Jacob Curtis

10/28/25

Roaster Problem Report

This assignment overall was both easy and difficult. There were certain parts that I had no problems with, ie; setting up the warehouse class, strategy class, and blend linked list. The difficult portion of the assignment was formatting the data from the .csv to the dictionary for ease of access. Also, removing the “ and ‘ in the strings. Formatting headers were also important, as if the headers are incorrectly read, this will result in potential gaps in the remaining portion of the row.

With the opportunity to incorporate any algorithm as we please, this served as a challenge, but was a thrilling experience. Since there are no direct approaches to this assignment, this serves as an uncertainty. This assignment indeed served its purpose as a midterm expectation. This assignment allowed us as a student to combine our knowledge in concepts learned from class. From linked list, to lambdas, to polymorphism, we were able to incorporate all such in the algorithm

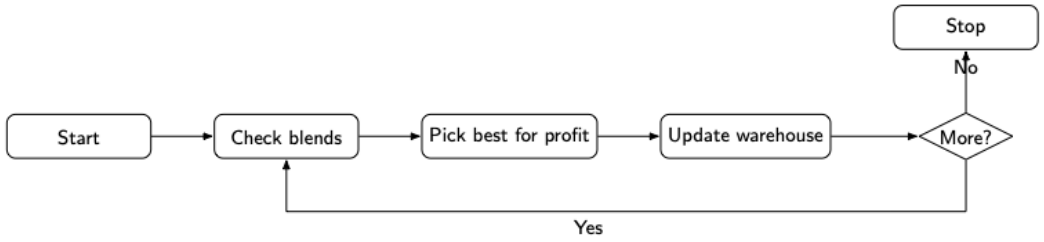
The algorithm can be expressed in a step format, therefore, let’s look at this as a solution of steps:

After reading in the data:

1. Assign the blends to a linked list
2. Reference the blends for the greedy algorithm
3. Compute the weighted income efficiency-score for each blend
4. Greedily pick the most efficient blend based on efficiency score
5. Produce this batch
6. Repeat steps 3-5
7. Report the results

Essentially we are given a list of blends, with corresponding bean requirement data. This data must be stored in an instance where we can constantly access and modify the existing totals of beans. The algorithm will deduct beans used and increment the blend name of the blend that was made. Specifically on the blends made, this is determined by the best blend given its revenue aspect, and its total bean quantity.

We continue through our bean list and at each iteration, we calculate this income-efficiency metric. We pick the best blend in comparison to its neighbors, the blend that has the best income efficiency, and make this blend.



The weighted score can be expressed formulaically as:

$$\text{WeightedScore}(b_i) = \frac{\text{IncomeEfficiency}(b_i)}{E_i w_E + H_i w_H + R_i w_R}$$

Where each bean is multiplied by its calculated weight, and summed.

This summation is divided by the Income Efficiency score.

Where Income Efficiency is:

$$\text{IncomeEfficiency}(b_i) = \frac{\text{Revenue per Batch}(b_i)}{\text{Total Bean Usage}(b_i)} = \frac{P_i \times 100}{E_i + H_i + R_i}$$

P is the price per blend times the batch size of 100. This is all over the sum of beans used in this blend, including Ethiopian, Honduras and Rwanda beans.

3.

The method I chose was the greedy algorithm, where a blend is picked based on an income efficiency calculation, in reference to its neighboring blends. This greedy algorithm is adaptive, given the weights for each bean. The weights are calculated based

on the usage of each bean, where the formula is 1 / remaining bean. The remaining bean amount is stored after each production of the blend. Upon a for loop, if a blend is a current “best blend” this blend will be picked for production, otherwise keep iterating until the best blend is found. The algorithm continues until we cannot make further of that blend via the blend bean requirements.

How do we know that this algorithm is optimal? Essentially every blend is evaluated based on the income efficiency calculation. It chooses the current best blend in a temporary array. The income efficiency for each blend changes upon the production of the blend. The algorithm evaluates and produces blends given this income efficiency score, which considers the scarcity of the bean. In essence, this algorithm tries every blend, and determines based on its prompted score.

