UNIVERSIDADE FEDERAL DE MINAS GERAIS
Instituto de Ciências Exatas
Departamento de Ciência da Computação
Bacharelado em Ciência da Computação

TRABALHO PRÁTICO

Nome: Mateus Mendes Alves Cabral
Disciplina: Matemática Discreta
Professor(a): Antonio Alfredo Ferreira Loureiro

Belo Horizonte
2025

Sumário

1 Introdução

Este documento trata sobre o processo e a abordagem para resolver os problemas levantados no enunciado do trabalho prático da disciplina Matemática Discreta, cursada no 1º semestre de 2025 na Universidade Federal de Minas Gerais, sob a tutoria do professor Antonio Alfredo Ferreira Loureiro.

O problema consiste na criação e no estudo e definição da complexidade de um algoritmo que calcule o n-ésimo ponto das espirais: quadrada, triangular e uma terceira espiral criada pelo estudante. O algoritmo deve ter no mínimo complexidade de $O(n)$, sendo n fornecido por meio da entrada padrão.

A documentação do trabalho conterá a descrição da metologia e a análise de cada espiral, comentários sobre a implementação do método encontrado decorrente do estudo e análise da complexidade do algoritmo e por fim a conclusão.

2 Método

A abordagem escolhida para resolver o problema foi encontrar um padrão e criar um método para construir espirais de forma progressiva, ou seja, passo a passo, em um algoritmo de complexidade $O(n)$ com n passos na espiral.

Definição 1 *Uma espiral pode ser definida em termos de ciclos de movimentos ordenados que se repetem em ordem.*

Por meio dessa noção sobre a natureza de uma espiral, vetores no plano podem ser utilizados para "desenhar" o ciclo fundamental de uma espiral e, uma vez definida a figura que será repetida em cada ciclo, basta definir a proporção dessa repetição.

O conjunto dos vetores-base e seus escalares, propostos em ordem, definem a direção e a proporção de cada passo para formar a figura fundamental da espiral.

Analisando o padrão do exemplo fornecido pelo enunciado do trabalho, infere-se a seguinte fórmula:

$$k_c = k_1 + p(c - 1) \tag{1}$$

Sendo k um escalar arbitrário e c o ciclo atual da espiral. A equação (1) expressa o aumento gradual dos escalares presentes na combinação linear dos vetores utilizados para formar uma determinada espiral. Assim, a magnitude do movimento expresso pela multiplicação vetor-escalar está definida de acordo com uma proporção de crescimento p predefinida e um ciclo c

2.1 Espiral quadrada

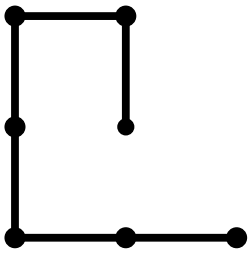

Figura 1: Figura fundamental da espiral quadrada

A direção dos movimentos, isto é, a disposição dos vetores-base, será sempre a mesma para formar cada repetição da figura fundamental. Contudo, o escalar de cada vetor terá sua proporção definida pelo ciclo atual da espiral, a fim de impedir sobreposições e dar forma à espiral.

Seja $B = \{(\vec{j}), (-\vec{i}), (-\vec{j}), (\vec{i})\}$ o conjunto de vetores-base que geram a espiral quadrada, c o ciclo da espiral, $K_c = \{d_c, e_c, f_c, g_c\}$ o conjunto de escalares do c-ésimo ciclo e p a proporção de crescimento.

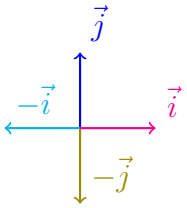

Figura 2: Vetores-base da espiral quadrada

$$\begin{aligned} \vec{j} &= (0,1) & d_1 &= 1 \\ -\vec{i} &= (-1,0) & e_1 &= 1 \\ -\vec{j} &= (0,-1) & f_1 &= 2 \\ \vec{i} &= (1,0) & g_1 &= 2 \end{aligned}$$

