



Programação Gulosa e Algoritmo Geométrico

Nesta apresentação, exploraremos os conceitos fundamentais da programação gulosa e geométrico, suas características, vantagens, desvantagens e aplicações, incluindo a resolução do problema do caixeiro viajante.

O que é a programação gulosa?

1 Definição

O programação gulosa é uma abordagem de resolução de problemas que toma decisões localmente ótimas a cada etapa, com a esperança de encontrar uma solução global ótima.

2 Tomada de Decisão

O algoritmo guloso toma decisões imediatas, sem considerar as consequências a longo prazo, buscando a melhor opção disponível no momento.

3 Simplicidade

Essa estratégia é fácil de implementar e geralmente requer menos tempo e esforço computacional do que outros algoritmos.



Características e Vantagens da Programação Gulosa

Características

Foco na solução local ideal, sem considerar o resultado global.

Rápida execução, pois não realiza análises complexas.

Consumo menor de recursos computacionais.

Vantagens

Simplicidade de implementação e facilidade de uso.

Eficiência em problemas que podem ser resolvidos localmente.

Rapidez na obtenção de resultados.





Desvantagens da Programação Gulosa

Solução Local Ótima

O algoritmo guloso pode encontrar apenas soluções localmente ótimas, não garantindo a solução global ideal.

Falta de Visão Global

Ao focar apenas na melhor opção imediata, o algoritmo pode ignorar informações importantes que poderiam levar a uma solução melhor.

Problemas Complexos

O algoritmo guloso pode não ser eficaz em problemas com muitas restrições ou que requerem uma análise mais abrangente.

Não Garantia de Optimalidade

Não há garantia de que o algoritmo guloso encontrará a solução ótima, mesmo que essa exista.



$t1 = 6, t2 = 5, t3 = 3 : 14$



$t1 = 1, t2 = 3, t3 = 6 : 10$



Quando Usar a Programação Gulosa

1

Problemas Simples

O algoritmo guloso é eficaz em problemas que podem ser resolvidos localmente, como encontrar o menor caminho entre dois pontos.

2

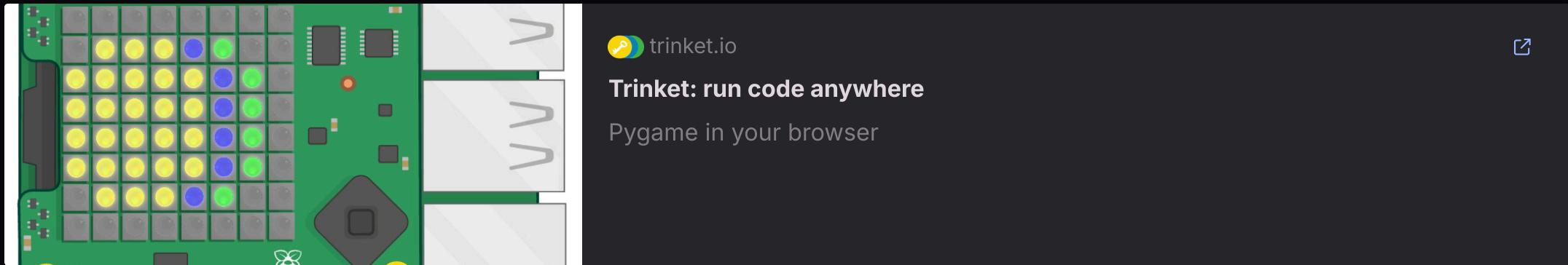
Restrições Limitadas

Quando o problema tem poucas restrições e as soluções locais são boas aproximações da solução global.

3

Rapidez Necessária

Quando a velocidade de execução é crucial e não é necessária a solução ótima global.



