DAT561: Midterm Project:

Smart Logistic Warehouse System in Amazon

Amazon Inc. is an American multinational retail corporation that operates a chain of online hypermarkets with huge warehouses in the United States. Amazon warehouses are located throughout the country to ship out products and provide on-time delivery for customers.

Example of one Amazon warehouse

Recently, Amazon has been planning a 25-day countdown promotion for Christmas, and David King, a manager of Amazon Company, must make sure that the stored products in each warehouse are proper and sufficient. After consideration, David breaks this project into two parts:

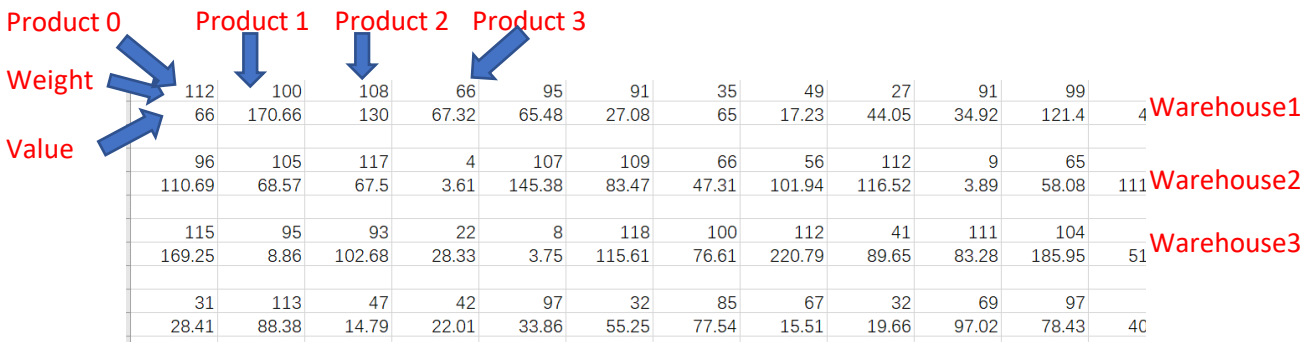
1. Selecting products for each warehouse before the promotion starts
2. Simulating cross-warehouse-transshipment solution for possible product shortage problem for a random warehouse after the promotion starts

Working as a consulting team, you would provide your analysis to David, and all the data you need is available in the “Products.csv” file.

Here is some information about this dataset:

Each column contains the weight and value of each product, and there is a total of 50 available product options (Product No. 0 - Product No. 49) that can be selected for one warehouse.

Every two rows contain all product information in one warehouse, and there are 300 warehouses (Warehouse No. 0 - Warehouse No. 299) in total.



112	100	108	66	95	91	35	49	27	91	99	
66	170.66	130	67.32	65.48	27.08	65	17.23	44.05	34.92	121.4	4
96	105	117	4	107	109	66	56	112	9	65	
110.69	68.57	67.5	3.61	145.38	83.47	47.31	101.94	116.52	3.89	58.08	111
115	95	93	22	8	118	100	112	41	111	104	
169.25	8.86	102.68	28.33	3.75	115.61	76.61	220.79	89.65	83.28	185.95	51
31	113	47	42	97	32	85	67	32	69	97	
28.41	88.38	14.79	22.01	33.86	55.25	77.54	15.51	19.66	97.02	78.43	40

For each warehouse, there are four variables:

1. *n* - total number of products chosen to be stored in the warehouse
2. *C* - total weight capacity of the warehouse
3. V_i - the value of available product *i*
4. W_i - the weight of available product *i*

Specifically, for each warehouse, there is a weight capacity (***C***) of 850, meaning that the total weight of n products selected to be stored in a warehouse should be less than or equal to 850. Each available product *i* has its own weight ($\boldsymbol{W_i}$) and value ($\boldsymbol{V_i}$). Besides, the product information for each warehouse is different.

Warehouses can store all boxes within **Capacity C**.

There are **n** products available for selection.

Each product i: **Value Vi** and **Weight Wi**

$$C \geq \sum W_{i(stored)}$$

Problem 1: The Estimated Value for Each Warehouse

In this part, David wants to select products for each warehouse. He will choose as many products as possible with the top "Value Weight Ratios" before reaching the capacity of each warehouse (850).

$$Value\_Weight\ Ratio = \frac{Value\ of\ a\ Product}{Weight\ of\ a\ Product}$$

That is, he will first choose the product with the highest "Value_Weight Ratio" to store in the warehouse, then select the product with the second-highest "Value_Weight Ratio" to store, and so on until the warehouse reaches its capacity.

