Trabalho Prático 3

Thaís Ferreira da Silva - 2021092571

Universidade Federal de Minas Gerais (UFMG)
Belo Horizonte - MG - Brasil

thaisfds@ufmg.br

1 Introdução

O problema proposto para esse trabalho prático foi a modelagem de um algoritmo que fosse capaz de otimizar o processo de distribuição de ligas metálicas de uma fábrica, já que cada fabrica produz ligas metálicas de diferentes tamanhos para atender as diferentes demandas dos clientes.

Para solucionar esse problema é necessário implementar um algoritmo com programação dinâmica capaz de receber o número de casos de teste, número de tipos de ligas metálicas disponível e a demanda do cliente, em metros, e dizer a quantidade miníma de ligas necessárias para atender a demanda do cliente para cada caso de teste.

2 Modelagem

A implementação desse trabalho prático foi menor mas não foi simples de se pensar na ideia por traz do código. Dessa forma, foram implementados apenas uma classe chamada Fabrica que possui uma único método

- **otimizacaoLigasMetalicas:** Essa função recebe um vetor contendo os tamanhos das ligas metálicas em um vetor chamado ligasMetalicas, número de tipos de ligas metálicas disponível ($1 \leq N \leq 1000$) e a demanda do cliente, em metros ($1 \leq W \leq 1000000$).

  Nela criamos um vetor que é o responsável por armazenar todos os melhores mínimos para cada tamanho que é utilizado na programação dinâmica, desa forma evitamos cálculos repetitivos ao longo da execução do programa. Os volares do vetor são inicializados com a demanda do cliente + 1.

  Esse valor vai aos poucos reduzindo ao longo da execução do código até alcançarmos a menor quantidade possível.

  Por fim realizamos dois loops para calcular a menor quantidade de ligas que podemos utilizar para cada tamanho, indo do 1 até o W que é a demanda do cliente, e analisando para cada tamanho menor ou igual a W qual a melhor quantidade de ligas a se utilizar. Se o novo calculo for melhor do que o antigo, atualiza o vetor da programação dinâmica.

  O código segue o pseudocódigo a seguir:

PSEUDOCODIGO OTIMIZAÇÃO DINÂMICA
Inicia o método pegando a o vetor com os tamanhos das ligas metálicas da fábrica,
número de tipos de ligas metálicas disponível e a demanda do cliente, em metros
Inicializar o vetor com os melhores mínimos das ligas inicializando com a demanda + 1
Primeiro menor liga é inicializado com 0
Para cada tamanho de liga metálica i até a demanda W
  Para cada liga metálica disponível
    se i - ligasMetalicas[j] $\geq$ 0
      atualiza melhorMinimoLigas[i] com o mínimo do antigo ou novo calculo
se melhorMinimoLigas[Demanda] diferente do valor inicial
  retorna melhorMinimoLigas[Demanda]
se não
  retorna -1 como erro

---

3 Análise

3.1 Complexidade de Tempo

Em relação a complexidade de tempo do algoritmo temos 3 loops alinhados ao longo do código. No método otimizacaoLigasMetalicas da fábrica temos uma complexidade O(W*N) já que iteramos para cada demanda do cliente verificando a otimização de acordo com o número de tipos de ligas metálicas disponível. Além disso, no main é necessário realizar a chamada do método de otimização para cada caso de teste, o que torna a complexidade final O(T*(W*N)).

3.2 Complexidade de Espaço

Sobre a complexidade de Espaço do algoritmo, realizamos a alocação de 2 vetores ao longo do código, um contendo o número de tipos de ligas metálicas disponível (ligasMetalicas) com complexidade O(W) e o outro contendo os dados da programação dinâmica (melhorMinimoLigas) com complexidade O(N). Dessa forma a complexidade total do código é O(W+N).

3.3 Complexidade de Tempo em função do tamanho da entrada

A complexidade de tempo e espaço deve ser baseada no número de bits necessários para representar o tamanho da entrada. Tendo isso em mente, sabemos que a complexidade de tempo do código é O(T*(W*N)), onde T é o número de casos de teste, W é a quantidade de ligas solicitadas e N é o número de ligas metálicas disponíveis. Para representar T, W e N, são necessários log2(T), log2(W) e log2(N) bits, respectivamente. Portanto, a complexidade de tempo baseada no número de bits necessários para representar o tamanho da entrada é O($2^{log2(T)+log2(W)+log2(N)}$).

4 Redução NP-Completude

Vamos reduzir o problema estudado em sala Soma de Subconjuntos que já sabemos que é NP-Completo para o problema das ligas metálicas. Para isso considere os seguintes problemas de decisão:

Soma de Subconjuntos:
- Entrada: n inteiros positivos $w_1, ..., w_n$ e um inteiro W
- Pergunta: existe um conjunto I $\subseteq$ 1, ... , n tal que $\sum_{i \in I} w_i = W$

Ligas Metálicas:
- Entrada: m inteiros positivos $1 = lm_0 < lm_1 < ... < lm_{m-1}$, um inteiro S e uma demanda T
- Pergunta: existe um multiconjunto I tal que $\sum_{i \in I} lm_i = S$ e $|I| \leq$ T?

Dessa forma, dada uma entrada da soma de subconjuntos, faça $b = max(n+1, W+1)$, $lm_0 = 1$, e para cada i $\in 1, ..., n$, definimos:

$$lm_i = w_i * b^{n+1} + b^i$$
$$lm`_i = b^i$$
$$S = W * b^{n+1} + \sum_{i=1}^{n} b^i$$
$$T = n$$

Então ((w_1,...,w_n),W) é uma instancia na Soma de Subconjuntos se e somente se ((lm_0, ... , $ml_n w_n$, $lm`_1$ , ... , $lm`_n$),S,T) é uma instancia de Ligas Metálicas. Dessa forma podemos perceber que o problema da Soma de Subconjuntos é um caso especifico do problema das Ligas Metálicas onde não podemos repetir um mesmo tamanho de liga.

Construímos assim uma representação na base b onde garantimos que selecionando no máximo n ligas dentre as opções disponibilizadas por cada fábrica, nenhum carry pode existir nos últimos n+1.