The iterations

Iter.	Blend	E Left	H Left	R Left	Weights (w_E, w_H, w_R)	Income ($)	Income Eff.	Weighted Eff. Score
1	B1	9940	14985	6975	(0.000100, 0.000067, 0.000143)	1,500	15.0	1418.9
70	B1	5800	14400	5250	(0.000172, 0.000069, 0.000190)	105,000	15.0	1380.4
140	B3	7100	8750	3500	(0.000141, 0.000114, 0.000286)	198,000	13.0	1305.7
200	B3	3100	4750	1750	(0.000323, 0.000210, 0.000571)	309,000	13.0	1287.3
280	B3	550	3450	0	(0.001818, 0.000290, ∞)	378,000	13.0	1265.0

The results are as follows:

Batches produced:

Coffee Blend 1: Blunderbuss: 70

Coffee Blend 3: Bongos: 210

Coffee Blend 2: Tomfoolery: 0

Total Batches of Blends Made: 280

Remaining Beans:

Ethiopia: 550.0 lbs

Honduras: 3450.0 lbs

Rwanda: 0.0 lbs

Blend 1 Income - $105,000

Blend 2 Income - $0

Blend 3 Income - $273,000

Total Maximized Income: 378,000

Therefore, the maximized total income was $378,000

This is determined as a good outcome as each bean has an assigned weight and the best blend is decided based on the income efficiency calculation. The income efficiency calculation incorporates these weights as:

Weighted_bean = 1 / the remaining number of bean

Overall score = blend's income efficiency / overall weights on beans

The overall score is the evaluated metric when choosing which blend to produce

4.

The grade for this I believe should be an 85 as the greedy algorithm that is adaptive could include improvements. Such as, incorporating other strategies such as minimizing Rwanda, Ethiopia, and Honduras beans. For this project, I worked diligently on adjusting the algorithm. Upon creating the baseline files such as warehouse.py, driver.py, blend.py, I utilized the logic from these files in creating, within my research, the most optimal algorithm for maximizing income. The weights was an addition I made after realizing that blend 3 was a more optimal blend to make, as this blend required less Ethiopia beans and prompted $2 less in price/per lb. Therefore, after including weights into the greedy algorithm, the algorithm became adaptive. This resulted in a higher maximized total

income. The original algorithm without weights prompted a total maximized income of 306,300. After implementing and recognizing the blend 3 optimization, the total income increased to 378,000.

5.

Given the leftover beans:

Ethiopia: 550.0 lbs

Honduras: 3450.9 lbs

Rwanda: 0.0 lbs

It would be optimal to purchase 500lbs of Rwanda, as we can make the blend 1 and blend 2, where additional income amounts to 30,000.

The breakdown of the additional 500 lb of Rwanda bean:

500/25 = 20 blends of Blend 1

20 Blends * 100 lbs * 15 price per lb = $30,000

Whereas, purchasing 3,000 lbs of Ethiopia would not change the overall maximized total income. Same for purchasing 1,200 lb of Honduras, where 3,450.90 lbs of Honduras beans are remaining. Rwanda is therefore the only bean worth purchasing when considering the adaptive greedy algorithm.

Works Cited

BeyondVerse. “Greedy Algorithms: Strategies for Optimization.” *Medium*, 25 Nov. 2023,

The Python Lab. “Building a Greedy Algorithm in Python: Finding Optimal Solutions at Each Step.” *Medium*, 2 Feb. 2023, thepythonlab.medium.com/building-a-greedy-algorithm-in-python-finding-optimal-solutions-at-each-step-6ffdaae38626.

Nakhleh, Luay. *Weighted Graphs and Greedy Algorithms.* Rice University, https://www.cs.rice.edu/~nakhleh/COMP182/WeightedGraphsAndGreedyAlgorithms.pdf

Cormen, Thomas H., Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms.* 4th ed., The MIT Press, 2022..