Algoritmos Gulosos

- Exemplo R\$ 0,15
- Moedas: 1, 5, 10, 25, 100
 - 10 e 5
- Moedas: 1, 7 e 10
 - 10, 1, 1, 1, 1 e 1 (**ótimo local**)
 - É a melhor solução?
 - Não
 - A melhor solução seria: 7, 7 e 1 (**ótimo global**)

```
#https://trinket.io/features/pygame

# Coins in reverse bigger to smaller
def greedy_coin_change(coins, amount):
    coins.sort(reverse=True)
    result = []
    total_coins = 0

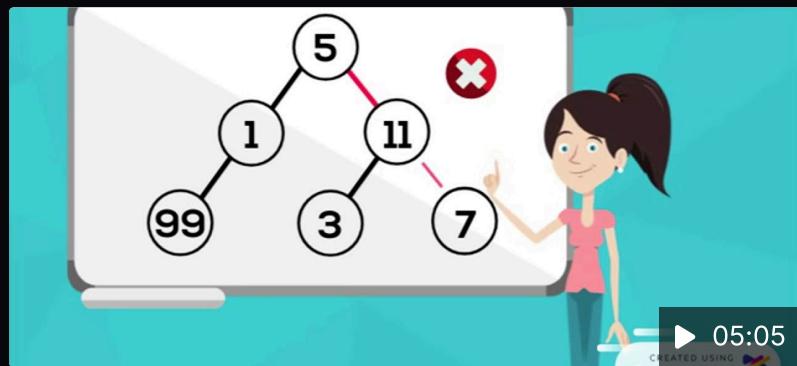
    for coin in coins:
        while amount >= coin:
            amount -= coin
            result.append(coin)
            total_coins += 1

    if amount > 0:
        print("Não é possível fornecer troco exato com as moedas disponíveis.")
        return None

    return result, total_coins

if __name__ == "__main__":
    coins = list(map(int, input("Insira as moedas disponíveis separadas por espaços: ").split()))
    amount = int(input("Insira o valor que deseja trocar: "))

    change, num_coins = greedy_coin_change(coins, amount)
    if change:
        print(f"Moedas usadas: {change}")
        print(f"Número total de moedas: {num_coins}")
```



YouTube

Greedy Algorithm | Algoritmo Gúloso

Feito pelos alunos Letícia Aragão, Brendon Angelo e Lucca Bortoloso, da Universidade Tiradentes, para a disciplina Teoria da Computação, lecionada pelo...





O que é o Algoritmo Geométrico?

1 Definição

O algoritmo geométrico utiliza conceitos e técnicas da geometria para resolver problemas, geralmente relacionados a elementos gráficos e espaciais.

2 Abordagem Espacial

Essa abordagem leva em consideração a localização, a forma e a distância entre os elementos envolvidos no problema.

3 Aplicações

Muito utilizado em jogos, design gráfico, realidade virtual e outras áreas que envolvem representações visuais e interações espaciais.

Características e Vantagens do Algoritmo Geométrico

Características

Utiliza conceitos e técnicas geométricas, como distância, ângulo e posição.

Lida com representações gráficas e interações espaciais.

Abordagem matemática e sistemática para resolver problemas.

