Warehouse Item Placement Optimization

Less Walking, More Working

PRESENTED BY BLEST (T02A)

Overview of Problem

Warehouse item placement often fails to reflect actual order patterns, leading to excessive worker travel, inefficiency, and fatigue. Optimizing the layout based on item demand and co-purchase relationships can reduce walking time and save the effort of handling.

Glossary

$I = \{i_1, i_2, \ldots, i_m\}$	Set of item types.
$A = \{a_1, a_2, \ldots, a_m\}$	Amount that the warehouse have to store for each item type.
$B = \{b_1, b_2, \ldots, b_m\}$	Set of blocks in the warehouse.
$\mathcal{O} = \{O_1, O_2, \ldots, O_n\}$	Set of customer orders. $each\ O_k \subseteq I$
$D = \{d_1, d_2, \ldots, d_m\}$	Set of item demand frequency from the order.
C	Block capacity.
k_i	The number of blocks needed to store item i. (consider quantity&size)
$Dist(b_1, b_2)$	Manhattan distance between two locations or blocks.
$co(i, j)$	The co-occurrence or relationship between items i and j.
CO	Items co-occurrence matrix
$G = (V, E)$	Graph representing the warehouse layout.

Glossary

$PPS(i)$	Product layout Priority Score : The priority score for item i.
w_{freq}, w_{cooc}	Weighting factors that balance the importance of "frequency" and "co-occurrence."
Depot	Entrance of the warehouse.
Block	The place in the warehouse that store item.
Demand	The frequency that the item appear in the order.
Distance	Manhattan Distance, assuming that the warehouse layout is a grid.
Junction node	A vertex in the graph that is not a block, use to find walking distance.
Candidate block	Wording to explain and shorted the algorithm.
Unclassified items	Wording to explain and shorted the algorithm.

Glossary

$LCS(i, b)$	The cost score of placing item i into empty location b.
$w_{depot}, w_{affinity}$	Weights to balance the two objectives.
$Dist(Depot, b)$	The distance from the entrance to location b.
I_{placed}	The set of items that have already been assigned a location.
Traveling distance	Total walking distance to complete all order.
Handling effort	Evaluation score to put heavier item closer to the depot.
w_i	The weight of item.

Computational Objective

We assign items to warehouse blocks to minimize the seasonal cost of total walking distance and weight-handling effort.

Inputs

1. **Seasonal customer orders** from the previous year (or any time period that the warehouse satisfy) that includes CustomerID, ItemID, and Amount for each item.
2. **List of all the items type** in the warehouse.
3. List of item **weight**.
4. List of item **size(Volume)**.
5. **Total amount sale** from the last year season.

$$\mathcal{O} = \{O_1, O_2, \ldots, O_n\}$$

$$I = \{i_1, i_2, \ldots, i_m\}$$

$$W = \{w_1, w_2, \ldots, w_m\}$$

$$V = \{v_1, v_2, \ldots, v_m\}$$

$$A = \{a_1, a_2, \ldots, a_m\}$$

Date	CustomerID	ItemID	Amount
2024-05-02	MKV-1	038KC68	2.00
2024-05-02	SMN	06381-KVB-900	2.00
2024-05-02	NMM	06381-KVB-900	10.00
2024-05-02	PTA	06381-KVB-900	2.00
2024-05-02	JL	1/4	800.00
2024-05-02	JKC	1200	40.00
2024-05-02	SKB	6203C3 KU	500.00
2024-05-02	SKB	6303CN	400.00
2024-05-02	SKB	6304(OEM)	350.00

Input example : List of customer orders

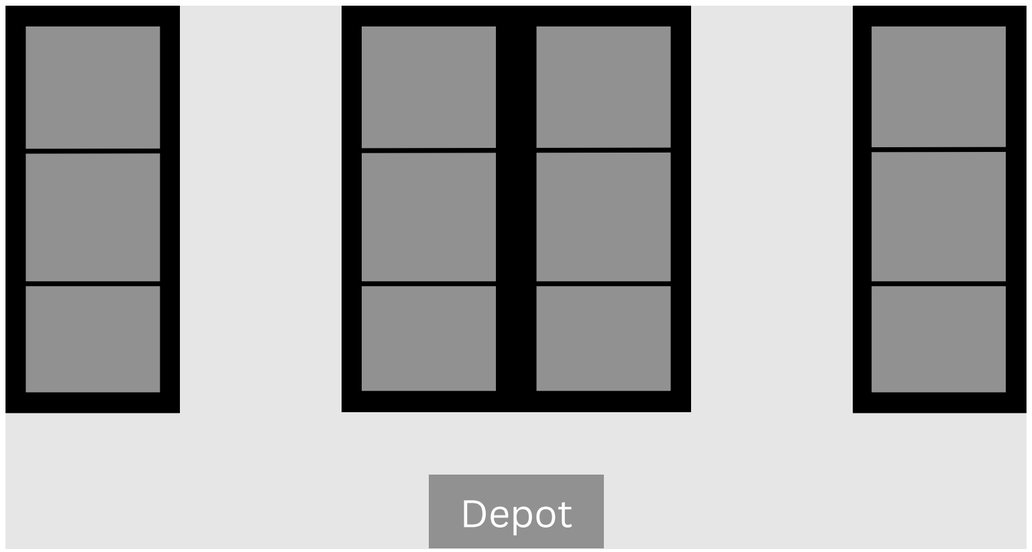
item	total amount sale in last summer
A	546
B	214
C	92
D	564
E	242
F	164

Input example :Total amount sale in last summer

Inputs

6. **Block capacity** (Total item volume that each block can store). C

8. **Warehouse plan**, from top view, showing each block and the walk path clearly.

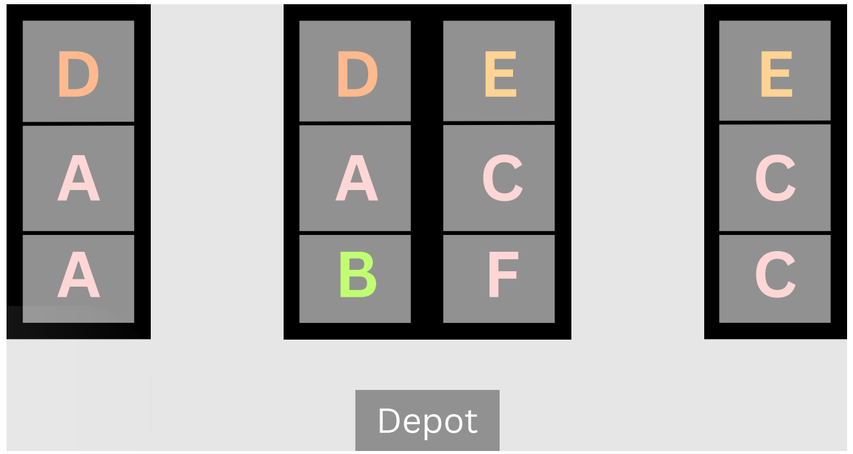


Input example : warehouse plan

Outputs

Items placement plan in the warehouse for the season that will minimize walking distance and satisfy all constraints.

Example output



Total walking distance = 140 meters
Handling Effort = 500 Kg*m

Evaluation

We evaluate the layout using two metrics:

1. **Total Walking Distance**:

For each order, the worker starts at the entrance, walks to the furthest item along a walk path, then picks the remaining items on the way back.

- if the order spans multiple paths, the worker returns to the entrance after each path to drop off items before continuing.

2. **Handling Effort**:

$$\text{Handling Effort} = \sum_{i \in I} w_i \cdot a_i \cdot Dist(Depot, i)$$

Defined as the sum of each item's weight multiplied by item’s amount and the distance to its assigned block.

Constraints

- **Unique Assignment Constraint:** Each block $b \in B$ can be assigned to at most one item type i in I.
- **Storage Requirement Constraint:** Each item i in I must be assigned exactly its required number of blocks, k_i and their total volume cannot exceed the block's maximum volume capacity V_b .
- **Fixed Location Constraint:** The set of storage blocks B, including their total number and physical locations, is predefined and static.
- **Complete Assignment Constraint:** Every item type i in I must be fully assigned to its required set of blocks.

Assumption

- **Distance Metric:**

  The travel distance between any two locations is calculated using the Manhattan distance.
- **No Congestion:**

  The model assumes there are no worker-related delays, such as congestion or route conflicts.
- **Sufficient Inventory:**

  The total storage capacity of the warehouse (in terms of both volume and weight) is sufficient to accommodate all items requiring storage.
- **Sufficient Data:**

  The warehouse can provide us all the inputs. Ex. the physical attributes of each item i, including its size and volume.
- **Dynamic Demand Model:**

  The model assumed that item demand follows significant seasons patterns and assumed that future order patterns will follow similar seasons cycles.

Optimization Criteria

Primary Objective: Minimize Total Walking Distance:

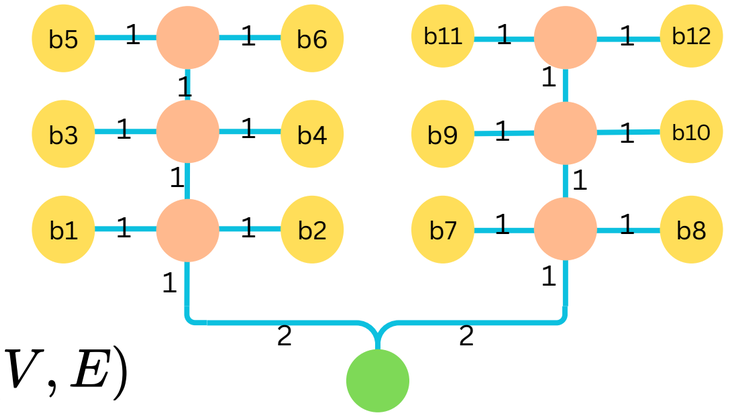
The fundamental goal is to find an assignment for a specific seasons that minimizes the overall traveling distance to fulfill all orders Os.

Secondary Objective: Minimize handling effort:

Since items vary in weight, heavy items should be carried for the shortest distance possible. This objective aims to reduce the physical effort and time required for workers to complete the orders.

Algorithm: Preprocess

1. **Find Item demand list** from order, which is the amount of time that the item appear in the order. $D = \{d_1, d_2, \ldots, d_m\}$
2. Find I**tem co-occurrence matrix** from order. CO
3. **Create a graph** G of the warehouse plan that has blocks (and junction node if needed) as vertex and walk path as edge put all blocks in a list B .
4. **Calculate item count** k_i (number of blocks needed for that item). $k_i = \dfrac{a_i \cdot v_i}{C}$
5. **Prepare the item list** I so that the item type appear in the list k_i times.



$G = (V, E)$

Algorithm: Before Greedy loops

Defining some formula

//Step 1 Decide a formula for picking items

- Design a formula;

if (higher demand and higher correlation)

the higher the score

$$PPS(i) = w_{\text{freq}} \cdot \frac{d_i \cdot w_i}{\max\limits_{j \in I}(d_j \cdot w_j)} + w_{\text{cooc}} \cdot \frac{\sum\limits_{j \in I_{placed}} co(i,j)}{\max\limits_{k \in I} \sum\limits_{j \in I, j \neq k} co(k,j)}$$

//Step 2 Decide a formula for prioritizing/ placing items

- Design a formula;

if (closer /higher correlation)

the lower the score

$$LCS(i,b) = w_{\text{depot}} \cdot \frac{\text{Dist}(\text{Depot}, b) \cdot w_i}{k_i} + w_{\text{affinity}} \cdot \sum_{j \in I_{\text{placed}}} co(i,j) \cdot \text{Dist}(b, b'_j)$$

$\max\limits_{j \in I}(d_j)$	: The **highest demand frequency** among all items. (This is done to ensure that the demand quantity scores of all items are **compared on the same scale**)
$\max\limits_{k \in I}\left(\sum\limits_{l \in I, l \neq k} co(k,l)\right)$	: The **highest total co-occurrence value** among all items.
$\sum\limits_{j \in I_{placed}} co(i,j)$	: The sum of item i's **co-occurrence** with all other placed items.
$LCS(i,b)$	: The cost score of placing item i into empty location b.
$w_{depot}, w_{affinity}$	: Weights to balance the two objectives.
$Dist(Depot, b)$	:The distance from the entrance to location b.
k_i	: the number of blocks required for each item.