Here is a simplified example for one warehouse, assuming its capacity is 265, and 10 products could be selected. The weights and values of these products are listed below:

Product No	0	1	2	3	4	5	6	7	8	9
Weight	112	67	94	58	47	9	98	31	81	12
Value	195.71	62.59	105.44	57.15	64.21	8.79	71.14	11.19	62.89	1.64

First of all, calculate the Value Weight Ratio for each product.

Product No	0	1	2	3	4	5	6	7	8	9
Value/Weight	1.7474	0.9342	1.1217	0.9853	1.3662	0.9767	0.7259	0.3610	0.7764	0.1367

Secondly, the products should be sorted based on the Value-Weight ratio in ascending order.

Thirdly, start selecting products, and at the same time, look at the accumulated weight of products to ensure the accumulated weight does not exceed the warehouse weight capacity. With the capacity of this warehouse (265), after we select products 0, 4, and 2, the accumulated weight approaches 253, and only 12 are left for the next product. When we choose the fourth product, we select product 5 rather than product 3 because the weight of product 3 (58) is larger than the remaining capacity (12). Moreover, this warehouse has no space to contain extra products. As a result, we finally have products 0, 2, 4, and 5 stored in this warehouse.

Lastly, calculate the corresponding estimated value of the warehouse, which, in this case, is 374.15.

Product No	Weight	Value	Ratio	cum_weight	cum_value
9	12	1.64	0.1367	262	374.15
7	31	11.19	0.3610	262	374.15
6	98	71.14	0.7259	262	374.15
8	81	62.89	0.7764	262	374.15
1	67	62.59	0.9342	262	374.15
5	**9**	**8.79**	**0.9767**	**262**	**374.15**
3	58	57.15	0.9853	253	365.36
2	**94**	**105.44**	**1.1217**	**253**	**365.36**
4	**47**	**64.21**	**1.3662**	**159**	**259.92**
0	**112**	**195.71**	**1.7474**	**112**	**195.71**

You should perform this step with python codes!

Now, you would try to help David with his product selection problem for those 300 warehouses (each capacity is 850) with the dataset "Products.csv". Calculating the estimated total value and the accumulated weight of each 300 warehouses will be helpful for the later decision of the cross-warehouse-transshipment solution.

Notes:

1. Please do not use other packages to read the CSV file.

2. You should perform all the steps above with python codes rather than a spreadsheet
3. Sometimes, when the warehouse is nearly full, the next product with the next highest ratio cannot be put in the warehouse, but there are still chances that one or more remaining products can be placed in the warehouse. Make sure you take every product into consideration; otherwise, you may have the risk of missing products.
4. You canNOT use the dynamic programming method in this problem; otherwise, you may get 0.

Problem 2: Top Alternative Selections

Thank you for your great job helping David with the warehouse storage arrangement! In this part, he would like to simulate the situation when some products in the warehouses may run out during the promotion period. Please treat this part as an independent part, which means the settings in Problem 1 don't apply here.

David manages the following types of products: Product A, Product B, and Product C. Each warehouse stores these three types of products. A warehouse is expected to run out of these three products during promotion. Cross-region shipment from other warehouses (let's call them "Helpers") is necessary to keep the promotion activities and ensure on-time delivery in one region.

Before seeking help from those Helpers, David must figure out which Helpers he could choose. For the Helpers selection, David would choose 3 Helpers for each product. For each product, the three helpers have the top 3 highest "Value_per_Weight" ratios. It's acceptable to recommend more than 3 Helpers for each product because two or more Helpers may have the same "Value_per_Weight" ratio.

$$Value\_per\_Weight = \frac{Total\_Value_i}{Total\_Weight_i} - Distance \times Transportation\_Cost$$

1. *Total_Value: total value of the product i stored in one warehouse*
2. *Total_Weight: total weight of the product i stored in one warehouse*
3. *Distance: the distance, generated by you, between the out-of-stock warehouse and the Helpers*
4. *Transportation_Cost:* 0.012 per distance

