Problema da Mochila Estendido (com caminhões)

Douglas Pereira Luiz

6 de janeiro de 2025

1 Problema

Considere o seguinte problema: Suponha que uma empresa de entregas possui um conjunto caminhões $T = \{t_1, t_2, ..., t_n\}$. Cada caminhão $t_i \in T$ possui uma capacidade de carga igual a $c_i \in Z^+$. A empresa recebeu demandas de entrega de um conjunto de itens $G = \{g_1, g_2, ..., g_m\}$, no qual cada item g_i possui peso $w_i \in Z^+$ e trará um lucro de $v_i \in Z^+$ na entrega. Sabe-se que é possível que a empresa não entregue todos os itens e que cada caminhão pode ser utilizado uma vez (os caminhões saem simultaneamente para entrega). Deseja-se saber quais itens não foram entregues e o lucro obtido na entrega.

1.1 Decisões de implementação

O Algoritmo 1 apresenta o algoritmo desenvolvido para solucionar o problema da questão. É um algoritmo de programação dinâmica em sua forma recursiva e baseado na solução do Problema da Mochila, encontrando o valor máximo. D é o conjunto de itens incluídos para obtenção do máximo, e para resolver o problema, a solução retorna o conjunto $G - D$, que corresponde aos itens que não foram incluídos nos caminhões. Foi utilizado uma matriz com $2 + |T|$ dimensões inicializada com todos os valores em *null* para memoização. Apesar de que, conceitualmente, a matriz M possui $2 + |T|$ dimensões e acessos na forma

$$M[item][caminhão][restante\ de\ c_1][restante\ de\ c_2], ..., [restante\ de\ c_{|T|}]$$

, a implementação foi feita com uma estrutura de três dimensões e acessos na forma

$$M[item][caminhão][capacidades\ restantes]$$

, com os índices da dimensão correspondente as capacidades restantes sendo tuplas do tipo

$$(restante\ de\ c_1, restante\ de\ c_2, ..., restante\ de\ c_{|T|})$$

O algoritmo foi implementado na linguagem Python3 e utiliza como entrada um arquivo com os conjuntos T, C, G, $W = \{w_1, w_2, ..., w_{|G|}\}$ e $V = \{v_1, v_2, ..., v_{|G|}\}$. O arquivo da implementação se encontra em `3/caminhoes.py` a partir do diretório de entrega (Obs: deixei um arquivo `3/caminhoes_clean.py` que resolve somente o valor máximo, sendo de mais fácil leitura e comparação com a função OPT discutida na parte **(b) da questão**).

Os aquivos de entrada seguem os seguinte formato:

1ª linha: Conjunto T. Nomes dos caminhões separados por espaço;

2ª linha: Conjunto C. Capacidades dos caminhões separadas por espaço;

3ª linha: Conjunto G. Nomes dos items separados por espaço;

4ª linha: Conjunto W. Pesos dos items separados por espaço, sendo equivalente a função w;

5ª linha: Conjunto V. Valores dos items separados por espaço, sendo equivalente a função v.

Existem dois arquivos de instância do problema, *teste1.txt* e *teste2.txt*. Para testar o programa, execute no terminal `python3 3/caminhoes.py 3/teste2.txt` no diretório raiz da entrega.

Algorithm 1 Caminhões

```
1: function OPT(i, j, C, D)
2:     result ← null
3:     if i = −1 then
4:         result ← (0, D)
5:     else
6:         memory ← M[i][j][c_1][c_2]...[c_|T|]
7:         if memory ≠ null then
8:             result ← memory
9:         else if j = −1 then
10:            result ← OPT(i − 1, |C| − 1, C, D)
11:        else if c_j < w(i) then
12:            result ← OPT(i, j − 1, C, D)
13:        else
14:            value′, D′ ← OPT(i − 1, |C|, C com c_j ← c_j − w(i), D)
15:            value′ ← value′ + v(i)
16:            value″, D″ ← OPT(i − 1, |C| − 1, C, D)
17:            if value′ > value″ then
18:                result ← (value′, D′ ∪ G_i)
19:            else
20:                result ← (value″, D″)
21:            end if
22:        end if
23:    end if
24:    M[i][j][c_1][c_2]...[c_|T|] ← result
25:    return result
26: end function
27: C ← {c_1, c_2, ..., c_|T|}
28: maxValue, D ← OPT(|G|, |T|, C, {})
29: notDelivered ← G − D
30: return (maxValue, notDelivered)
```