Algorithm: High Level

Dynamic Greedy Algorithm:

- **Quantifies item's individual popularity, weight and its relationships** and **blocks' placing cost every loop.**
- **Choose the most important item** and put it in **the most convenient block dynamically**.

$$PPS(i) = w_{\text{freq}} \cdot \frac{d_i \cdot w_i}{\max\limits_{j \in I}(d_j \cdot w_j)} + w_{\text{cooc}} \cdot \frac{\sum\limits_{j \in I_{placed}} co(i,j)}{\max\limits_{k \in I} \sum\limits_{j \in I, j \neq k} co(k,j)}$$

$$LCS(i,b) = w_{\text{depot}} \cdot \frac{\text{Dist}(\text{Depot}, b) \cdot w_i}{k_i} + w_{\text{affinity}} \cdot \sum_{j \in I_{\text{placed}}} co(i,j) \cdot \text{Dist}(b, b'_j)$$

Algorithm: Greedy loops

while (items have not been fully allocated)**{**

- For each **unclassified item** in list I , **Calculate PPS**.
- **Select** the **unclassified item** type **i** from I with the **highest PPS score**.
- For each available **candidate block** ***b*** **,** **Calculate LCS**.
- **Select** the block with the **lowest LCS scores**.
- **Assign** this block **b** to item **i**
- **Remove** the selected block **b** from the list B.
- **Remove** the selected item **i** from the list I .

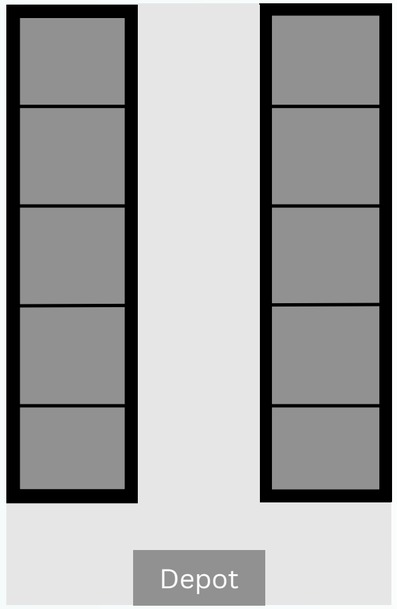
}

End while

candidate blocks = blocks that have **not been deleted from the list.**

Unclassified items = items that are **not allocated** in the ranked list of the current iteration

Problem Instance 1



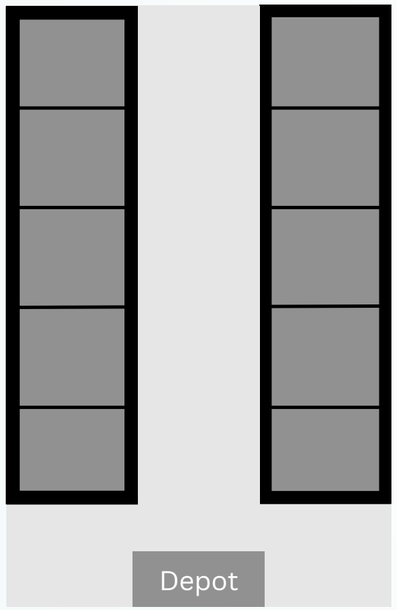
This example is designed to show that even with a simple warehouse layout, item placement based on relationships still matters.

work flow：Input → Preprocess → Running Greedy Loop → Output → Test Layout

Problem Instance 1

Inputs

Orders $\mathcal{O}$:



Warehouse Plan

C = 60 Units

Custom	ItemID	amount
p1	A	10
p1	B	7
p2	A	11
p2	C	3
p3	B	8
p3	C	2
p3	F	2
p4	D	25
p4	E	16
p5	A	9
p5	F	4
p6	C	1
p6	D	13
p7	A	8
p7	B	5
p7	D	16
p8	E	4
p8	F	3
p9	A	6
p9	C	1
p9	F	6
p10	B	3
p10	E	8

total oders for a month last summer

Items(long-term) : $I = \{A, B, C, D, E, F, G\}$

$A = \{60,30,10,60,30,20,10\}$

item	total amount sale in last summer
A	178
B	86
C	29
D	172
E	84
F	57
G	24
H	0
I	0

item	total amount stored in a month of last summer
A	60
B	30
C	10
D	60
E	30
F	20
G	10
H	0
I	0

Total sale from last summer → Total amount stored in the summer of this month
(This is not our job, this is the warehouse job)

Problem Instance 1

Inputs

item	volume
A	3
B	2
C	6
D	2
E	2
F	3
G	6
H	4
I	6

the size of different types of item

item	weight(kg)
A	2
B	8
C	3
D	4
E	2
F	6
G	4
H	2
I	5

the weight of different types of item

Problem Instance 1

Preprocessing

1.Get d_i

Total damand by item	
item	Demand d_i
A	5
B	4
C	4
D	3
E	3
F	4
G	0

Item A is ordered most frequently, then Item D, B and E.

2.Get CO

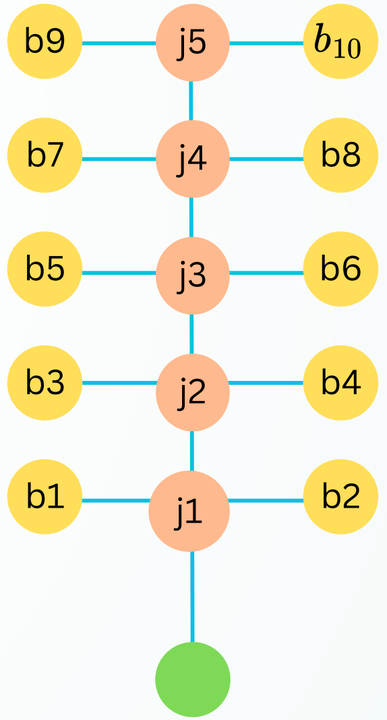
Here is the matrix form of item co-occurrence derived from customer orders:

```
    A  B  C  D  E  F  G
A [[0, 2, 2, 1, 0, 2, 0],
B  [2, 0, 1, 1, 1, 1, 0],
C  [2, 1, 0, 1, 0, 2, 0],
D  [1, 1, 1, 0, 1, 0, 0],
E  [0, 1, 0, 1, 0, 1, 0],
F  [2, 1, 2, 0, 1, 0, 0],
G  [0, 0, 0, 0, 0, 0, 0]]
```

Each cell indicates the number of customer orders where item and item appeared together.

3.Get $G = (V, E)$

Graph



4.Get k_i

The number of blocks needed to store item i.	
item	Block needed k_i
A	3
B	1
C	1
D	2
E	1
F	1
G	1

Item A appear the most frequent, then Item D.

5. Prepare I

$I_{prepared} =$ {A, A, A, B, C, D, D, E, F, G}

Problem Instance 1

Before Greedy loops

- Define weights for PPS and LCS

$$PPS(i) = w_{\text{freq}} \cdot \frac{d_i \cdot w_i}{\max\limits_{j \in I}(d_j \cdot w_j)} + w_{\text{cooc}} \cdot \frac{\sum\limits_{j \in I, j \neq i} co(i,j)}{\max\limits_{k \in I} \sum\limits_{j \in I, j \neq k} co(k,j)}$$

$$w_{\text{freq}} = 0.5 \quad w_{\text{cooc}} = 0.5$$

$$LCS(i,b) = w_{\text{depot}} \cdot \frac{\text{Dist}(\text{Depot}, b) \cdot w_i}{k_i} + w_{\text{affinity}} \cdot \sum_{j \in I_{\text{placed}}} co(i,j) \cdot \text{Dist}(b, b'_j)$$

$$w_{depot} = 0.5 \quad w_{affinity} = 0.5$$

- We set every weight to be 0.5 (for now) because we can’t run a simulation to find the best weights (yet).

Step-by-Step Running of Algorithm

$I_{prepared}$ = {A, A, A, B, C, D, D, E, F, G} B = {b1,b2,b3,b4,b5,b6,b7,b8,b9,b10}

while (items have not been fully allocated)**{**

- For each **unclassified item** in list I , **Calculate PPS**.
- **Select** the **unclassified item** type **i** from I with the **highest PPS score**.
- For each available **candidate block** ***b*** , **Calculate LCS**.
- **Select** the block with the **lowest LCS scores**.
- **Assign** this block **b** to item **i**
- **Remove** the selected block **b** from the list B.
- **Remove** the selected item **i** from the list I .

}

End while

OUTPUT： **Final Item-to-Block Placement Plan**

Item	PPS
A	0.15625
B	0.5
C	0.1875
D	0.1875
E	0.09375
F	0.375
G	0

Step-by-Step Running of Algorithm

$I_{prepared}$ = {A, A, A, B, C, D, D, E, F, G} B = {b1,b2,b3,b4,b5,b6,b7,b8,b9,b10}

while (items have not been fully allocated)**{**

- For each **unclassified item** in list I , **Calculate PPS**.
- **Select** the **unclassified item** type **i** from I with the **highest PPS score**.
- For each available **candidate block** ***b*** , **Calculate LCS**.
- **Select** the block with the **lowest LCS scores**.
- **Assign** this block **b** to item **i**
- **Remove** the selected block **b** from the list B.
- **Remove** the selected item **i** from the list I .

}

End while

OUTPUT: Final Item-to-Block Placement Plan

Item	PPS
A	0.15625
B	0.5
C	0.1875
D	0.1875
E	0.09375
F	0.375
G	0

Step-by-Step Running of Algorithm

$I_{prepared}$ = {A, A, A, B, C, D, D, E, F, G} B = {b1,b2,b3,b4,b5,b6,b7,b8,b9,b10}

while (items have not been fully allocated)**{**

- For each **unclassified item** in list I , **Calculate PPS**.
- **Select** the **unclassified item** type **i** from I with the **highest PPS score**.
- For each available **candidate block** ***b*** , **Calculate LCS**.
- **Select** the block with the **lowest LCS scores**.
- **Assign** this block **b** to item **i**
- **Remove** the selected block **b** from the list B.
- **Remove** the selected item **i** from the list I .

}

End while

OUTPUT： **Final Item-to-Block Placement Plan**

Block	LCS
b1	8
b2	8
b3	12
b4	12
b5	16
b6	16
b7	20
b8	20
b9	24
b10	24

Step-by-Step Running of Algorithm

$I_{prepared}$ = {A, A, A, B, C, D, D, E, F, G} B = {b1,b2,b3,b4,b5,b6,b7,b8,b9,b10}

while (items have not been fully allocated)**{**

- For each **unclassified item** in list I , **Calculate PPS**.
- **Select** the **unclassified item** type **i** from I with the **highest PPS score**.
- For each available **candidate block *b*, Calculate LCS**.
- **Select** the block with the **lowest LCS scores**.
- **Assign** this block **b** to item **i**
- **Remove** the selected block **b** from the list B.
- **Remove** the selected item **i** from the list I .