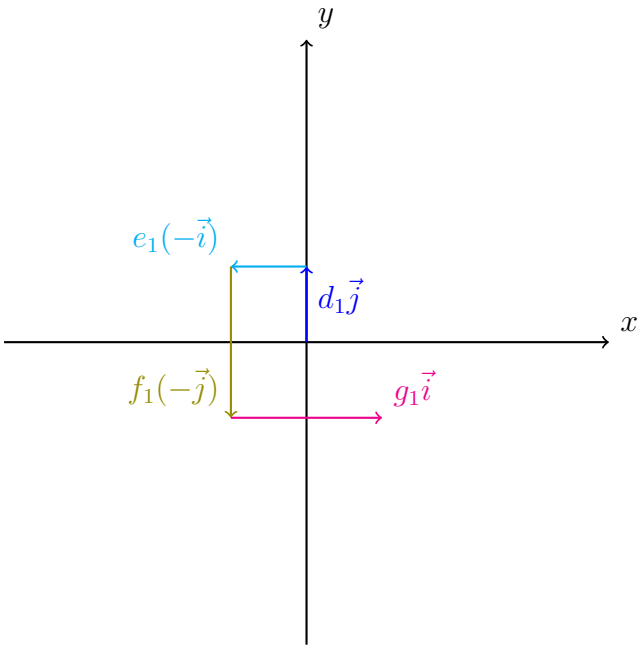

Figura 3: Espiral quadrada no ciclo $c = 1$

Aplicando a equação (1) para o ciclo $c = 2$ e proporção $p = 2$ para todos os escalares:

$$d_2 = 1 + 2(2 - 1) = 3$$
$$e_2 = 1 + 2(2 - 1) = 3$$
$$f_2 = 2 + 2(2 - 1) = 4$$
$$g_2 = 2 + 2(2 - 1) = 4$$

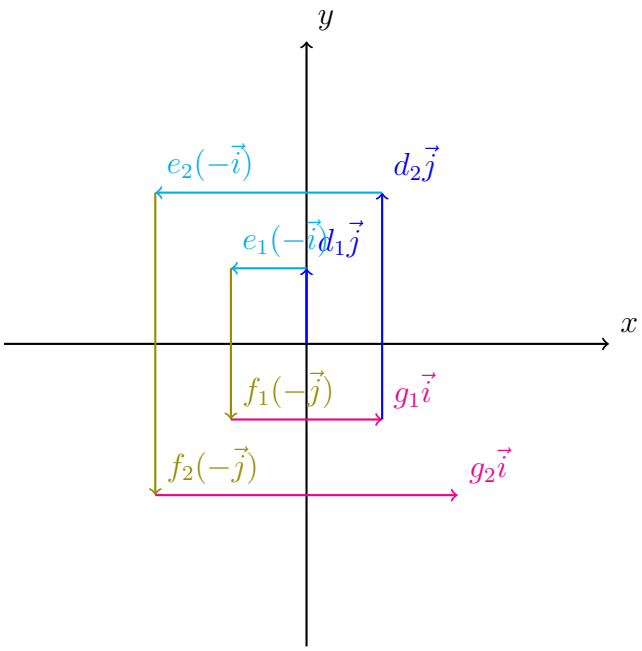

Figura 4: Espiral quadrada no ciclo $c = 2$

2.2 Espiral triangular

Seja $B = \{(\vec{i}), (-\vec{i}+\vec{j}), (-\vec{i}-\vec{j})\}$ o conjunto de vetores-base que geram a espiral triangular, c o ciclo da espiral, $K_c = \{d_c, e_c, f_c\}$ o conjunto de escalares do c-ésimo ciclo e p a proporção de crescimento.

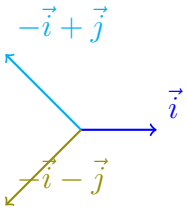

Figura 5: Vetores-base da espiral triangular

$$\begin{aligned} \vec{i} &= (1,0) & d_1 &= 1 \\ -\vec{i}+\vec{j} &= (-1,1) & e_1 &= 1 \\ -\vec{i}-\vec{j} &= (-1,-1) & f_1 &= 2 \end{aligned}$$