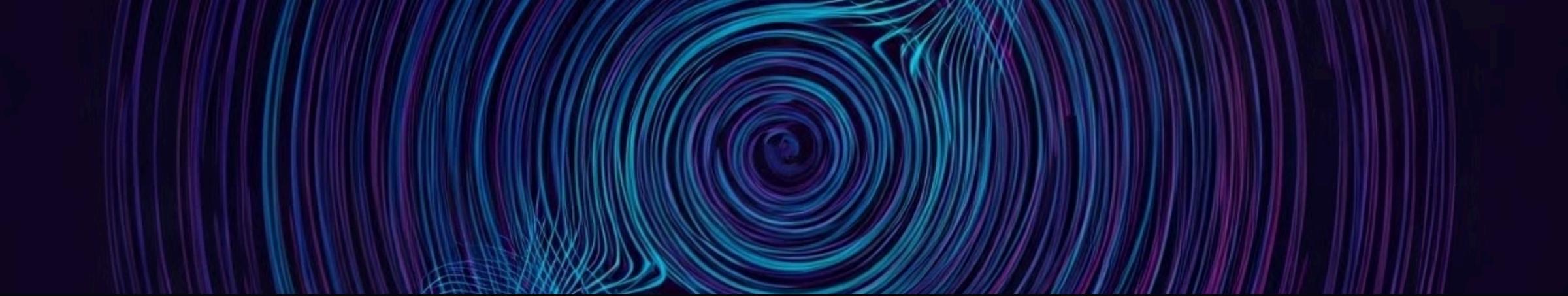
Vantagens

Eficiência em problemas que envolvem elementos gráficos e espaciais.

Precisão nos cálculos e resultados.

Facilidade de integração com sistemas de gráficos e visualização.





Desvantagens do Algoritmo Geométrico

Complexidade Computacional

Os cálculos geométricos podem ser mais complexos e exigir maior capacidade computacional.

Limitações em Problemas Não-Espaciais

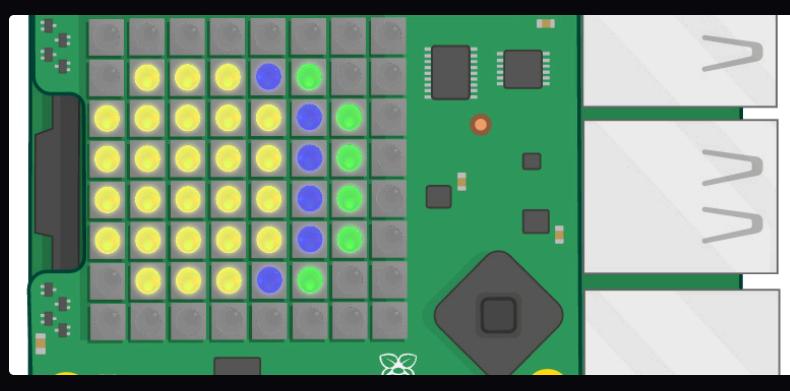
O algoritmo geométrico pode não ser tão eficaz em problemas que não envolvem elementos gráficos ou espaciais.

Sensibilidade a Precisão

Os resultados do algoritmo geométrico podem ser sensíveis a erros de precisão numérica.

Dificuldade de Implementação

A implementação de algoritmos geométricos pode ser mais complexa do que a de outros tipos de algoritmos.



trinket.io

Trinket: run code anywhere

Pygame in your browser



```
import pygame
import math

WIDTH, HEIGHT = 800, 600
FPS = 15

# Define general class
class Ball:
    def __init__(self, x, y, radius, color, velocity):
        self.x = x
        self.y = y
        self.radius = radius
        self.color = color
        self.velocity = velocity

    def move(self):
        self.x += self.velocity

    def draw(self, screen):
        pygame.draw.circle(screen, self.color, (int(self.x), int(self.y)), self.radius)

def is_colliding(ball1, ball2):
    distance = math.sqrt((ball1.x - ball2.x) ** 2 + (ball1.y - ball2.y) ** 2)
    return distance <= (ball1.radius + ball2.radius)

# init pygame
pygame.init()
screen = pygame.display.set_mode((WIDTH, HEIGHT))
pygame.display.set_caption("Crash two orb")
clock = pygame.time.Clock()

# Create two balls
ball1 = Ball(100, HEIGHT // 2, 30, (0, 255, 0), 3) # green orb to right
ball2 = Ball(700, HEIGHT // 2, 30, (155, 155, 255), -3)

# Loop
running = True
while running:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False

    # Moving
    ball1.move()
    ball2.move()

    # Verify colision
    if is_colliding(ball1, ball2):
        # Reverse when crash
        ball1.velocity = -ball1.velocity
        ball2.velocity = -ball2.velocity
        ball1.color = (255, 0, 0) # change color if crash green
        ball2.color = (255, 0, 255)

    # Draw
    screen.fill((0, 0, 0)) # clean screen
    ball1.draw(screen)
    ball2.draw(screen)

    pygame.display.flip() # refresh screen
    clock.tick(FPS)

pygame.quit()
```

Aplicações do Algoritmo Geométrico



Jogos

Essencial para a detecção de colisões, cálculo de trajetórias e renderização gráfica em jogos.