}

End while

OUTPUT: Final Item-to-Block Placement Plan

Block	LCS
b1	8
b2	8
b3	12
b4	12
b5	16
b6	16
b7	20
b8	20
b9	24
b10	24

Step-by-Step Running of Algorithm

$I_{prepared}$ = {A, A, A, B, C, D, D, E, F, G} B = {b1,b2,b3,b4,b5,b6,b7,b8,b9,b10}

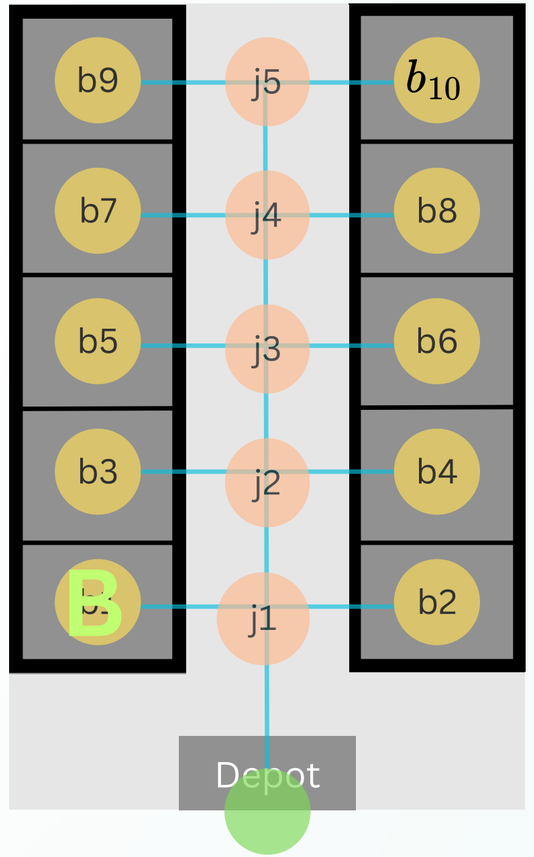
while (items have not been fully allocated){

- For each unclassified item in list I , **Calculate PPS**.
- **Select** the unclassified item type **i** from I with the **highest PPS score**.
- For each available candidate block ***b*** , **Calculate LCS**.
- **Select** the block with the **lowest LCS scores**.
- **Assign** this block **b** to item **i**
- **Remove** the selected block **b** from the list B.
- **Remove** the selected item **i** from the list I .

}

End while

OUTPUT： Final Item-to-Block Placement Plan



Step-by-Step Running of Algorithm

$I_{prepared}$ = {A, A, A, B, C, D, D, E, F, G} B = {b2,b3,b4,b5,b6,b7,b8,b9,b10}

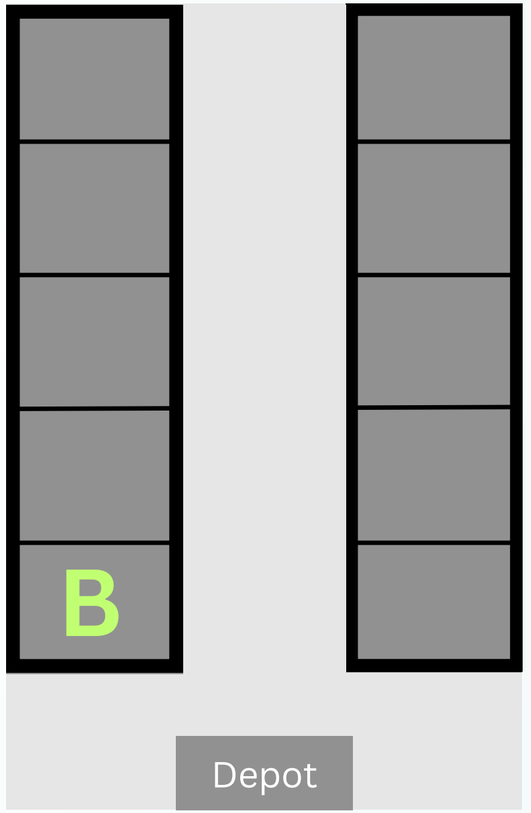
while (items have not been fully allocated)**{**

- For each **unclassified item** in list I , **Calculate PPS**.
- **Select** the **unclassified item** type **i** from I with the **highest PPS score**.
- For each available **candidate block** ***b*** , **Calculate LCS**.
- **Select** the block with the **lowest LCS scores**.
- **Assign** this block **b** to item **i**
- **Remove** the selected block **b** from the list B.
- **Remove** the selected item **i** from the list I .

}

End while

OUTPUT： **Final Item-to-Block Placement Plan**



Step-by-Step Running of Algorithm

$I_{prepared}$ = {A, A, A, C, D, D, E, F, G} B = {b2,b3,b4,b5,b6,b7,b8,b9,b10}

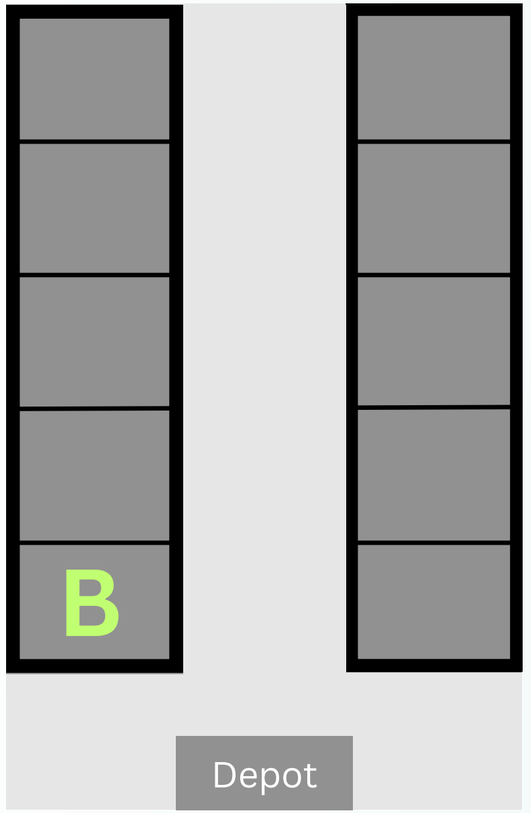
while (items have not been fully allocated)**{**

- For each unclassified item in list I , **Calculate PPS**.
- **Select** the unclassified item type **i** from I with the **highest PPS score**.
- For each available candidate block ***b*** , **Calculate LCS**.
- **Select** the block with the **lowest LCS scores**.
- **Assign** this block **b** to item **i**
- **Remove** the selected block **b** from the list B.
- **Remove** the selected item **i** from the list I .

}

End while

OUTPUT： **Final Item-to-Block Placement Plan**



Step-by-Step Running of Algorithm

$I_{prepared}$ = {A, A, A, C, D, D, E, F, G} B = {b2,b3,b4,b5,b6,b7,b8,b9,b10}

while (items have not been fully allocated)**{**

- For each unclassified item in list I , **Calculate PPS**.
- **Select** the unclassified item type **i** from I with the **highest PPS score**.
- For each available candidate block ***b*** , **Calculate LCS**.
- **Select** the block with the **lowest LCS scores**.
- **Assign** this block **b** to item **i**
- **Remove** the selected block **b** from the list B.
- **Remove** the selected item **i** from the list I .

}

End while

OUTPUT： Final Item-to-Block Placement Plan

Item	PPS
A	0.2991
B	
C	0.2589
D	0.2589
E	0.1652
F	0.4464
G	0

Step-by-Step Running of Algorithm

$I_{prepared}$ = {A, A, A, C, D, D, E, F, G} B = {b2,b3,b4,b5,b6,b7,b8,b9,b10}

while (items have not been fully allocated)**{**

- For each **unclassified item** in list I , **Calculate PPS**.
- **Select** the **unclassified item** type **i** from I with the **highest PPS score**.
- For each available **candidate block** ***b*** , **Calculate LCS**.
- **Select** the block with the **lowest LCS scores**.
- **Assign** this block **b** to item **i**
- **Remove** the selected block **b** from the list B.
- **Remove** the selected item **i** from the list I .

}

End while

OUTPUT： **Final Item-to-Block Placement Plan**

Item	PPS
A	0.2991
B	
C	0.2589
D	0.2589
E	0.1652
F	0.4464
G	0

Step-by-Step Running of Algorithm

$I_{prepared}$ = {A, A, A, C, D, D, E, F, G} B = {b2,b3,b4,b5,b6,b7,b8,b9,b10}

while (items have not been fully allocated)**{**

- For each **unclassified item** in list I , **Calculate PPS**.
- **Select** the **unclassified item** type **i** from I with the **highest PPS score**.
- For each available **candidate block** ***b*** , **Calculate LCS**.
- **Select** the block with the **lowest LCS scores**.
- **Assign** this block **b** to item **i**
- **Remove** the selected block **b** from the list B.
- **Remove** the selected item **i** from the list I .

}

End while

OUTPUT： **Final Item-to-Block Placement Plan**

Block	LCS
b1	
b2	7
b3	10.5
b4	10.5
b5	14
b6	14
b7	17.5
b8	17.5
b9	21
b10	21

Step-by-Step Running of Algorithm

$I_{prepared}$ = {A, A, A, C, D, D, E, F, G} B = {b2,b3,b4,b5,b6,b7,b8,b9,b10}

while (items have not been fully allocated)**{**

- For each **unclassified item** in list I , **Calculate PPS**.
- **Select** the **unclassified item** type **i** from I with the **highest PPS score**.
- For each available **candidate block *b*，Calculate LCS**.
- **Select** the block with the **lowest LCS scores**.
- **Assign** this block **b** to item **i**
- **Remove** the selected block **b** from the list B.
- **Remove** the selected item **i** from the list I .

}

End while

OUTPUT： Final Item-to-Block Placement Plan

Block	LCS
b1	
b2	7
b3	10.5
b4	10.5
b5	14
b6	14
b7	17.5
b8	17.5
b9	21
b10	21

Step-by-Step Running of Algorithm

$I_{prepared}$ = {A, A, A, C, D, D, E, F, G} B = {b2,b3,b4,b5,b6,b7,b8,b9,b10}

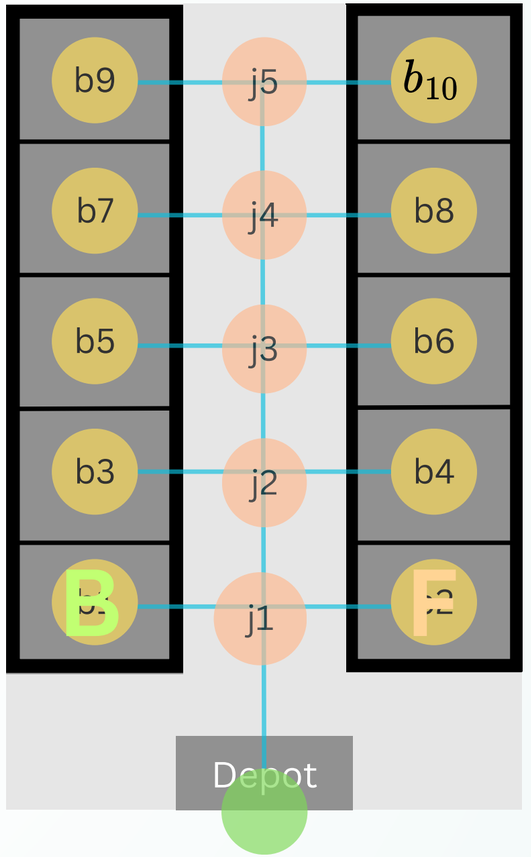
while (items have not been fully allocated)**{**

- For each unclassified item in list I , **Calculate PPS**.
- **Select** the unclassified item type **i** from I with the **highest PPS score**.
- For each available candidate block ***b*** , **Calculate LCS**.
- **Select** the block with the **lowest LCS scores**.
- **Assign** this block **b** to item **i**
- **Remove** the selected block **b** from the list B.
- **Remove** the selected item **i** from the list I .