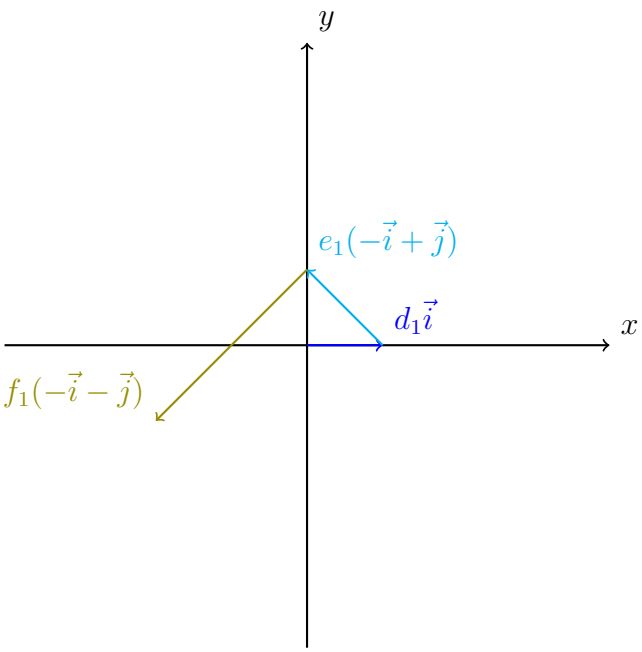

Figura 6: Espiral triangular no ciclo $c = 1$

Aplicando a equação (1) para o ciclo $c = 2$ e proporção $p = 4$ para d_c e $p = 2$ para e_c e f_c:

$$\begin{aligned} d_2 &= 1 + 4(2-1) = 5 \\ e_2 &= 1 + 2(2-1) = 3 \\ f_2 &= 2 + 2(2-1) = 4 \end{aligned}$$

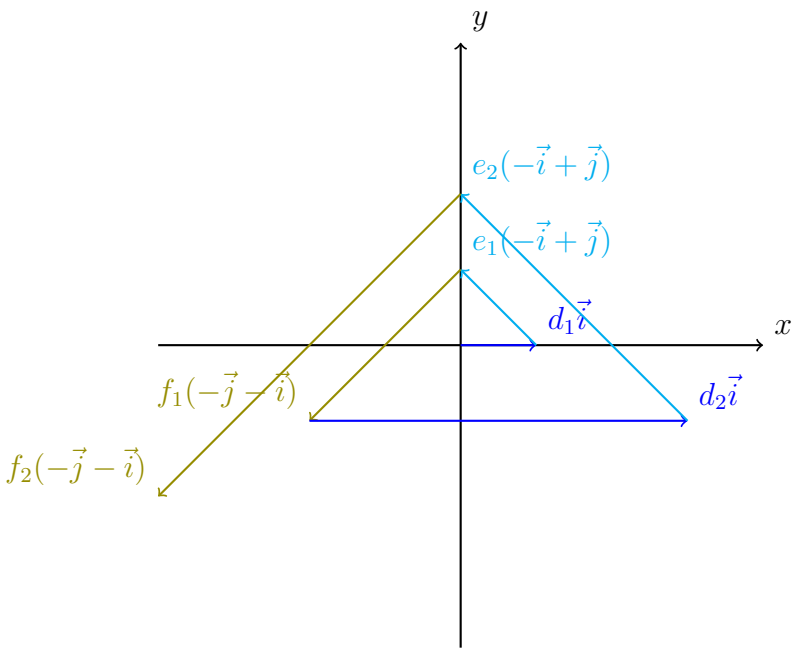

Figura 7: Espiral triangular no ciclo $c = 2$

2.3 Minha espiral

Seja $B = \{(\vec{i}), (-\vec{i}+\vec{j}), (-\vec{i}), (-\vec{i}-\vec{j}), (\vec{i}-\vec{j}), (\vec{i}+\vec{j})\}$ o conjunto de vetores-base que geram a minha espiral, c o ciclo da espiral, $K_c = \{d_c, e_c, f_c, g_c, h_c, l_c\}$ o conjunto de escalares do c-ésimo ciclo e p a proporção de crescimento.

