**Introdução**

Este relatório descreve a solução desenvolvida para o problema de encontrar a maior sequência de caixas aninhadas, baseado em um catálogo de caixas com dimensões especificadas. O problema envolve determinar quantas caixas podem ser colocadas uma dentro da outra, de forma que cada caixa subsequente seja menor em todas as dimensões que a anterior.

**Descrição do Problema**

O objetivo é automatizar a tarefa de identificar a maior sequência de caixas aninhadas a partir de um catálogo de caixas de papelão. Cada caixa é caracterizada por três dimensões: largura, altura e comprimento. A tarefa consiste em encontrar o comprimento máximo de uma sequência onde cada caixa subsequente pode ser completamente inserida na anterior.

**Descrição da Solução**

**Implementação**

A solução foi implementada em Java, dividida em três principais classes:

1. **Classe Caixa**: Representa uma caixa de papelão com dimensões ordenadas automaticamente no momento da criação. Implementa métodos para comparação entre caixas e verificação de aninhamento.
2. **Classe Gerenciador**: Responsável por ler um arquivo contendo as dimensões das caixas, armazená-las em uma lista, ordená-las e calcular a maior sequência de caixas aninhadas usando programação dinâmica.
3. **Classe Main**: Contém o método main que inicializa o Gerenciador, lê o arquivo de entrada, ordena as caixas e imprime o comprimento da maior sequência aninhada.

**Estrutura de Dados e Algoritmos**

* **Caixa**:
  + Armazena e ordena automaticamente suas dimensões.
  + Implementa método de comparação para ordenação.
  + Implementa método para verificar se pode conter outra caixa.
* **Gerenciador**:
  + Usa BufferedReader para ler o arquivo de entrada linha por linha.
  + Cria instâncias de Caixa com base nas dimensões lidas e as armazena em uma lista.
  + Ordena as caixas usando Collections.sort.
  + Utiliza programação dinâmica para calcular o comprimento da maior sequência aninhada.

**Testes Realizados**

Foram realizados testes com diferentes conjuntos de caixas para verificar a corretude e eficiência da solução implementada. Os testes envolveram:

* Conjuntos pequenos de caixas para verificar casos simples.
* Conjuntos maiores e variados para garantir que o algoritmo fosse escalável.
* Verificação manual dos resultados contra os resultados esperados.

**Resultados e Conclusões**

A solução desenvolvida se mostrou eficiente e robusta para encontrar a maior sequência de caixas aninhadas. Os testes realizados confirmaram que o algoritmo funciona corretamente e é capaz de lidar com diferentes tamanhos de entrada de forma adequada. A estrutura de programação dinâmica utilizada proporcionou um desempenho satisfatório, mesmo para conjuntos de dados mais extensos.

**Recomendações Futuras**

Para melhorar ainda mais a solução, poderiam ser considerados os seguintes pontos:

* Implementar técnicas de otimização para reduzir o tempo de execução em casos extremamente grandes.
* Adicionar validações adicionais para garantir que os dados de entrada estejam corretos e completos.
* Expandir a solução para lidar com formatos diferentes de entrada de dados, como arquivos CSV ou bancos de dados.

**Referências**

A solução foi desenvolvida com base nos conceitos de algoritmo e estrutura de dados estudados, bem como na prática de programação em Java.

Este relatório conclui a descrição e análise da solução para o problema de encontrar a maior sequência de caixas aninhadas.