}

End while

OUTPUT： **Final Item-to-Block Placement Plan**



Step-by-Step Running of Algorithm

$I_{prepared}$ = {A, A, A, C, D, D, E, F, G} B = {b3,b4,b5,b6,b7,b8,b9,b10}

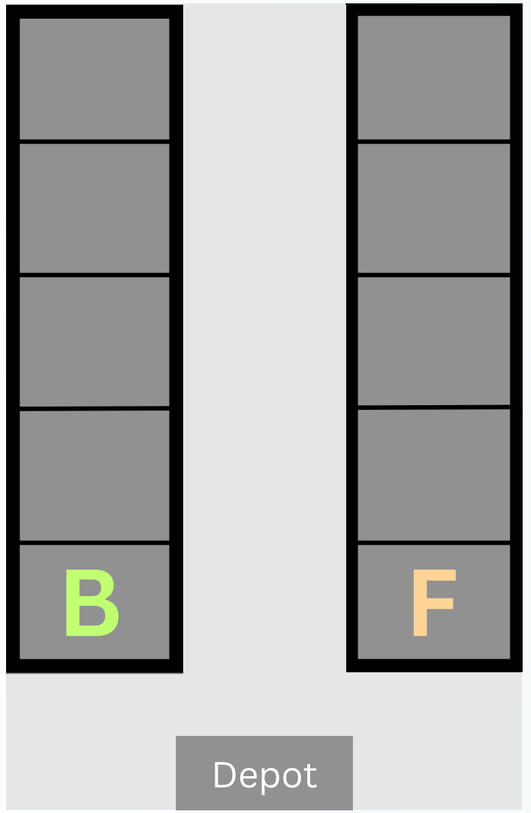
while (items have not been fully allocated)**{**

- For each unclassified item in list I , **Calculate PPS**.
- **Select** the unclassified item type **i** from I with the **highest PPS score**.
- For each available candidate block ***b*** , **Calculate LCS**.
- **Select** the block with the **lowest LCS scores**.
- **Assign** this block **b** to item **i**
- **Remove** the selected block **b** from the list B.
- **Remove** the selected item **i** from the list I .

}

End while

OUTPUT： **Final Item-to-Block Placement Plan**



Step-by-Step Running of Algorithm

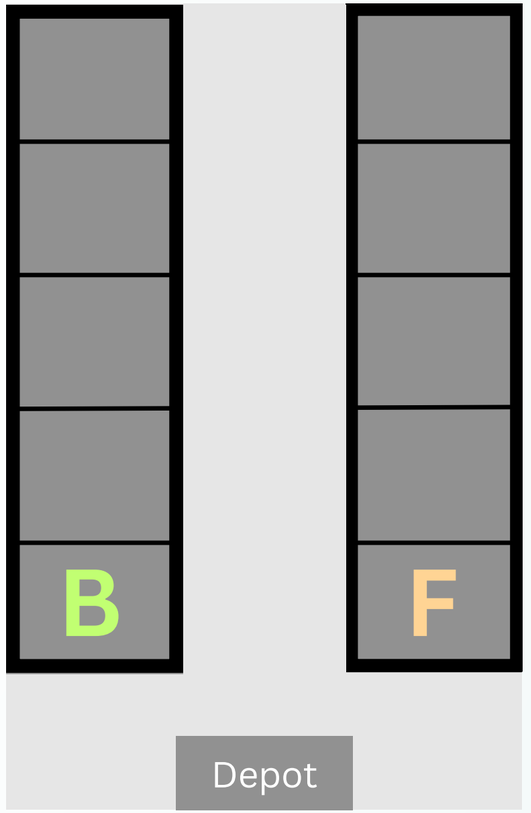
$I_{prepared}$ = {A, A, A, C, D, D, E, G} B = {b3,b4,b5,b6,b7,b8,b9,b10}

while (items have not been fully allocated)**{**

- For each **unclassified item** in list I , **Calculate PPS**.
- **Select** the **unclassified item** type **i** from I with the **highest PPS score**.
- For each available **candidate block** ***b*** , **Calculate LCS**.
- **Select** the block with the **lowest LCS scores**.
- **Assign** this block **b** to item **i**
- **Remove** the selected block **b** from the list B.
- **Remove** the selected item **i** from the list I .

}

End while



OUTPUT： Final Item-to-Block Placement Plan

Step-by-Step Running of Algorithm

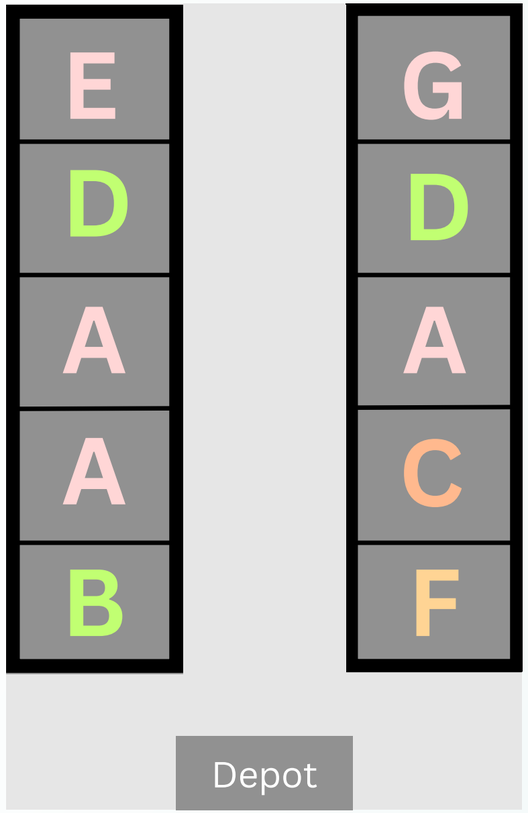
$I_{prepared} = \{\}$ $B = \{\}$

while (items have not been fully allocated)**{**

- For each **unclassified item** in list I , **Calculate PPS**.
- **Select** the **unclassified item** type **i** from I with the **highest PPS score**.
- For each available **candidate block** ***b*** , **Calculate LCS**.
- **Select** the block with the **lowest LCS scores**.
- **Assign** this block **b** to item **i**
- **Remove** the selected block **b** from the list B.
- **Remove** the selected item **i** from the list I .

}

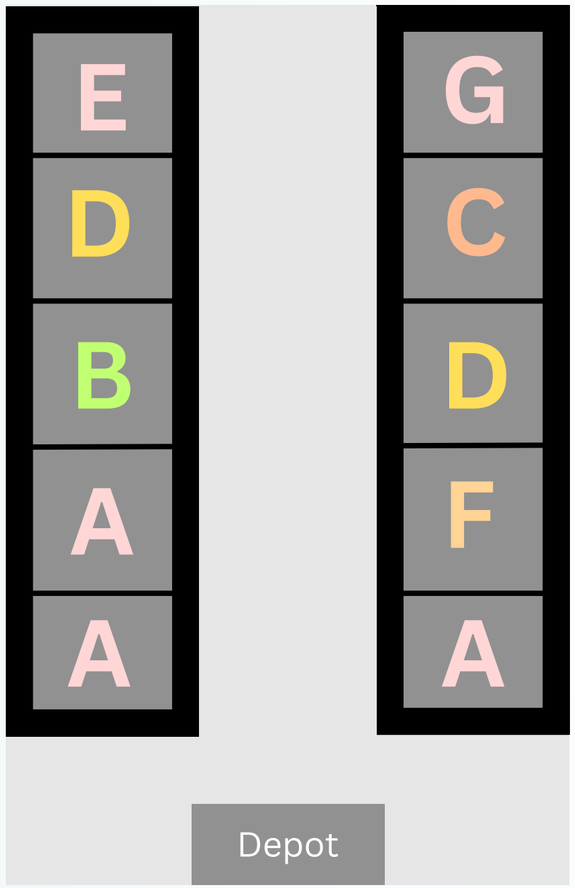
End while



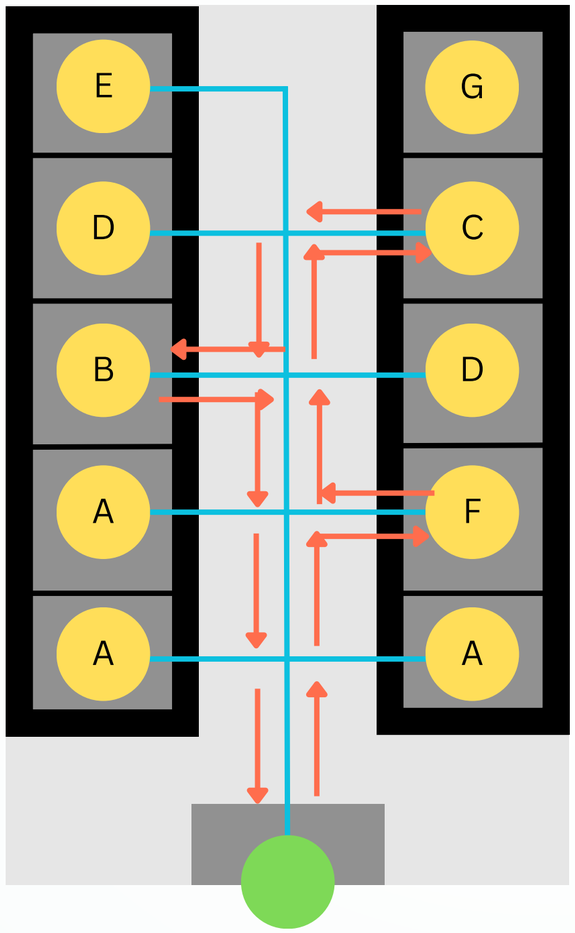
OUTPUT： Final Item-to-Block Placement Plan

Problem Instance 1

Comparison with our M2 plan (Without dynamic PPS and weight)



Warehouse plan without much thought



Example walking path for P3 (B,C,F)

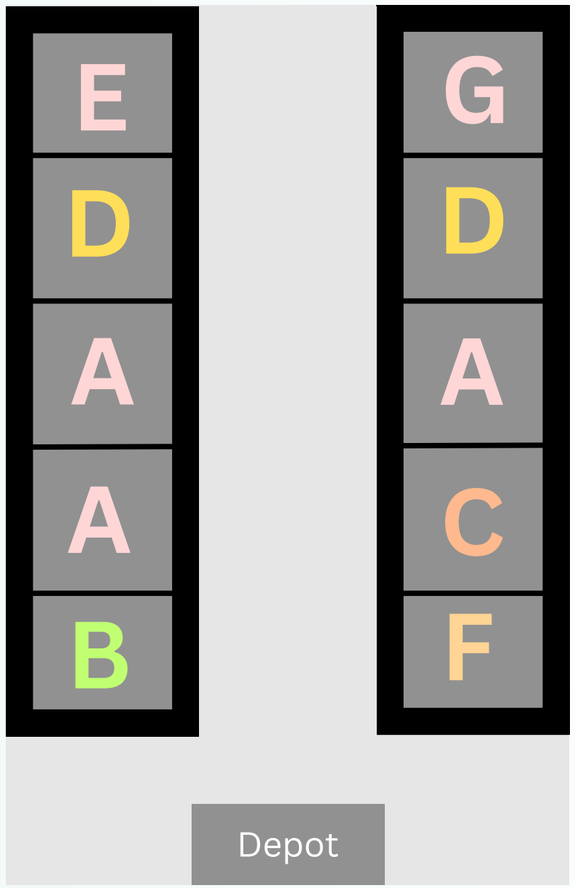
Customer	Effort	distance
P1	274	10
P2	147	14
P3	336	14
P4	606	14
P5	116	8
P6	497	14
P7	542	14
P8	116	14
P9	161	14
P10	206	14
Total	3001	130

