A princípio, pode-se perceber a grande similaridade entre o famoso problema da mochila 0-1 e o proposto no projeto em questão, partindo desta ideia, a análise do problema foi feita na mesma lógica e pensamento do seu correlato.

**Estruturação do problema e solução recursiva:**

A grande pergunta que rodeia o desafio é se um item J, escolhido entre 1 e n, deve ou não estar inserido nesse vagão, já que se trata da tentativa de obter o maior lucro possível com um vagão de capacidade W, partindo deste pressuposto pode-se criar uma condição de entrada no vagão para o produto J, se o valor do frete total ,considerando o produto J inserido no vagão, for maior que as possíveis combinações de produtos sem J, então, o produto J estará incluso no vagão, somente, é claro, se o peso Wj não ultrapassar a capacidade da mochila, transformando isso numa chamada recursiva se obtém o código que obtém a resposta da inserção do J, entretanto está faltando os casos bases, que para este problema serão dois, caso o vagão não caiba mais nada, ou seja, W == 0, o programa retorna o valor 0, o outro será, caso não haja mais produtos a serem checados, J == 0, o programa retorna 0.

**Estratégia de armazenar respostas e solução bottom-up:**

Analisando-se as chamadas recursivas facilmente se obtém um padrão de re-cálculo, o algoritmo checa muitos vezes se é vantajoso inserir ou não um produto J no vagão com variações do peso, assumindo que os valores dos pesos são números inteiros, dá pra perceber um padrão e traçar uma maneira de armazenar os valores já computados, exemplo:

* O programa irá checar se um produto J de Wj= 10 cabe em um vagão de peso W=15, neste caso, existem 16 possibilidades de pesos, de 0 até 15, logo, o programa checará se é vantajoso ter o produto J em uma das 16 possibilidades, o algoritmo irá se perguntar, qual o maior valor, o do item anterior a J porém com a mesma capacidade atual, W, ou o valor já computado quando a mochila tinha W - Wj de capacidade no item anterior somado ao valor de J.

partindo da ideia do exemplo, pode-se armazenar os cálculos em uma matriz, onde as linhas seriam respectivamente o item, indo de 0, correspondente ao vagão sem nenhum item, até o valor de Jn, este sendo o último produto na listagem, as colunas correspondem aos possíveis pesos, logo W+1 colunas, ou seja, as respostas podem ser facilmente guardadas numa matriz de n+1 linhas e W+1 colunas.

O algoritmo bottom-up se baseia na ideia de preencher esta matriz até a última verificação, que corresponde a resposta propriamente dita, no problema recursivo começamos do último item caminhando rumo ao primeiro, no algoritmo iterativo os cálculos serão iniciados do caso mais básico, caso base, correspondente ao peso da mochila ou o número de produtos ser 0, partindo daí, a matriz será preenchida da esquerda para direita de cima para baixo, melhor dizendo, quando não há itens, qual o valor máximo nos W+1 possíveis pesos, depois, quando estamos com o produto 1, J=1, qual o valor máximo nos W+1 possíveis pesos, e assim por diante, na última linha e coluna obtemos, “qual a resposta para o máximo valor quando temos o último item e o peso completo da mochila”, que por sua vez se trata do caso inicial do problema recursivo e portanto é a resposta que se deseja, o valor máximo que pode se obter através de uma mochila de capacidade W.

**Listando Itens contidos na resposta:**

Para fazer a listagem dos itens contidos no vagão precisamos verificar quando houve modificações de valor na de uma coluna, isto basicamente indica que o item deve estar inserido em S, solução. Deve-se varrer a matriz de baixo para cima na coluna correspondente a W com uma variável guardando a capacidade total do vagão, Y, quando a condição “matriz[j][capacidade] != matriz[j-1][capacidade]“, ou seja, quando o valor máximo no item J e peso W for diferente do valor contido na linha J-1 na coluna do mesmo peso isso indica que o item J é parte da solução do problema e deve ser adicionada em S, ao adicionar J em S deve-se reduzir o Wj da variável que está guardando a capacidade do vagão e na próxima iteração será verificado na coluna Y.