Trabalho Prático 3 - Centro de Distribuição

Filipe de Araújo Mendes - 2021031920

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

flipeara@ufmg.br

1. Introdução

O trabalho prático 3 propõe um problema em encontrar a quantidade mínima de ligas metálicas necessárias para completar um pedido de um cliente, a partir de tamanhos disponíveis no centro de distribuição. Foi uma proposta para aprender e treinar a programação dinâmica, guardando os valores calculados em um vetor, para não precisar fazer o mesmo cálculo várias vezes.

2. Modelagem

São passados como entrada os tamanhos de ligas metálicas e o pedido do cliente, e então deve ser impresso o número mínimo de ligas necessárias para completar o pedido.

Em um primeiro momento, pode-se imaginar que ao utilizar um algoritmo guloso, poderia ser possível encontrar a solução ótima. O algoritmo em questão seria sempre pegar a maior liga disponível até que o pedido tenha sido concluído, mas é fácil pensar em um contra exemplo para isso.

Por exemplo, se pegarmos ligas de tamanho 1, 4 e 5 metros, e um pedido de 12 metros em ligas metálicas, utilizando o algoritmo guloso, temos que o pedido pode ser realizado utilizando duas ligas de 5 metros e duas ligas de 1 metro, totalizando 12 metros com 4 ligas. Entretanto, a solução ótima utiliza apenas 3 ligas metálicas de 4 metros, uma liga a menos que a solução gulosa.

Assim, como a solução gulosa falhou, foi escolhido o método de força bruta para solucionar o problema, com a utilização da programação dinâmica, o qual será explicado mais tarde.

2.1. Centro de Distribuição

Foi criada a classe do centro de distribuição, responsável por encontrar a quantidade mínima de ligas necessárias. Para isso, é necessário guardar os tamanhos de ligas e um outro vetor, que guarda as informações já calculadas para não precisar calcular novamente.

2.2. Algoritmo força bruta:

Para encontrar a solução ótima foi usado um algoritmo de força bruta, calculando o número de ligas metálicas necessárias para completar pedidos de tamanho 0 até o tamanho n pedido pelo cliente, utilizando uma estratégia bottom up, para evitar centenas de milhares de chamadas recursivas, as quais não seriam possíveis pela quantidade limitada de memória.

Se, por exemplo, temos ligas de tamanho 1, 2 e 5 o número mínimo de ligas metálicas necessárias para completar um pedido de tamanho n é uma liga a mais do que o necessário para completar o min(n-1, n-2, n-5) e para completar um pedido de tamanho 0, não são necessárias ligas metálicas. Assim, temos a equação de recorrência e o algoritmo para solucionar o problema.

A equação de recorrência para a função FLA (Find Least Alloys), dado um vetor AT (Alloy Types) de tamanho m é a seguir:

FLA(0) = 0;
FLA(n) | n > 0 = min(FLA(n - AT[0]), … , FLA(n - AT[m]))

Agora, precisamos analisar a complexidade do algoritmo. Consideremos então, o seguinte problema de decisão:

Dado n, tamanho[t1,...,tn], dois inteiros P e k.
Pergunta: Consigo completar o pedido P com um número = k?

Pensando no tamanho da entrada, o vetor de tamanhos de liga utiliza espaço n*log m, onde n é a quantidade de ligas diferentes e m é o tamanho da maior liga metálica do vetor. O pedido, por sua vez, utiliza log x bits | log x = p, em que representa o pedido do cliente.

Para a complexidade de tempo do algoritmo, a parte importante é a função FLA(), uma vez que todo o resto pode ser executado em tempo polinomial.

A função FLA, então, executa de 0 a p, para um total de p + 1 vezes e cada execução testa todos os n elementos do vetor, o que faz com que a complexidade normal seja O(n*p). Entretanto, em relação ao tamanho da entrada, a complexidade em função da entrada, já que p = log x, é O(n*2^x), o que faz do problema pseudo-polinomial.