Handling Effort = 3001 kg*Meters

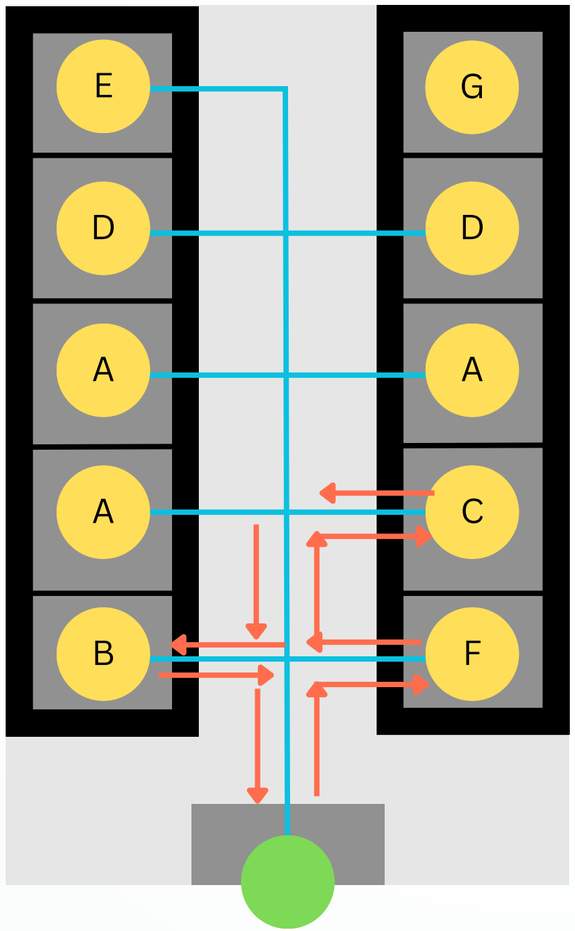
Total Walking Distance = 130 Meters

Problem Instance 1

Our warehouse plan



Warehouse plan from the algorithm



Example walking path for P3 (B,C,F)

Customer	Effort	distance
P1	180	8
P2	193	12
P3	180	10
P4	706	14
P5	130	10
P6	541	12
P7	526	14
P8	98	14
P9	141	12
P10	158	14
Total	2853	120

Handling Effort = 2853 kg*Meters

Total Walking Distance = 120 Meters

Problem Instance 1 (Season 2)

Inputs

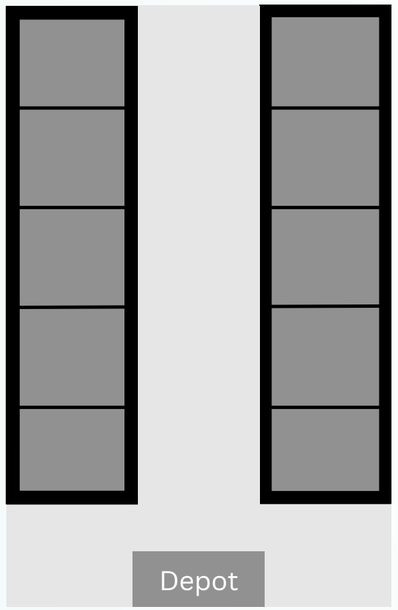
Items(long-term) : $I =$ {A, B, C, D, H,I}

$A =$ {40,30,10,60,45,10}

Orders $\mathcal{O}$:

Custom	ItemID	amount
P1	A	23
P1	B	12
P1	H	14
P2	B	18
P2	D	11
P3	I	3
P3	A	5
P4	C	10
P4	D	8
P4	H	16
P5	H	14
P5	D	25
P6	A	10
P7	D	13
P7	I	7

total oders for a month last autumn



Warehouse Plan

C = 60 Units

item	total amount sale in last autumn
A	117
B	85
C	26
D	170
E	0
F	0
G	0
H	130
I	29

item	total amount stored last autumn
A	40
B	30
C	10
D	60
E	0
F	0
G	0
H	45
I	10

Total sale from last autumn→ Total amount stored in the summer of this month
(This is not our job, this is the warehouse job)

Problem Instance 1 (Season 2)

Inputs

item	volume
A	3
B	2
C	6
D	2
E	2
F	3
G	6
H	4
I	6

the size of different types of item

item	weight(kg)
A	2
B	8
C	3
D	4
E	2
F	6
G	4
H	2
I	5

the weight of different types of item

Problem Instance 1 (Season 2)

Preprocessed Inputs

1. Get k_i

The number of blocks needed to store item	
item	Block needed k_i
A	2
B	1
C	1
D	2
H	3
I	1

Item A is the most popular, then Item D.

2. Get d_i

Total damand by item	
item	Demand d_i
A	3
B	2
C	1
D	4
H	3
I	2

Item D is ordered most frequently, then Item A, B and I.

3. Get CO

Here is the matrix form of item co-occurrence derived from customer orders:

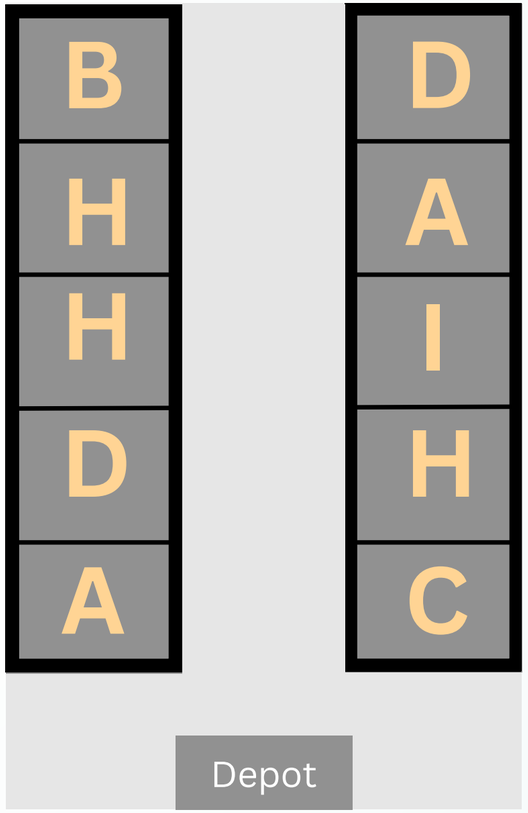
	A	B	C	D	H	I
A	0	1	0	0	1	1
B	1	0	0	1	1	0
C	0	0	0	1	1	0
D	0	1	1	0	2	1
H	1	1	1	2	0	0
I	1	0	0	1	0	0

Each cell indicates the number of customer orders where item and item appeared together.

Problem Instance 1 (Season 2)

unoptimized plan

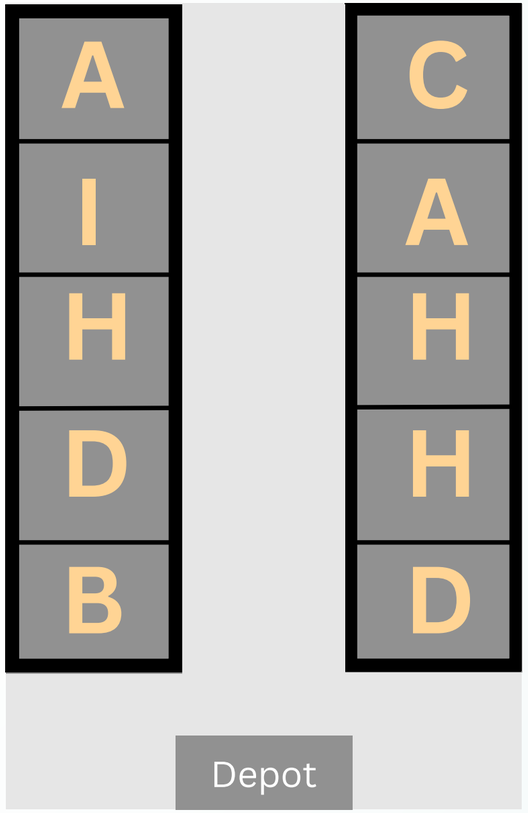
Evaluation(Randomly,S2)		
Customer	Effort	distance
P1	998	16
P2	1010	14
P3	122	12
P4	428	16
P5	1055	15
P6	110	10
P7	466	14
Total	4189	97



Problem Instance 1 (Season 2)

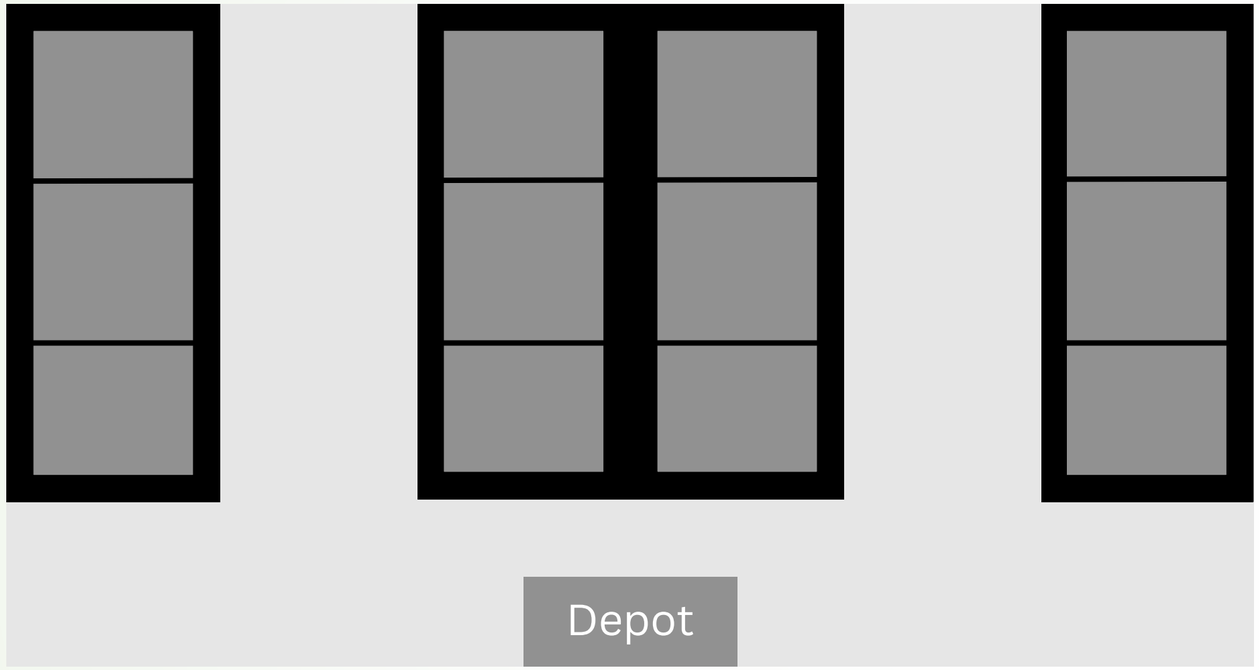
Our warehouse plan

Evaluation(Optimized,S2)		
Customer	Effort	distance
P1	800	18
P2	382	6
P3	149	14
P4	390	18
P5	424	12
P6	132	12
P7	343	12
Total	2620	92



Handling Effort reduced by 37.5%

Problem Instance 2(Season 1)



Warehouse plan

6 Item types :

A, B, C, D, E, F

Block Capacity C: 60 Units

Warehouse Capacity : 12 Blocks

Orders

Customer ID	ItemID	Amount
P1	A	12
P1	B	8
P1	E	13
P2	B	7
P2	C	10
P3	C	8
P3	D	9
P3	F	10
P4	B	10
P4	D	6
P4	E	30
P5	A	8
P5	E	10
P6	E	12
P7	C	11
P7	E	14

Problem Instance 2(Season 1)

Inputs

Orders $\mathcal{O}$:

Customer ID	ItemID	Amount
P1	A	12
P1	B	8
P1	E	13
P2	B	7
P2	C	10
P3	C	8
P3	D	9
P3	F	10
P4	B	10
P4	D	6
P4	E	30
P5	A	8
P5	E	10
P6	E	12
P7	C	11
P7	E	14

$$\mathcal{O} = \{O_1, O_2, \ldots, O_n\}, each\ O_k \subseteq I$$

Items(long-term) : $I = \{i_1, i_2, \ldots, i_m\}$

Item	total amount sales
A	87
B	88
C	85
D	42
E	235
F	27

Item	total amount stored this month
A	30
B	30
C	30
D	15
E	80
F	10

Problem Instance 2(Season 1)

Inputs

Item	volume
A	4
B	6
C	2
D	4
E	3
F	6

The volume of each item.