CAD

Usado para modelagem 3D, análise de formas e planejamento de layouts em aplicativos de design assistido por computador.



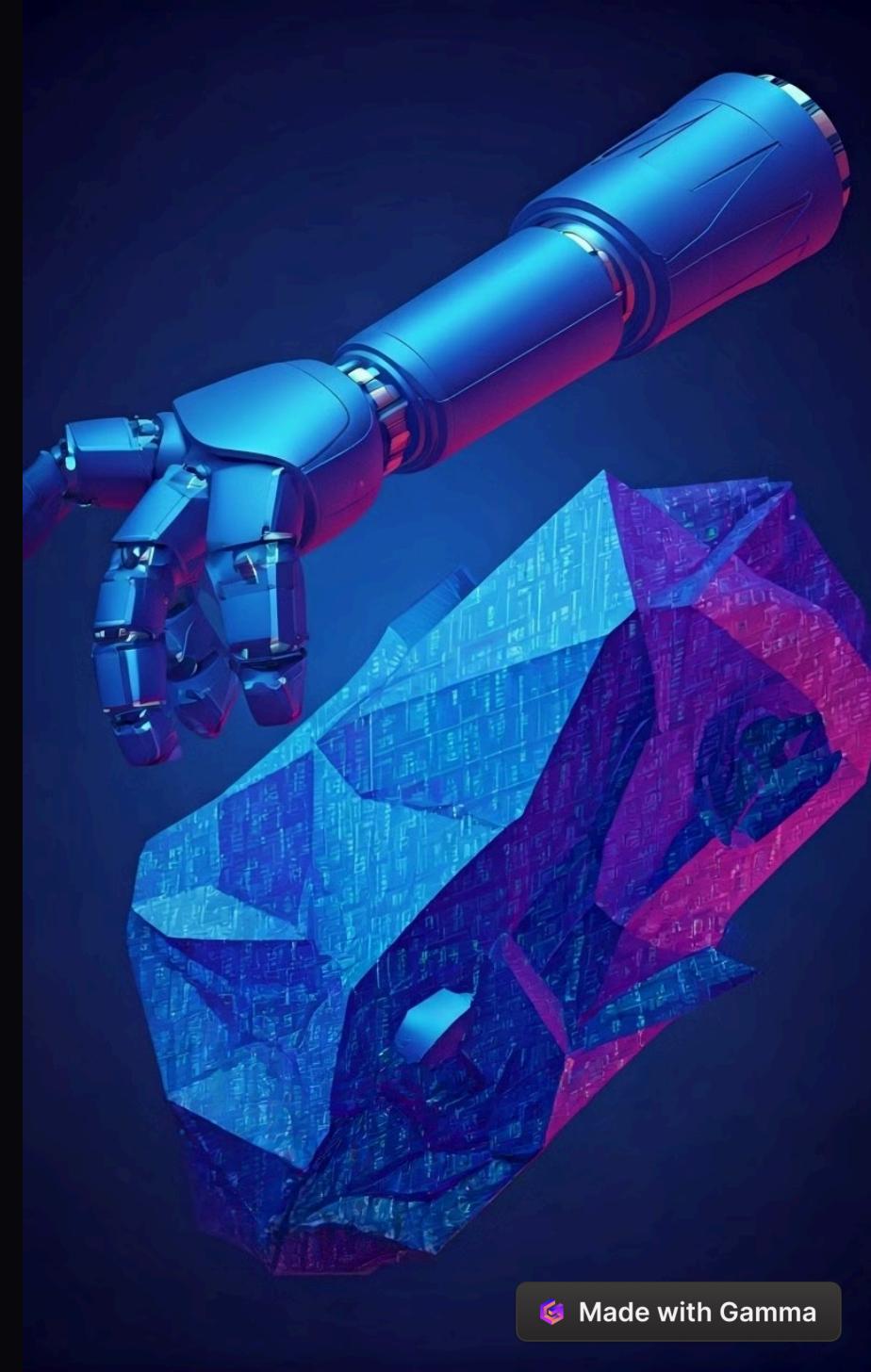
Design Gráfico

Fundamental para operações como transformações, posicionamento e manipulação de elementos gráficos.