Cross-warehouse-transshipment begins with the warehouse that has three products out-of-stock (total number of the stored products i is 0), and this warehouse would contact Helper to see if help is available. Once the Helper agrees to do the Cross-warehouse-transshipment, it will transport all product i in its warehouse. In this case, David assumes that Helpers have not sold any products i. In other words, the total number of stored product i in each Helper remains the same as before the promotion started. Since the distances between this product-shortage warehouse and the Helpers are different, the transportation fee will be correspondingly different. By calculating the "Value_per_Weight," David could find Helpers with the top 3 highest "Value_per_Weight" after considering transportation costs (The cost of a one-way trip from the Helper to the product-shortage warehouse).

Step 1: First, generate a distance matrix among the 150 warehouses. Each distance can be generated using a normal distribution with a mean of 500 and a standard deviation of 150. (Please make sure that each generated distance is positive). Below is an example of a 10*10 distance matrix.

	0	1	2	3	4	5	6	7	8	9
0	0	551	625	496	749	578	573	581	459	400
1	386	0	709	503	450	637	331	623	622	482
2	571	491	0	550	560	250	318	276	507	759
3	517	372	282	0	555	644	734	306	493	358
4	373	574	491	493	0	449	527	577	246	602
5	412	567	163	258	595	0	617	529	364	520
6	346	827	321	442	419	703	0	166	637	715
7	543	198	741	602	608	472	639	0	427	389
8	431	580	637	410	892	387	373	80	0	340
9	655	578	375	460	848	487	400	577	627	0

You must create your own distance matrix with a size of 150*150. Be careful, all the distances generated should be positive numbers and rounded to an integer using round (). After successfully developing the distance matrix, please write it to a new CSV file called “Distances.csv”, without including row or column indices.

Notes:

1. Please do not use any packages to write the CSV file.
2. For better understanding, we included the warehouse index in the screenshot, but the distance matrix you generated does not need to include the index.

Step 2: Let's generate product data.

First of all, David would randomly choose a warehouse (from Warehouse No.0 to Warehouse No.149) and assume it would face out-of-stock problems in this simulation.

Secondly, for each warehouse (including the out-of-stock warehouse), generate random values for the following attributes for three types of products (A, B, and C):

1. Total_Value: Random integer between 3000 and 15000 for Product A, 4000 to 20000 for Product B, and 2000 to 12000 for Product C.

2. Total_Weight: Random integer between 500 and 3000 for Product A, 1000 to 5000 for Product B, and 700 to 3500 for Product C.

In the next step, you will use these values to calculate the Value_per_Weight for each Helper warehouse. Below is a sample of 10 warehouses:

	0	1	2	3	4	5	6	7	8	9
Total_Value_A	13490	12667	3532	6834	13075	10834	11666	11417	14965	3168
Total_Weight_A	1215	1673	1767	2018	867	681	1551	1148	1632	2444
Total_Value_B	11570	8543	17286	19192	10617	19720	5747	6274	11301	9717
Total_Weight_B	4993	1726	4584	1843	1425	1824	4520	4990	4945	2461
Total_Value_C	8374	9964	9649	10225	9119	8770	7326	6804	5004	4882
Total_Weight_C	3121	2401	3259	3242	2421	2163	742	2098	2411	2132

Step 3: Now, you could calculate the "Value_per_Weight" and help David decide the 3 Helpers for each product with the top 3 highest "Value_per_Weight".

Firstly, based on the out-of-stock warehouse he picks in Step 2, you need to find the corresponding distance between this warehouse and each of the other Helpers from the distance matrix you generated in Step 1. In addition, you also need to find the corresponding total value and the total weight of each of the three products stored in each Helper.

Thirdly, calculate the "Value_per_Weight" ratio for each product in each Helper, sort the ratio in descending order, and choose the top helpers with the three highest "Value_per_Weight" ratios for each product. Recall this formula:

$$Value\_per\_Weight = \frac{Total\_Value_i}{Total\_Weight_i} - Distance \times Transportation\_Cost$$

5. *Total_Value: total value of the product i stored in one warehouse*
6. *Total_Weight: total weight of the product i stored in one warehouse*
7. *Distance: the distance, generated by you, between the out-of-stock warehouse and the Helpers*
8. *Transportation_Cost:* 0.012 per distance

Here is a simplified example. Suppose now there are ten warehouses in total, and David needs Helpers with the top 3 "Value_per_Weight" ratios. He assumes Product A, B, and C in warehouse No. 5 are out-of-stock, and as a result, the total number of all three products in it becomes 0, while the rest of the warehouses (Helpers) remain with the same amount of product as you generated in Step 2. The corresponding distance data is located in the 6th row of the distance matrix you have generated in Step 1 (the row starts with index 5, highlighted by the red rectangle below).