Item	weight(kg)
A	2
B	5
C	1
D	6
E	3
F	6

The weight of each item.

Problem Instance 2(Season 1)

Preprocessed Inputs

1.Get d_i

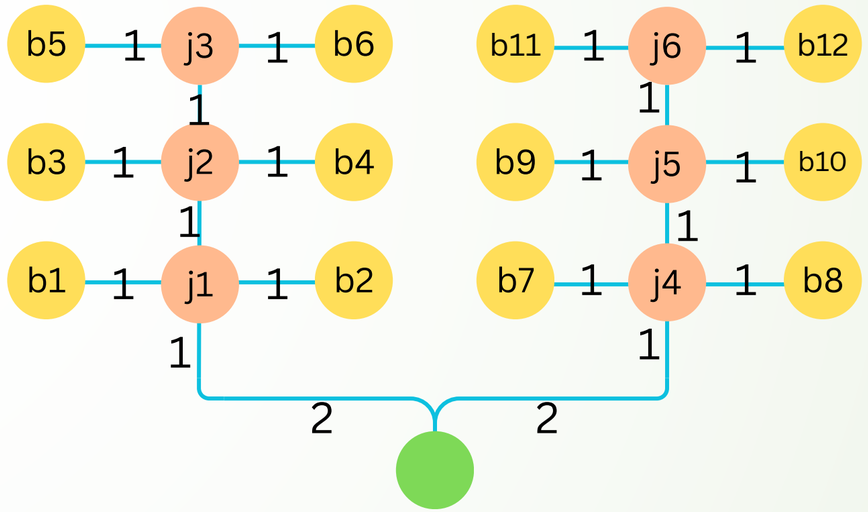
Total demand by item	
item	d_i
A	2
B	3
C	2
D	2
E	5
F	1

2.Get CO

	A	B	C	D	E	F
A	0	1	0	0	2	0
B	1	0	1	1	2	0
C	0	1	0	1	0	1
D	0	1	1	0	1	1
E	2	2	0	1	0	0
F	0	0	1	1	0	0

3.Get the $G = (V, E)$

Graph



4.Get k_i

The number of blocks needed to store item i.	
item	Block needed k_i
A	2
B	3
C	1
D	1
E	4
F	1

5. Prepare I

$$I_{prepared} = \{A, A, B, B, B, C, D, E, E, E, E, F\}$$

Item E appear the most frequent, then Item B.

Step-by-Step Running of Algorithm

$I_{prepared}$ = {A, A, B, B, B, C, D, E, E, E, E, F} B ={b1,b2,b3,b4,b5,b6,b7,b8,b9, b10,b11,b12}

while (items have not been fully allocated)**{**

- For each **unclassified item** in list I , **Calculate PPS**.
- **Select** the **unclassified item** type **i** from I with the **highest PPS score**.
- For each available **candidate block** ***b*** , **Calculate LCS**.
- **Select** the block with the **lowest LCS scores**.
- **Assign** this block **b** to item **i**
- **Remove** the selected block **b** from the list B.
- **Remove** the selected item **i** from the list I .

}

End while

OUTPUT： **Final Item-to-Block Placement Plan**

Item	PPS
A	0.1333
B	0.5
C	0.1
D	0.4
E	0.5
F	0.2

Step-by-Step Running of Algorithm

$I_{prepared}$ = {A, A, B, B, B, C, D, E, E, E, E, F} B ={b1,b2,b3,b4,b5,b6,b7,b8,b9, b10,b11,b12}

while (items have not been fully allocated)**{**

- For each **unclassified item** in list I , **Calculate PPS**.
- **Select** the **unclassified item** type **i** from I with the **highest PPS score**.
- For each available **candidate block** ***b*** , **Calculate LCS**.
- **Select** the block with the **lowest LCS scores**.
- **Assign** this block **b** to item **i**
- **Remove** the selected block **b** from the list B.
- **Remove** the selected item **i** from the list I .

}

End while

OUTPUT： **Final Item-to-Block Placement Plan**

Item	PPS
A	0.1333
B	0.5
C	0.1
D	0.4
E	0.5
F	0.2

Step-by-Step Running of Algorithm

$I_{prepared}$ = {A, A, B, B, B, C, D, E, E, E, E, F} B ={b1,b2,b3,b4,b5,b6,b7,b8,b9, b10,b11,b12}

while (items have not been fully allocated)**{**

- For each **unclassified item** in list I , **Calculate PPS**.
- **Select** the **unclassified item** type **i** from I with the **highest PPS score**.
- For each available **candidate block** ***b*** , **Calculate LCS**.
- **Select** the block with the **lowest LCS scores**.
- **Assign** this block **b** to item **i**
- **Remove** the selected block **b** from the list B.
- **Remove** the selected item **i** from the list I .

}

End while

OUTPUT： **Final Item-to-Block Placement Plan**

Block	LCS
b1	3.3333
b2	3.3333
b3	4.1667
b4	4.1667
b5	5
b6	5
b7	3.3333
b8	3.3333
b9	4.1667
b10	4.1667
b11	5
b12	5

Step-by-Step Running of Algorithm

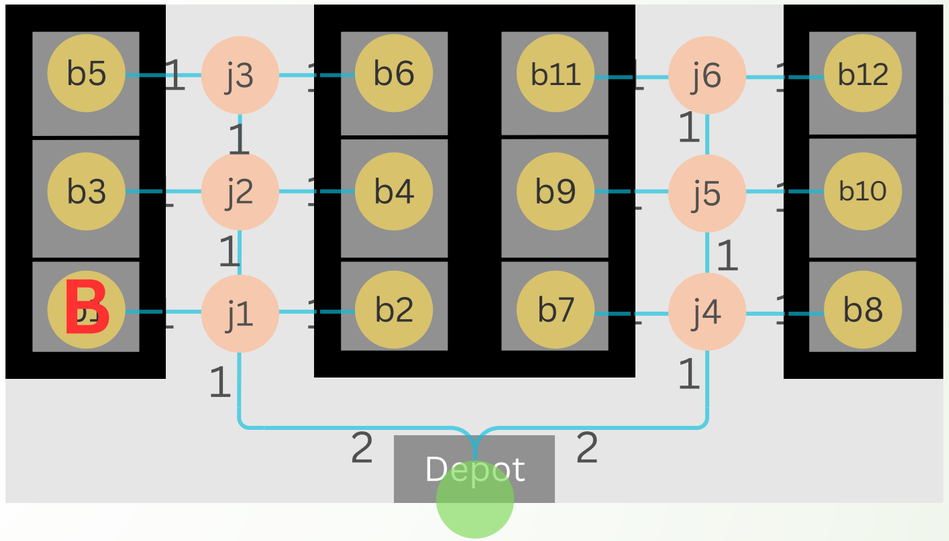
$I_{prepared}$ = {A, A, B, B, B, C, D, E, E, E, E, F} B ={b1,b2,b3,b4,b5,b6,b7,b8,b9, b10,b11,b12}

while (items have not been fully allocated)**{**

- For each **unclassified item** in list I , **Calculate PPS**.
- **Select** the **unclassified item** type **i** from I with the **highest PPS score**.
- For each available **candidate block** ***b*** , **Calculate LCS**.
- **Select** the block with the **lowest LCS scores**.
- **Assign** this block **b** to item **i**
- **Remove** the selected block **b** from the list B.
- **Remove** the selected item **i** from the list I .

}

End while



OUTPUT： **Final Item-to-Block Placement Plan**

Step-by-Step Running of Algorithm

$I_{prepared}$ = {A, A, B, B, C, D, E, E, E, E, F}

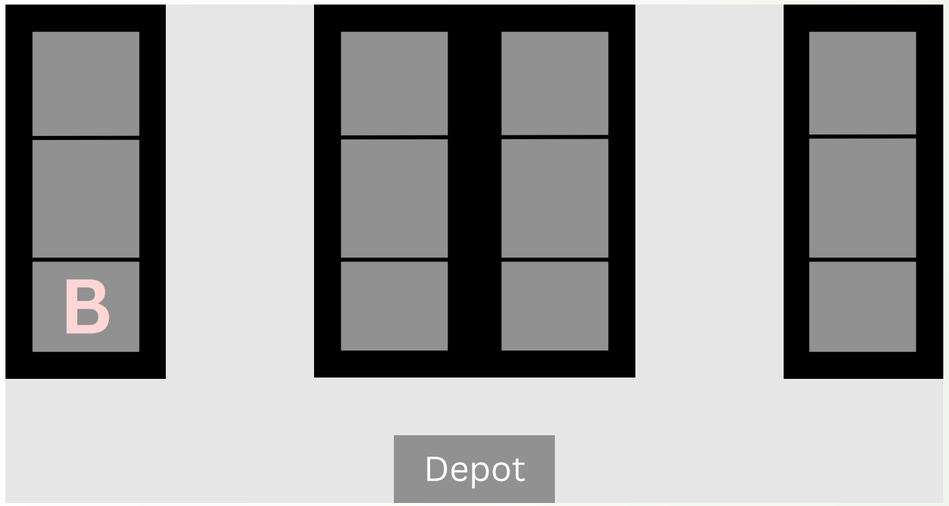
B ={b2,b3,b4,b5,b6,b7,b8,b9, b10,b11,b12}

while (items have not been fully allocated)**{**

- For each **unclassified item** in list I , **Calculate PPS**.
- **Select** the **unclassified item** type **i** from I with the **highest PPS score**.
- For each available **candidate block** ***b*** , **Calculate LCS**.
- **Select** the block with the **lowest LCS scores**.
- **Assign** this block **b** to item **i**
- **Remove** the selected block **b** from the list B.
- **Remove** the selected item **i** from the list I .

}

End while



OUTPUT： **Final Item-to-Block Placement Plan**

Step-by-Step Running of Algorithm

$I_{prepared}$ = {A, A, C, D, E, E, E, F}

B ={b5,b6,b7,b8,b9,b10,b11,b12}

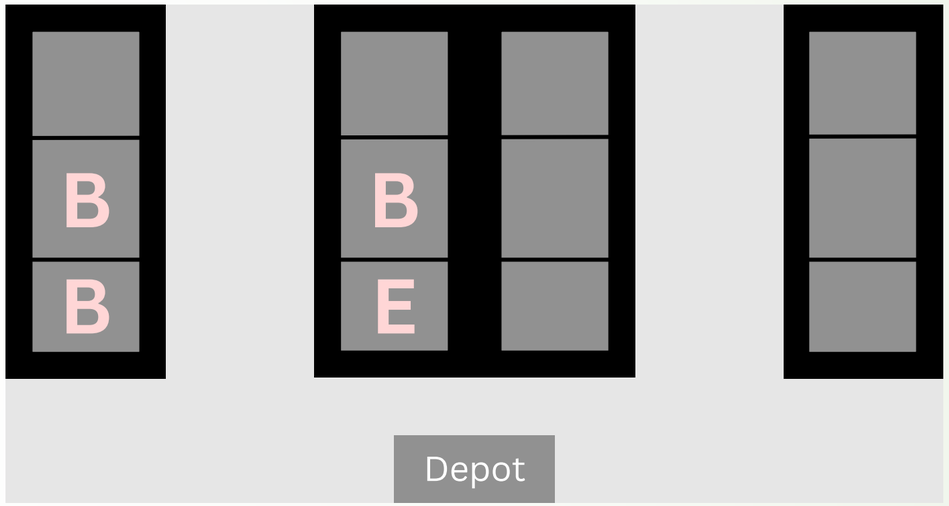
while (items have not been fully allocated)**{**

- For each **unclassified item** in list I , **Calculate PPS**.
- **Select** the **unclassified item** type **i** from I with the **highest PPS score**.
- For each available **candidate block** ***b*** , **Calculate LCS**.
- **Select** the block with the **lowest LCS scores**.
- **Assign** this block **b** to item **i**
- **Remove** the selected block **b** from the list B.
- **Remove** the selected item **i** from the list I .

