# Otimização do Descarregamento de Containers

Aluno: Ruan Santos Paiva de Alvernaz

Algoritmos e Estruturas de Dados

Professor: Alexandre de Oliveira

1. Identificação da Complexidade do Problema

O problema apresentado é uma variação do problema clássico de Bin Packing, que é classificado como NP-Difícil (NP-Hard). O objetivo é alocar volumes variáveis de containers em armazéns e posteriormente em caminhões, ambos com capacidades fixas, de modo a minimizar o número total de unidades utilizadas. Isso requer encontrar a melhor combinação possível, sem um algoritmo de tempo polinomial conhecido que garanta a solução ótima. Portanto, o problema é classificado como NP-Difícil.

2. Estratégias de Resolução Sugeridas

A estratégia escolhida é o algoritmo guloso utilizando a heurística First Fit Decreasing (FFD). Primeiro os volumes são ordenados em ordem decrescente e, em seguida, cada item é alocado no primeiro local onde couber. Caso não caiba em nenhum, um novo armazém (ou caminhão) é aberto. Essa técnica é simples, eficiente e proporciona bons resultados para problemas NP-difíceis como este.

3. Desenvolvimento do Algoritmo em Java

import java.util.\*;  
  
public class OtimizadorLogistico {  
  
 public static int calcularBins(int[] volumes, int capacidadeMaxima) {  
 Arrays.sort(volumes);  
 List<Integer> bins = new ArrayList<>();  
  
 for (int i = volumes.length - 1; i >= 0; i--) {  
 boolean alocado = false;  
 for (int j = 0; j < bins.size(); j++) {  
 if (bins.get(j) + volumes[i] <= capacidadeMaxima) {  
 bins.set(j, bins.get(j) + volumes[i]);  
 alocado = true;  
 break;  
 }  
 }  
 if (!alocado) {  
 bins.add(volumes[i]);  
 }  
 }  
 return bins.size();  
 }  
  
 public static void main(String[] args) {  
 Scanner sc = new Scanner(System.in);  
 System.out.println("Informe os volumes dos containers (separados por espaço):");  
 String[] entrada = sc.nextLine().split(" ");  
 int[] volumes = new int[entrada.length];  
 for (int i = 0; i < entrada.length; i++) {  
 volumes[i] = Integer.parseInt(entrada[i]);  
 }  
  
 System.out.print("Informe a capacidade de um armazém: ");  
 int capacidadeArmazem = sc.nextInt();  
  
 System.out.print("Informe a capacidade de um caminhão: ");  
 int capacidadeCaminhao = sc.nextInt();  
  
 int armazens = calcularBins(volumes, capacidadeArmazem);  
 int totalCarga = Arrays.stream(volumes).sum();  
 int caminhoes = (int) Math.ceil((double) totalCarga / capacidadeCaminhao);  
  
 System.out.println("Número mínimo de armazéns necessários: " + armazens);  
 System.out.println("Número mínimo de caminhões necessários: " + caminhoes);  
 }  
}

Explicação do Código:

- O método calcularBins organiza os volumes e tenta alocar cada um no primeiro armazém ou caminhão onde caiba.

- Caso não haja espaço, um novo é criado.

- O método main coleta os dados de entrada, executa o algoritmo e imprime os resultados.

Exemplo de teste no console:

Informe os volumes dos containers (separados por espaço):  
10 20 15 5 25 30

Informe a capacidade de um armazém: 50

Informe a capacidade de um caminhão: 60

Saída esperada:

Número mínimo de armazéns necessários: 3

Número mínimo de caminhões necessários: 2

4. Análise da Complexidade

O algoritmo possui duas etapas principais:  
- Ordenação dos volumes: O(n log n)  
- Alocação nos armazéns: O(n²) no pior caso  
Portanto, a complexidade total é O(n²). Essa complexidade se deve ao fato de que, para cada volume, pode ser necessário verificar todos os armazéns já existentes.