Realidade Virtual

Aplicado no cálculo de interações, detecção de colisões e renderização de ambientes 3D imersivos.

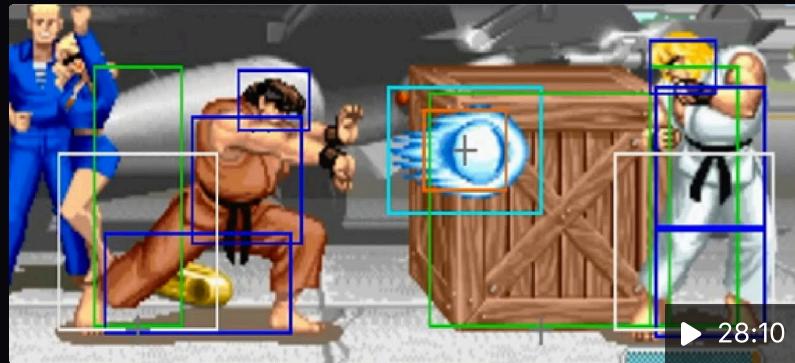


Algoritmos Geométricos em Jogos

são a espinha dorsal de muitos dos jogos que amamos. Eles são a matemática por trás da criação de mundos virtuais realistas e interativos. Desde a detecção de colisões simples até a renderização de paisagens complexas, a geometria desempenha um papel fundamental.

Por que a geometria é tão importante em jogos?

- **Detecção de colisões:** Quando um personagem pula, atira em um alvo ou simplesmente anda por um cenário, o jogo precisa saber se houve algum contato com outros objetos. Algoritmos geométricos determinam se dois objetos se interseccionam, permitindo que o jogo responda de forma realista.
- **Criação de mundos 3D:** A construção de ambientes virtuais envolve a definição de formas geométricas básicas (cubos, esferas, cilindros) e a combinação delas para criar objetos mais complexos. Algoritmos geométricos são usados para calcular a posição e orientação desses objetos no espaço.
- **Iluminação e sombras:** A forma como a luz interage com os objetos em um jogo é crucial para criar uma atmosfera realista. Algoritmos geométricos são utilizados para calcular a direção da luz, a formação de sombras e os reflexos.
- **Caminho de personagens e objetos:** Para que personagens e objetos se movimentem de forma natural em um ambiente virtual, é necessário calcular o caminho mais curto entre dois pontos, evitando obstáculos. Algoritmos de busca de caminho, como o algoritmo A*, utilizam conceitos geométricos para encontrar a melhor rota.
- **Física:** A simulação de física em jogos, como a gravidade, a força e o movimento, também se baseia em princípios geométricos. Por exemplo, para calcular a trajetória de uma bola, é necessário considerar sua forma, massa e as forças que atuam sobre ela.



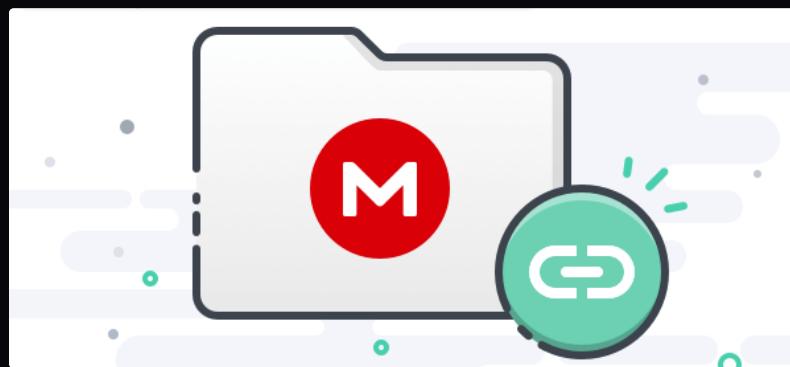
YouTube

Como os jogos de luta funcionam? Analisando as caixas de colisão!

