

Lab 2: Cargo bike courier

20226758 - Jingqi Fan

May 20, 2024

1 Greedy Algorithm

This method employs a greedy algorithm approach, prioritizing items with the highest profit per unit weight until the load limit is reached, thereby maximizing total profits as much as possible. In this Java code, the main purpose of *choice(.)* is to select the optimal combination of items based on their profitability per kilogram, ensuring that the total weight does not exceed the maximum load capacity (*maxQuantity*).

```
public List<ShopItem> choice(.)  
1 public List<ShopItem> choice(List<ShopItem> wishes, float maxQuantity) {  
2     // Sort the wishes list based on profit per kg in descending order  
3     Collections.sort(wishes, new Comparator<ShopItem>() {  
4         @Override  
5         public int compare(ShopItem o1, ShopItem o2) {  
6             return Float.compare(o2.profitPerKg(), o1.profitPerKg());  
7         }  
8     });  
9     List<ShopItem> selectedItems = new ArrayList<>();  
10    float remainingCapacity = maxQuantity;  
11    // Select items based on the sorted order  
12    for (ShopItem item : wishes) {  
13        if (remainingCapacity == 0) {  
14            break;  
15        }  
16        float quantityTaken = Math.min(item.quantity, remainingCapacity);  
17        float profitForTakenQuantity = quantityTaken * item.profitPerKg();  
18        // Add to the list of selected items  
19        selectedItems.add(new ShopItem(item.label, quantityTaken,  
20            profitForTakenQuantity));  
21        remainingCapacity -= quantityTaken;  
22    }  
    return selectedItems; }
```

2 Complexity Analysis

Theorem 1 *The complexity of my algorithm is $\mathcal{O}(n \log(n))$.*

Proof The algorithm can be analysed from three steps. Firstly, I sort the list based on the profit per kilogram. The complexity of sorting is $\mathcal{O}(n \log(n))$. Secondly, I iterate through the sorted list, whose complexity is $\mathcal{O}(n)$. Finally, the algorithm calculate profit per kg and select the item. These operations are done in constant time $\mathcal{O}(1)$. Therefore, the total complexity is $\mathcal{O}(n \log(n))$. \square

3 Optimality Explanation

Theorem 2 *My algorithm is optimal.*

Proof We can abstrat the problem into mathematical version: Define a as the unit price of flour and b as the unit price of sugar. And I also denote transport quality of flour as x and transport quality of sugar is y . According to the requirement of the question, the value ranges of x and y are $15 \leq x \leq 20.5$ and $4.5 \leq y \leq 10$. Therefore, our goal is to find the maximum value:

$$V = \max\{ax + by\} \text{ s.t. } 15 \leq x \leq 20.5, 4.5 \leq y \leq 10 \text{ and } x + y = 25 \quad (1)$$

where $a = \frac{140}{20.5} = \frac{280}{41}$ and $b = \frac{80}{10} = 8$.

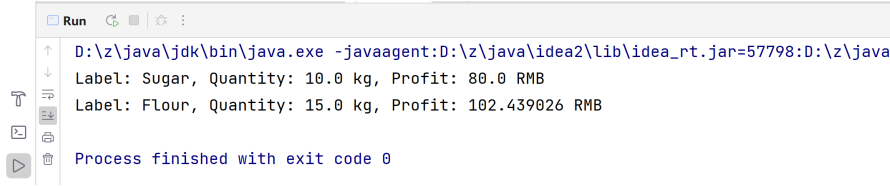
Plug $\{a, b, y = 25 - x\}$ into (1) and we have

$$\begin{aligned} V &= \max\left\{\frac{280}{41}x + 200 - 8x\right\} \\ &= \max\left\{200 - \frac{48}{41}x\right\} \end{aligned} \quad (2)$$

Becasue the goal is to maximize (2), we plug in $x_{\min} = 10.5$ and get the result

$$\begin{aligned} V &= 182.439026 \\ x &= 10.5 \\ y &= 15 \end{aligned} \quad (3)$$

which is the same as my algorithm output:



```

Run
D:\z\java\jdk\bin\java.exe -javaagent:D:\z\java\idea2\lib\idea_rt.jar=57798:D:\z\java
Label: Sugar, Quantity: 10.0 kg, Profit: 80.0 RMB
Label: Flour, Quantity: 15.0 kg, Profit: 182.439026 RMB
Process finished with exit code 0

```

Figure 1: the result of my algorithm

More generally, suppose that we have n items $\{x_1, x_2, \dots, x_n\}$. The unit prices of there items are $\{p_1, p_2, \dots, p_n\}$ in which we define $p_1 \geq p_n \geq \dots \geq p_n$. Then our goal is

$$V = \max \sum_{i=1}^n p_i x_i \quad (4)$$

Because the situation of this problem has no bonus for special combination of items, to maximize (4), what we need do is maximizing item whose profit is highest. When the most profitable item exceed the max quantity, we begin to select the item with the second highest value. Thus, the idea is the same as my greedy algorithm and it is optimal. □

4 Tpye Change from Float to Int

Theorem 3 *If we are only allowed to transport integer quantities, then greedy algorithm is still optimal.*

Proof As said before, our goal is to maximize the total value $V = \max \sum_{i=1}^n p_i x_i$. While we are only allowed to transport integer quantities, we can still select items with highest unit price firstly with its maximum integer quantities. In other words, no matter whether the required value is float or integer, the greedy algorithm which selects the item with highest unit value is optimal. □