}

End while

OUTPUT： **Final Item-to-Block Placement Plan**



Skip to loop 5

Step-by-Step Running of Algorithm

$I_{prepared}$ = {A, A, C, D, E, E, E, F}

B ={b5,b6,b7,b8,b9,b10,b11,b12}

while (items have not been fully allocated)**{**

- For each **unclassified item** in list I , **Calculate PPS**.
- **Select** the **unclassified item** type **i** from I with the **highest PPS score**.
- For each available **candidate block** ***b*** , **Calculate LCS**.
- **Select** the block with the **lowest LCS scores**.
- **Assign** this block **b** to item **i**
- **Remove** the selected block **b** from the list B.
- **Remove** the selected item **i** from the list I .

}

End while

OUTPUT： **Final Item-to-Block Placement Plan**

Item	PPS
A	0.3833
B	
C	0.2667
D	0.5667
E	0.6667
F	0.2

Step-by-Step Running of Algorithm

$I_{prepared}$ = {A, A, C, D, E, E, E, F}

B ={b5,b6,b7,b8,b9,b10,b11,b12}

while (items have not been fully allocated){

- For each **unclassified item** in list I , **Calculate PPS**.
- **Select** the **unclassified item** type **i** from I with the **highest PPS score**.
- For each available **candidate block** ***b*** , **Calculate LCS**.
- **Select** the block with the **lowest LCS scores**.
- **Assign** this block **b** to item **i**
- **Remove** the selected block **b** from the list B.
- **Remove** the selected item **i** from the list I .

}

End while

OUTPUT： Final Item-to-Block Placement Plan

Item	PPS
A	0.3833
B	
C	0.2667
D	0.5667
E	0.6667
F	0.2

Step-by-Step Running of Algorithm

$I_{prepared}$ = {A, A, C, D, E, E, E, F}

B = {b5,b6,b7,b8,b9,b10,b11,b12}

while (items have not been fully allocated)**{**

- For each **unclassified item** in list I , **Calculate PPS**.
- **Select** the **unclassified item** type **i** from I with the **highest PPS score**.
- For each available **candidate block** ***b*** , **Calculate LCS**.
- **Select** the block with the **lowest LCS scores**.
- **Assign** this block **b** to item **i**
- **Remove** the selected block **b** from the list B.
- **Remove** the selected item **i** from the list I .

}

End while

OUTPUT： **Final Item-to-Block Placement Plan**

Block	LCS
b5	12.25
b6	12.25
b7	27.5
b8	27.5
b9	30.875
b10	30.874
b11	34.25
b12	34.25

Step-by-Step Running of Algorithm

$I_{prepared}$ = {A, A, C, D, E, E, F}

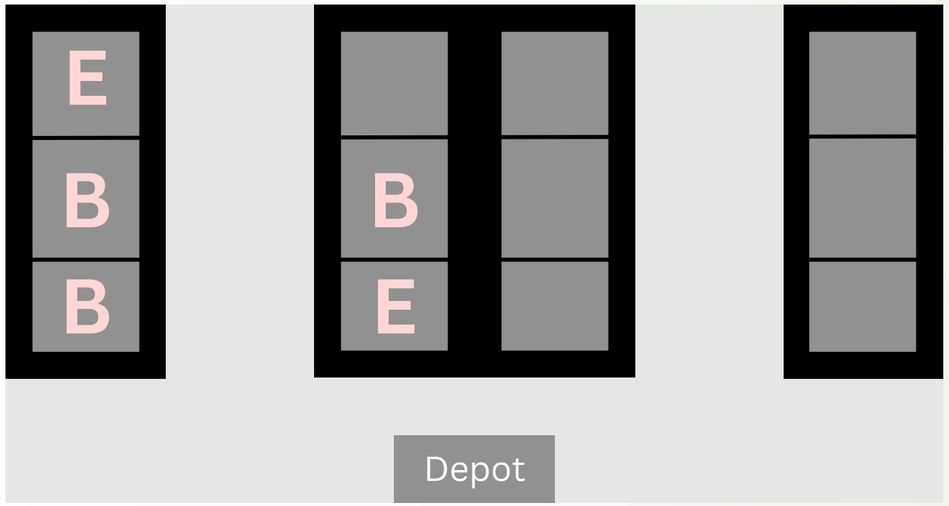
B ={b6,b7,b8,b9,b10,b11,b12}

while (items have not been fully allocated)**{**

- For each **unclassified item** in list I , **Calculate PPS**.
- **Select** the **unclassified item** type **i** from I with the **highest PPS score**.
- For each available **candidate block** ***b*** , **Calculate LCS**.
- **Select** the block with the **lowest LCS scores**.
- **Assign** this block **b** to item **i**
- **Remove** the selected block **b** from the list B.
- **Remove** the selected item **i** from the list I .

}

End while



OUTPUT： **Final Item-to-Block Placement Plan**

Step-by-Step Running of Algorithm

$I_{prepared} = \{\}$ $B = \{\}$

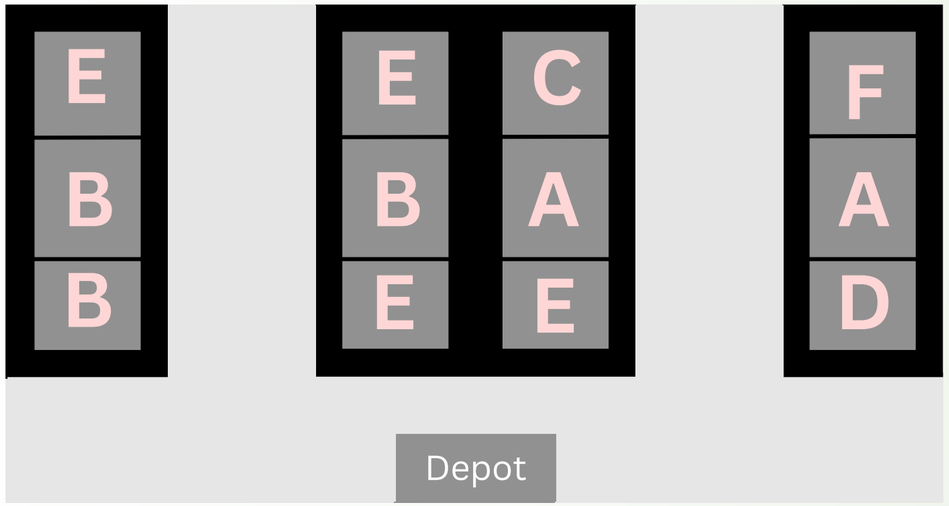
while (items have not been fully allocated)**{**

- For each **unclassified item** in list I , **Calculate PPS**.
- **Select** the **unclassified item** type **i** from I with the **highest PPS score**.
- For each available **candidate block** ***b*** , **Calculate LCS**.
- **Select** the block with the **lowest LCS scores**.
- **Assign** this block **b** to item **i**
- **Remove** the selected block **b** from the list B.
- **Remove** the selected item **i** from the list I .

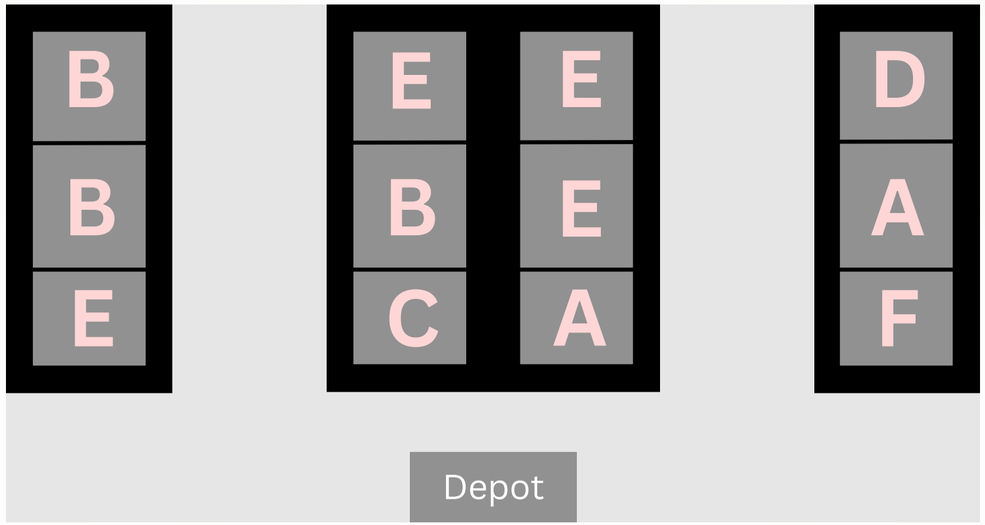
}

End while

OUTPUT： Final Item-to-Block Placement Plan



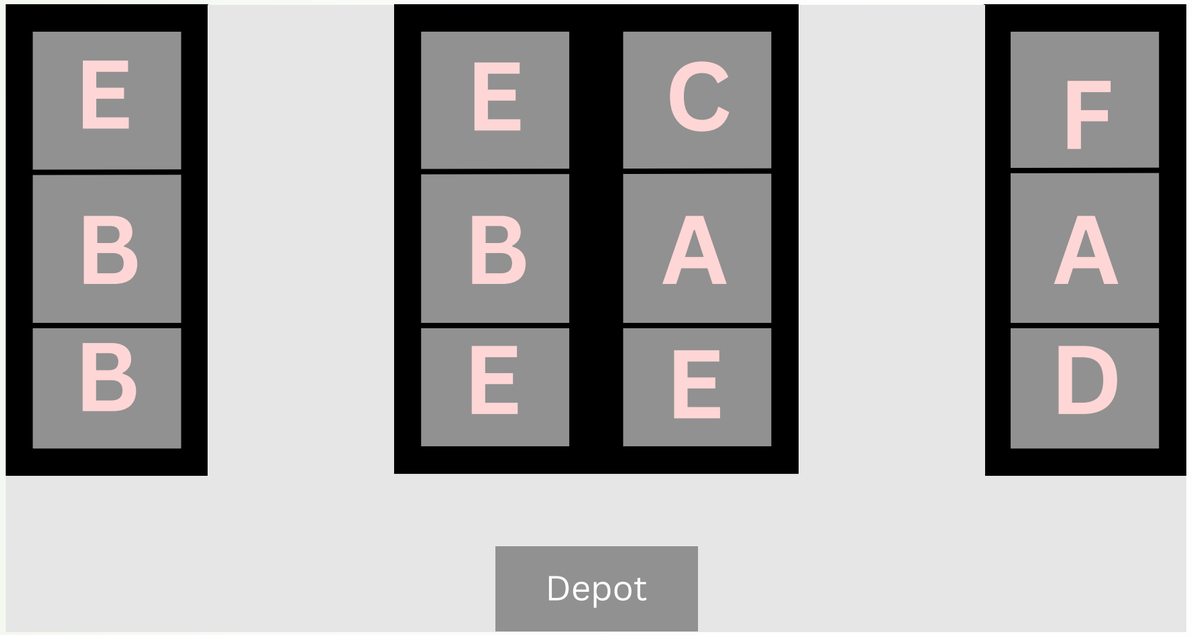
Problem Instance 2(Season 1)



Random placement

- **Total Walking Distance：168 Meters**
- **handling Effort：3088**

Problem Instance 2(Season 1)



optimized

customer	distance
p1	20
p2	24
p3	16
p4	28
p5	14
p6	20
p7	24
total	146

- **Total Walking Distance：146 meters**
- **handling Effort：2722**

Problem Instance 2(Season 2)

Inputs

Orders $\mathcal{O}$:

Customer ID	ItemID	Amount
P1	A	15
P1	E	5
P1	G	20
P2	A	10
P2	E	10
P3	B	10
P3	G	30
P3	H	6
P4	A	35
P4	B	5
P4	G	20
P5	B	5
P5	E	5
P6	G	20
P7	F	6
P7	H	6

$$\mathcal{O} = \{O_1, O_2, \ldots, O_n\}, each\ O_k \subseteq I$$

Items(long-term) : $I = \{i_1, i_2, \ldots, i_m\}$

Item	total amount sales
A	180
B	60
E	120
F	18
G	270
H	36

Item	total amount stored this month	k_i
A	60	4
B	20	2
E	20	1
F	6	1
G	90	3
H	12	1

Problem Instance 2(Season 2)

Inputs

Item	volumn
A	4
B	6
C	2
D	4
E	3
F	6
G	2
H	5

The volumn of each item.

