso da equação de recorrência de quantidade de símbolos, podemos ter os seguintes casos:

➔ Todos os símbolos (símbolos F, -, +, X e Y):

$S(n) = C(n) + F(n) + Y(n) + X(n)$

$S(n) = C(n-1) + 2 * (n-1) + 2 + F(n-1) + 2 * (n-1) + 2 + (n-1) + 1 + 1$

$S(n) = C(n-1) + F(n-1) + 5n + 1$

➔ Sem os símbolos F (símbolos -, +, X e Y):

$S(n) = C(n) + Y(n) + X(n)$

$S(n) = C(n-1) + 2 * (n-1) + 2 + (n-1) + 1 + 1$

$S(n) = C(n-1) + 3n + 1$

Complexidade dos Algoritmos

Com base nas equações de recorrência mostradas anteriormente, é possível determinarmos a complexidade dos algoritmos recursivos de cada fractal:

(I) Floco de neve onda senoidal 2 de von Koch: $O(n \log n)$

(II) Preenchimento de espaço de Peano: $O(n)$

(III) Fractal de criação própria: $O(n)$

Desenho de Fractais

O desenho de fractais se trata de um desenho de linhas de acordo com o L-sistema que definimos para as fractais. Nesse sentido, alguns softwares que podem ser utilizadas para esse desenho são:

- SDL: é uma biblioteca multiplataforma usada para criar aplicativos de mídia interativa, como jogos, em C e C++. Ele fornece uma API simples e de baixo nível para acesso a recursos de hardware, como gráficos, áudio, entrada de usuário e rede. No caso dos fractais, usamos esse a parte de renderização de gráficos 2D.
- GGPlot: é uma biblioteca de visualização de dados popular e de código aberto, originalmente criada para a linguagem de programação R. Ele fornece uma abordagem declarativa para a criação de gráficos, permitindo aos usuários especificar visualizações complexas por meio de camadas estéticas e estatísticas.
- Seaborn: é uma biblioteca de visualização de dados em Python construída sobre o matplotlib. Ele fornece uma interface de alto nível para criar gráficos estatísticos atraentes e informativos. O Seaborn simplifica a criação de gráficos complexos, oferecendo uma variedade de estilos visuais predefinidos e funções fáceis de usar para aprimorar a aparência dos gráficos.

Além disso, existem também aplicações web com regras de negócio direcionadas para realizar o desenho de fractais definidas em L-sistemas *(por exemplo o Online Math Tools - L system generator)*, usando bibliotecas gráficas web nos fundos dos panos.

Abaixo é possível visualizar os quatro primeiro estágios do ***(III) Fractal de criação própria*** mostrado na Especificação deste documento:

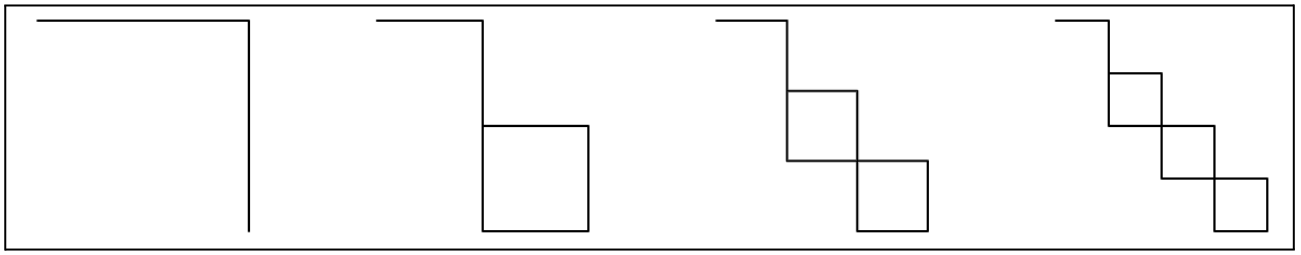

Figura 1: Quatro primeiros estágios do fractal de criação própria.