Classe Pilha e Torre de Hanói

Charles RIbeiro Chaves - 122086950
Filipe Viana da Silva - 121050053
Vinícius Brasil de Oliveira Barreto - 120029237

15 de junho de 2025

Sumário

1 Introdução

Este relatório apresenta o desenvolvimento de uma classe `Pilha` em Python, utilizando a estrutura `array` da biblioteca padrão, e sua aplicação na solução do problema clássico da Torre de Hanói com N discos. O objetivo foi implementar uma estrutura de dados eficiente e uma rotina recursiva que respeita as regras do problema, além de permitir a visualização interativa dos passos da solução.

2 Estruturas de Dados Utilizadas

A estrutura principal desenvolvida foi a classe `Pilha`, que armazena elementos de um mesmo tipo básico (inteiros) usando internamente um `array` da biblioteca padrão do Python. Essa escolha permite armazenamento eficiente e controle rígido do tipo dos dados, além de operações rápidas de empilhar e desempilhar.

A classe `Pilha` implementa os métodos fundamentais: `empilha`, `desempilha`, verificação de pilha vazia ou cheia, troca dos dois elementos do topo e consulta do tamanho. Exceções personalizadas foram criadas para tratar erros de tipo, pilha cheia e pilha vazia, garantindo robustez.

Para a solução da Torre de Hanói, três instâncias da classe `Pilha` representam os pinos, onde os discos são empilhados em ordem decrescente de tamanho.

3 Divisão de Módulos e Descrição das Rotinas

O programa foi estruturado em módulos funcionais:

- **Classe Pilha**: encapsula a estrutura de dados e suas operações básicas.
- **Função** `torre_de_hanoi`: rotina recursiva que implementa o algoritmo clássico da Torre de Hanói, movendo discos entre as pilhas conforme as regras.
- **Função de impressão visual**: imprime o estado atual das pilhas verticalmente, representando os discos com blocos de caracteres, facilitando a visualização da solução no terminal.
- **Função principal** `main`: gerencia a entrada do usuário, inicializa as pilhas, controla o fluxo da execução e a interação para visualização dos passos.

4 Complexidade de Tempo e Espaço

A complexidade do algoritmo da Torre de Hanói é clássica: o número mínimo de movimentos para N discos é $2^N - 1$, logo a complexidade de tempo é exponencial em N. Cada movimento consiste em operações constantes de empilhar e desempilhar, portanto o custo por movimento é $O(1)$.

Quanto ao espaço, a classe `Pilha` utiliza arrays com capacidade fixa, proporcional a N, para armazenar os discos. A recursão da Torre de Hanói utiliza pilha de chamadas com profundidade máxima N, o que é eficiente para valores moderados de N.

5 Problemas e Observações

Durante o desenvolvimento, foi observado que a verificação explícita para impedir empilhar discos maiores sobre menores não era necessária, pois a lógica recursiva do algoritmo já garante essa restrição. Remover essa verificação evitou exceções desnecessárias e possíveis loops.

Outro desafio foi implementar uma visualização clara e intuitiva das pilhas no terminal. A solução adotada imprime as pilhas verticalmente, com discos representados por blocos de caracteres proporcionais ao seu tamanho, facilitando o acompanhamento da solução passo a passo.

O controle interativo da execução, permitindo ao usuário definir quantos movimentos entre exibições e aguardar confirmação para continuar, aumentou a usabilidade e compreensão do algoritmo.

6 Conclusão

A implementação da classe `Pilha` mostrou-se adequada para representar os pinos da Torre de Hanói, fornecendo operações eficientes e seguras para manipulação dos discos. A rotina recursiva da Torre de Hanói, integrada com a classe, solucionou corretamente o problema respeitando as regras impostas.

A visualização gráfica no terminal, embora simples, contribuiu significativamente para o entendimento do processo, especialmente para usuários iniciantes. O controle interativo dos passos permitiu acompanhar a evolução da solução de forma didática.

Em resumo, os códigos desenvolvidos cumpriram os objetivos propostos, fornecendo uma base sólida para estudos posteriores sobre estruturas de dados, algoritmos recursivos e interação com o usuário em programas de terminal.