	0	1	2	3	4	5	6	7	8	9
0	0	551	625	496	749	578	573	581	459	400
1	386	0	709	503	450	637	331	623	622	482
2	571	491	0	550	560	250	318	276	507	759
3	517	372	282	0	555	644	734	306	493	358
4	373	574	491	493	0	449	527	577	246	602
5	412	567	163	258	595	0	617	529	364	520
6	346	827	321	442	419	703	0	166	637	715
7	543	198	741	602	608	472	639	0	427	389
8	431	580	637	410	892	387	373	80	0	340
9	655	578	375	460	848	487	400	577	627	0

Finding the corresponding distance

Take product A as an example (the same for products B and C); after calculating the "Value_per_Weight" ratio for each helper, you would sort those Helpers and return the top 3 "Value_per_Weight" and the corresponding index of the Helpers. (It is important you show the corresponding relationship between the "Value-per-Weight" and the index of the Helpers!)

=P4/Q4-R4*R2

	Product A		Transportation_Cost	0.012
	Total_Value_A	Total_Weight_A	Distance	Value_per_Weight
0	13490	1215	412	6.1589
1	12667	1673	567	0.7674
2	3532	1767	163	0.0429
3	6834	2018	258	0.2905
4	13075	867	595	7.9407
5			0	
6	11666	1551	617	0.1176
7	11417	1148	529	3.5971
8	14965	1632	364	4.8017
9	3168	2444	520	-4.9438

You should perform this step with python codes!

The logic of how you calculate the "Value_per_Weight"

Product A	
	Value_per_Weight
4	7.9407
0	6.1589
8	4.8017
7	3.5971
1	0.7674
3	0.2905
6	0.1176
2	0.0429
9	-4.9438
5	

You should perform this step with python codes!

Sorted result

(You should use the code to show this logic instead of Excel; you do not need to generate a file here.)

Notes:

1. You are NOT allowed to use dynamic programming in this problem. Otherwise, your project may receive 0 points.
2. In this problem, there is a chance that the "Value_per_Weight" of product i in the two warehouses is the same. Return the top 3 Value_per_Weight, the distance to the out-of-stock warehouse, and if multiple warehouses have the same Value_per_Weight, determine those warehouses in the output.

Submission of the Project

Your final submission will contain two files:

1. The first would be the "FL24_Midterm.ipynb" notebook. You need to provide the Python code as well as your answers. We strongly recommend that you explain critical steps in the code so that we can understand what you were trying to do when something goes wrong.
2. The second is the “Distances.csv” file.

Note: Please submit all files (Distances.csv, and FL24_Midterm.ipynb files) as one zip file on Canvas. Please add all teammates' IDs to the zip file's name. For example, 14325_34672_12345.zip.

Grading

Total - 100 points

10 points: towards how good your solution is.

We will check how you write the code and solve the problem and how efficient and optimized your code is. Specifically, your code will be graded based on the following factors:

1. **Correctness**: The code should produce the expected outputs and solve the problem correctly.
2. **Readability**: The code should be easy to read and understand with added comments.
3. **Structure and organization**: The code should be well-structured and organized. It should contain functions to modularize the code. Functions must have clear responsibilities and be appropriately organized.
4. **Documentation and comments**: The code should have clear and concise documentation to explain its purpose, inputs, outputs, and any important details.
5. **Efficiency**: The code should be efficient in terms of time and space complexity.

90 points:

The Estimated Value for Each Warehouse – 30 points

The 150*150 distance matrix generated – 30 points

Top 3 Helper Warehouses Chosen for each product - 30 points

- Your code needs to pass the auto-grader for us to see whether it works.
- We will look at your logic and whether your code is working.
- We will look at whether you submitted all files correctly.
- We will examine how effectively you solved the problems (functions, global scobe, etc.).