1.2 Desenvolvimento

O algoritmo se baseia na solução para o problema da mochila como apresentado nas anotações [Silva e Santiago 2024], que segue a seguinte recorrência, OPT_m, onde i é um índice em um conjunto de itens G e p é a capacidade da mochila:

$$OPT_{\mathrm{m}}(i,p) = \begin{cases} 0 & i = 0 \\ OPT_{\mathrm{m}}(i-1,p) & p < w(i) \\ \textbf{\textit{max}} \begin{cases} OPT_m(i-1,p) \\ v(i) + OPT_m(i-1,p-w(i)) \end{cases} & p \geq w(i) \end{cases}$$

O problema enunciado foi resolvido fazendo adaptações na recorrência anterior, resultando na recorrência OPT:

$$OPT(i,j,C) = \begin{cases} 0 & i = 0 \quad (1) \\ OPT(i-1,|C|,C) & j = 0 \quad (2) \\ OPT(i,j-1,C) & c_j < w(i) \quad (3) \\ \textbf{\textit{max}} \begin{cases} OPT(i,j-1,C) \\ v(i) + OPT(i-1,|C|,C \text{ com } c_j \leftarrow c_j - w(i)) \end{cases} & c_j \geq w(i) \quad (4) \end{cases}$$

O conjunto C contém as capacidades de todos os caminhões, sendo $C = \{c_1, c_2, ..., c_n\}$, com $n = |T|$. O conjunto C é usado na recorrência como uma tupla, sendo usado para registrar a ocupação atual dos caminhões, sendo análogo ao valor p em OPT_m. Além de i, que carrega o item sendo avaliado naquela recorrência, é utilizado o valor j, que carrega o índice do caminhão sendo avaliado naquela chamada.

De forma simplificada, as partes correspondentes da recorrência as opções possíveis são:

- 1: não há items para colocar;
- 2: item i não foi colocado em nenhum caminhão, deve-se avaliar o próximo item;
- 3: a capacidade restante do caminhão j não suporta o item i, portanto deve-se avaliar no próximo caminhão;
- 4: o caminhão j suporta o item i, então deve-se comparar entre ele não estar nesse caminhão e ele estar nesse caminhão.

Da mesma forma que a escolha do item, que começa pelo maior índice em G, a escolha do caminhão começa pelo maior índice em C, ou seja $|C|$, sendo o próximo caminhão resultante do decremento de j. Portanto, o valor máximo é obtido pela solução de $OPT(|G|, |C|, C)$

A solução de $OPT(i, j-1, C)$, nas opções 3 e 4, que resolve a adição do item no caminhão seguinte, eventualmente atinge $OPT(i, 0, C)$, resolvendo quando o item i não é colocado em nenhum caminhão. A solução de $v(i) + OPT(i-1, |C|, C \ com \ c_j \leftarrow c_j - w(i))$, na opção 4, garante a avaliação do resultado para os casos em que o item i está no caminhão j, incrementando o valor do item i na solução e avaliando a adição do próximo item, $i-1$, passando por todos os caminhões novamente, por isso a recorrência volta o valor de j para $|C|$. Dessa forma, a recorrência atinge todas as possíveis distribuições de itens para os caminhões.

Para obter os itens não entregues, foi feito uma adaptação na solução para manter controle dos itens entregues no conjunto D, que é passado na recursão. O resultado obtido da chamada de $OPT(|G|, |C|, C, \emptyset)$ contém tanto o valor máximo quanto o conjunto de itens entregues, dessa forma, o conjunto de itens não entregues é igual a $G - D$. A função OPT final é:

$$OPT(i, j, C, D) = \begin{cases} (0, D) & i = 0 \\ OPT(i-1, |C|, C, D) & j = 0 \\ OPT(i, j-1, C, D) & c_j < w(i) \\ \boldsymbol{max} \begin{cases} OPT(i, j-1, C, D) \\ (v(i) + OPT(i-1, |C|, C \ com \ c_j \leftarrow c_j - w(i), D), D \cup G_i) \end{cases} & c_j \geq w(i) \end{cases}$$

Referências

[Silva e Santiago 2024]SILVA, A. G.; SANTIAGO, R. de. *Anotações para a disciplina de Projeto e Análise de Algoritmos*. 2024. https://www.inf.ufsc.br/~r.santiago/downloads/INE410104.pdf. [Online; acessado em 2 de Dezembro de 2024].