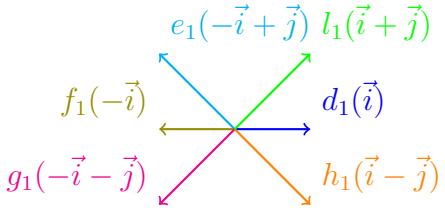

Figura 8: Vetores-base da minha espiral

$$\begin{aligned} \vec{i} &= (1,0) & d_1 &= 2 \\ -\vec{i}+\vec{j} &= (-1,1) & e_1 &= 1 \\ -\vec{i} &= (-1,0) & f_1 &= 2 \\ -\vec{i}-\vec{j} &= (-1,-1) & g_1 &= 1 \\ \vec{i}-\vec{j} &= (1,-1) & h_1 &= 2 \\ \vec{i}+\vec{j} &= (1,1) & l_1 &= 1 \end{aligned}$$

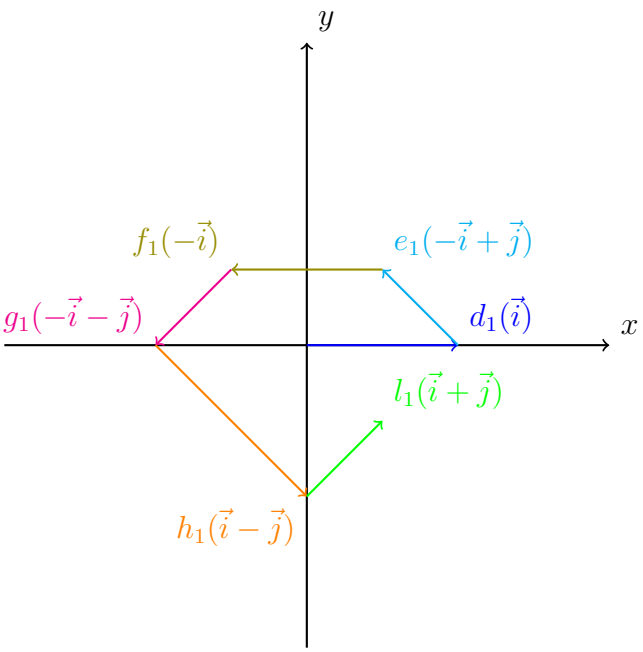

Figura 9: Minha espiral no ciclo $c = 1$

Aplicando a equação (1) para o ciclo $c = 2$ e proporção $p = 1$ para d_c, f_c, g_c, l_c e $p = 2$ para e_c e h_c:

$$\begin{aligned} d_2 &= 2 + 1(2-1) = 3 & \qquad g_2 &= 1 + 1(2-1) = 2 \\ e_2 &= 1 + 2(2-1) = 3 & \qquad h_2 &= 2 + 2(2-1) = 4 \\ f_2 &= 2 + 1(2-1) = 3 & \qquad l_2 &= 1 + 1(2-1) = 2 \end{aligned}$$

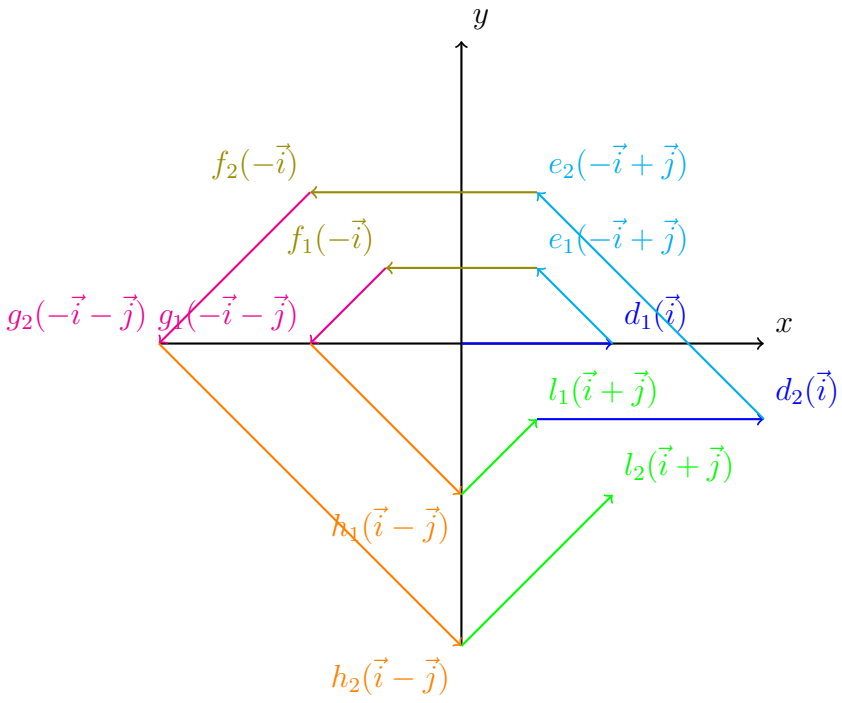

Figura 10: Minha espiral no ciclo $c = 2$

3 Desenvolvimento

O desenvolvimento da implementação do método criado para se calcular cada ponto de uma espiral foi concebido na linguagem de programação *C*.