Vamos estudar as caixas de colisão de alguns jogos famosos. Arquivos:

https://drive.google.com/file/d/1WbEhBfPuyd4e4TVtHid25lYrfQs4b_Z/view?...





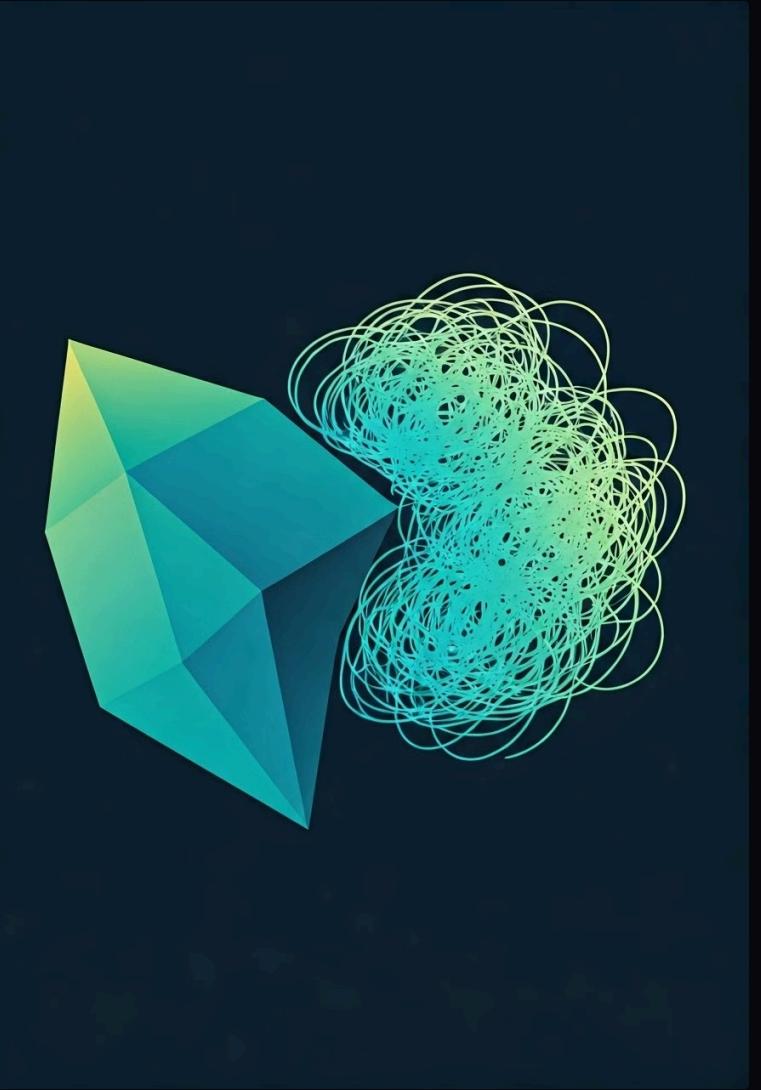
 mega.nz

125.05 MB folder on MEGA

3 files



Conclusão



Algoritmos geométricos e gulosos desempenham papéis importantes na computação e resolução de problemas. Enquanto os algoritmos geométricos se concentram na precisão e otimização global, os algoritmos gulosos se destacam pela simplicidade e eficiência computacional. Juntos, esses dois tipos de algoritmos fornecem soluções poderosas e abrangentes para uma ampla gama de problemas.