Item	weight(kg)
A	2
B	5
C	1
D	6
E	3
F	6
G	4
H	5

The weight of each item.

Problem Instance 2(Season 2)

Preprocessed Inputs

1.Get d_i

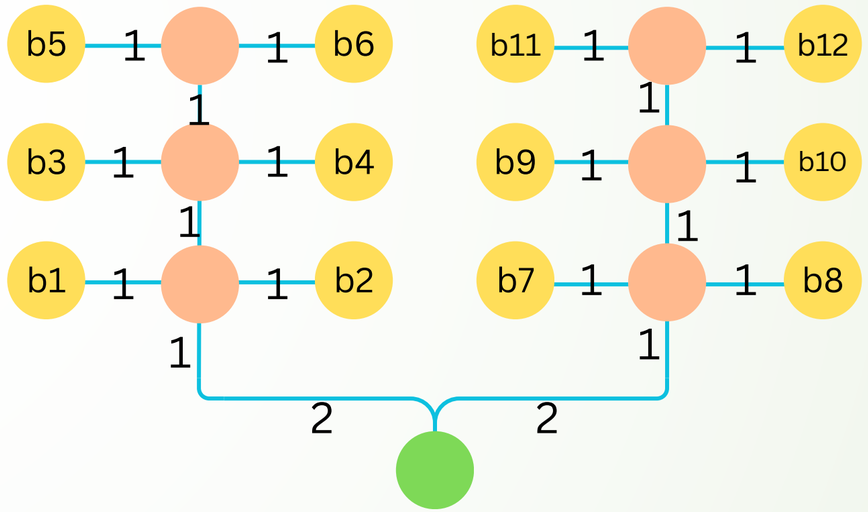
item	demand d_i
A	3
B	3
C	0
D	0
E	3
F	1
G	4
H	2

2.Get CO

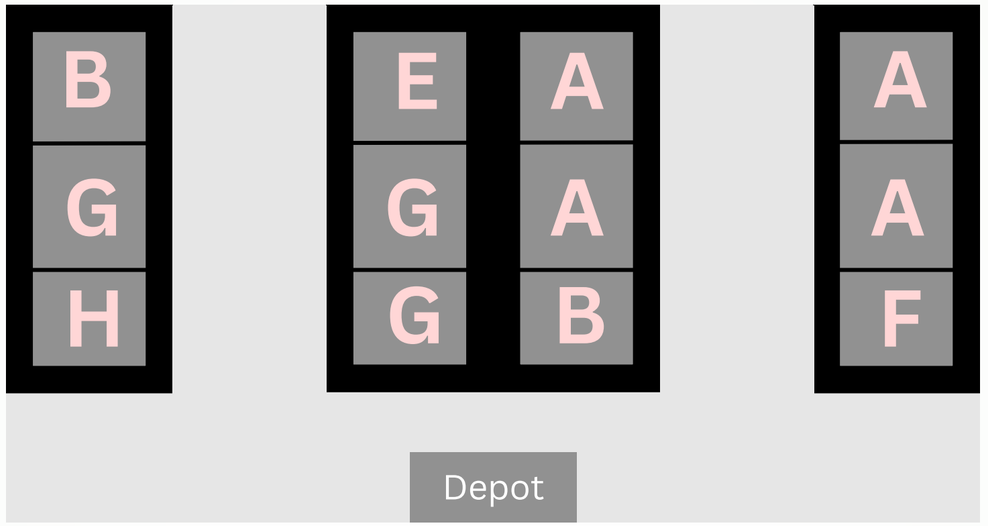
	A	B	E	F	G	H
A	0	1	2	0	2	0
B	1	0	1	0	2	1
E	2	1	0	0	1	0
F	0	0	0	0	0	1
G	2	2	1	0	0	1
H	0	1	0	1	1	0

3.Get the $G = (V, E)$

Graph



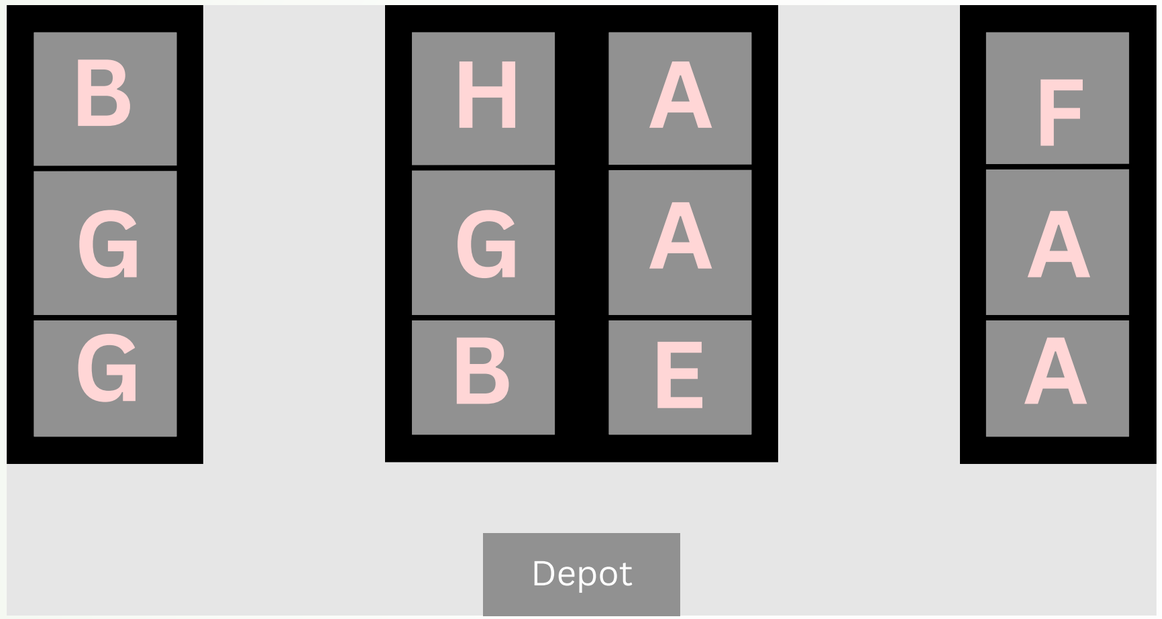
Problem Instance 2(Season 2)



Random placement

- **Total Walking Distance：164 meters**
- **handling Effort：3772**

Problem Instance 2(Season 2)



Optimized

customer	distance
p1	18
p2	12
p3	16
p4	32
p5	20
p6	10
p7	24
total	132

manually calculated total distance=132

- **Total Walking Distance：132 meters**
- **handling Effort：3351**

Looking Back

Summary of Achievements

Facing the real world issue that warehouse placement is somehow wasting a lot of extra effort when fetching items, we aimed to optimize warehouse item placement to reduce walking distance and handling effort. Through iterative algorithm design and evaluation, we developed a placement strategy based on PPS and LCS score system, dynamic inventory tracking, and graph-based distance calculation. The final solution significantly reduced total travel distance and better utilized block capacity.

Looking Back

interesting properties

- **Dynamic Greedy Algorithm:**

Quantifies item's individual popularity, weight and its relationships and **blocks' placing cost** every single loop. And always choose **the most important item** and put it in **the most convenient block dynamically**.

- **Considering Item Size and Weight:**

To make the problem more realistic and make our algorithm able to tackle with real world issue, we **remove our former assumption** that every item have the same size and weight and take them into consideration.

- **Graph-Based Warehouse Representation**:

highly flexible

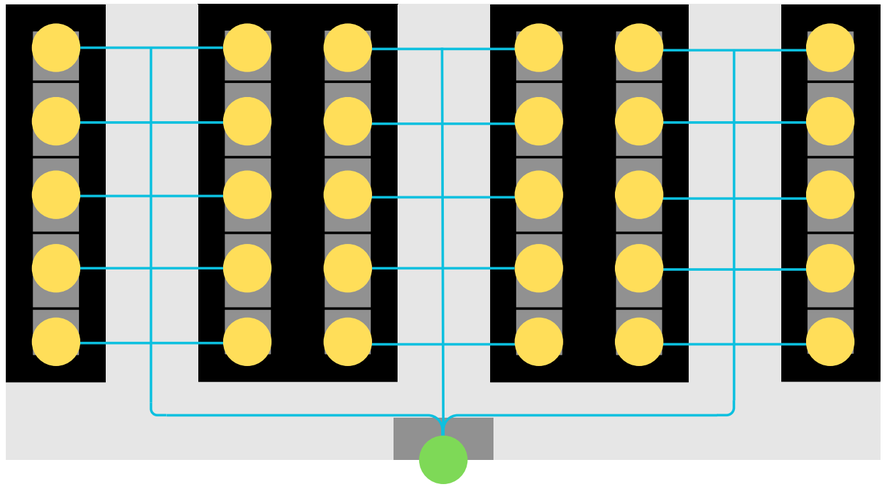
→allows the model to be **adapted to various and complex warehouse layouts**.

Looking Back

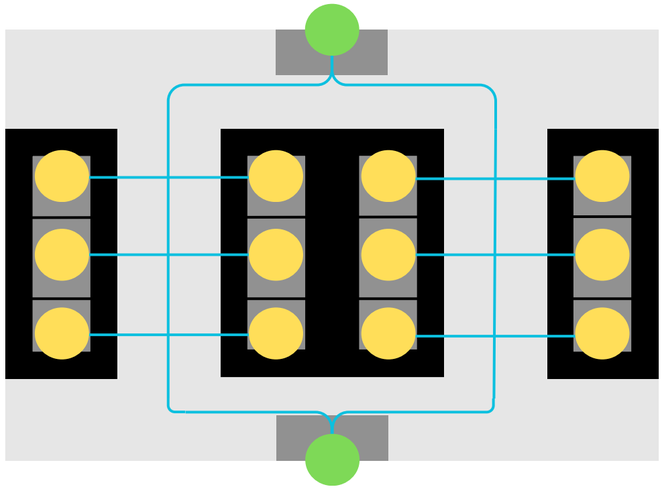
interesting properties

Exhilaratingly, Our algorithm is capable to deal with various warehouse structure. All we need to do is change the input, building a different graph model, and test and find the optimal coefficient in the formulas. Run the algorithm on the input, then it will find a optimal layout.

$w_{\text{freq}}, w_{\text{cooc}}$



$w_{depot}, w_{affinity}$



*As a matter of fact, the optimal coeffient in our formulas are related to the warehouse structure.

Looking Back

limitations & Potential Improvements

- **Failure to Consider Blocks with Different Capacity**

We still keep the assumption that **every block has the same capacity(or volume)**, which is not often the case. We can take it into consideration by **changing some input** without changing the core of our algorithm.

- **Failure to Consider Overflow**

Our current algorithm **assigns each block to store only one type of item** and does not allow overflow into secondary blocks unless the primary ones are full. In reality, allowing controlled overflow — where a small portion of a block can be shared between multiple items — could lead to **more balanced space usage** and improved flexibility.

- **Evaluation System is not Well-Developed**

The method that we evaluate layout is **approximate**. Calculating the exact effort can be another project.

What did we learn

- To formulate computational problem
- To look at problem from multiple angles (different stakeholders)
- How to apply CT to real world issue
- To assign tasks and split works so 1+1+1+1+1 > 5