3.1 Algoritmo

O algoritmo que aplica a lógica por trás da criação de espirais estudado até agora se baseou na criação de um laço de repetição que "anda" um ponto na espiral a cada iteração.

Código 1: Parte de uma função em C que calcula o n-ésimo ponto da espiral quadrada

```
    for (int i = 0; i < n; i++)
    {
        if (scalarD != 0)
        {
            yCoordinate++;
            scalarD--;
        }
        else if (scalarE != 0)
        {
            xCoordinate--;
            scalarE--;
        }
        else if (scalarF != 0)
        {
            yCoordinate--;
            scalarF--;
        }
        else if (scalarG != 0)
        {
            xCoordinate++;
            scalarG--;
        }
        else
        {
            spiralCycle++;
            scalarD = 1 + (2 * (spiralCycle - 1));
            scalarE = 1 + (2 * (spiralCycle - 1));
            scalarF = 2 + (2 * (spiralCycle - 1));
            scalarG = 2 + (2 * (spiralCycle - 1));

            if (scalarD != 0)
            {
                yCoordinate++;
                scalarD--;
            }
            else if (scalarE != 0)
            {
                xCoordinate--;
                scalarE--;
            }
            else if (scalarF != 0)
            {
                yCoordinate--;
                scalarF--;
            }
            else if (scalarG != 0)
```

```
47              {
48                  xCoordinate++;
49                  scalarG--;
50              }
51          }
52      }
```

Todas as funções criadas durante o trabalho seguem uma mesma lógica simples: seja n o n-ésimo ponto em uma determinada espiral, um laço de repetição *for* com n iterações calcula um "passo" na espiral a cada repetição.

Pela definição (1), os "movimentos" de uma espiral, descritos pela disposição de seus vetores-base em ordem, acompanhados por uma crescente proporção p imposta aos seus escalares em cada ciclo c, sempre acontecem em uma predefinida ordem. Assim, cada unidade de um escalar representa um "passo" na espiral, por serem dispostos em ordem, é possível descrever sua hierarquia por meio do aninhamento de estruturas condicionais *if* e *else*. Essas estruturas condicionais garantem a ordem da decomposição de cada "passo" acumulado em cada escalar na direção correta. Dessa forma, se um escalar arbitrário k for 0, então os movimentos que $k\vec{v}$ descreve já foram computados. Uma vez que todos os escalares são 0, um ciclo foi completo, se fazendo necessário atualizar os valores de todos os escalares para compreenderem a nova proporção $k_c = k_1 + p((c+1)-1)$

3.2 Análise de complexidade

Na análise assintótica de funções, uma função $f(n)$ é dita ser $O(g(n))$, lê-se "O grande de g de n", se, e somente se, existir uma constante c tal que $f(n) \leq cg(n)$, com $c, n \in \mathbb{R}$. Ou seja, a função $f(n)$ está limitada superiormente por $g(n)$ por um fator constante. Assim, seu crescimento é dito ser linear.

Como solicitado pelo enunciado do trabalho, a complexidade de tempo de todos os algoritmos é $O(n)$, pois a única operação variável do algoritmo se encontra no laço de repetição *for*, cujo número de iterações é definido por meio da entrada padrão. Uma vez que as estruturas condicionais e as operações dentro do laço possuem custo assintótico constante, elas não interferem de modo determinante na complexidade do algoritmo.

4 Conclusão

O trabalho foi muito enriquecedor e o problema apresentado foi interessante. Além do estudo de espirais para encontrar a implementação de um algoritmo em *C* que possua no mínimo uma complexidade linear, outras ferramentas essenciais para um estudante de Ciência da Computação foram utilizadas durante o projeto. Para o desenvolvimento da documentação foi utilizado o LaTeX, para a compilação de forma rápida e conveniente dos arquivos *C* foi utilizado um arquivo *Makefile* e para o controle de versionamento do código foram utilizadas as ferramentas *Git* e *GitHub*, cujo repositório pode ser encontrado de forma pública em *discrete-math-project*.