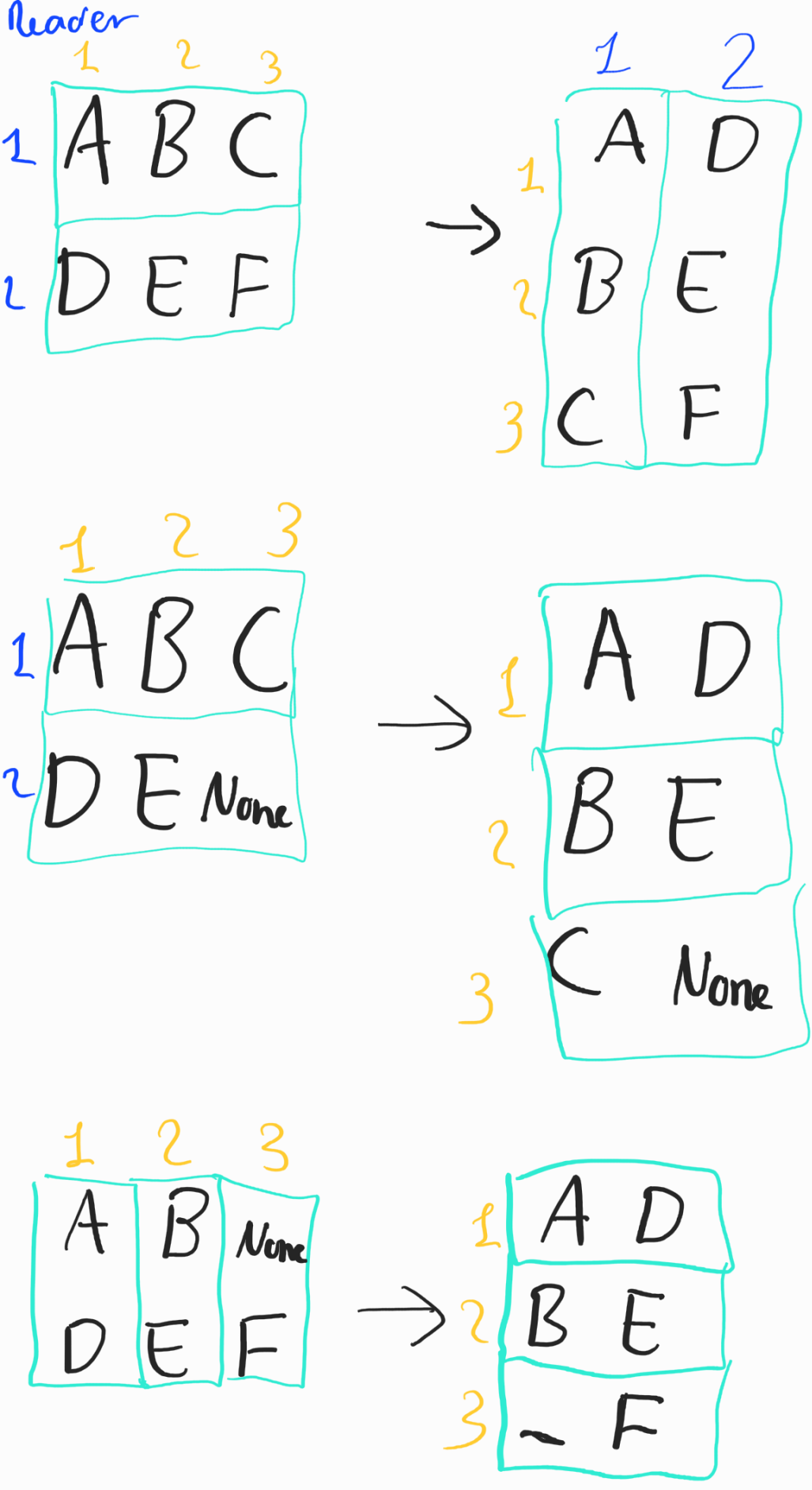
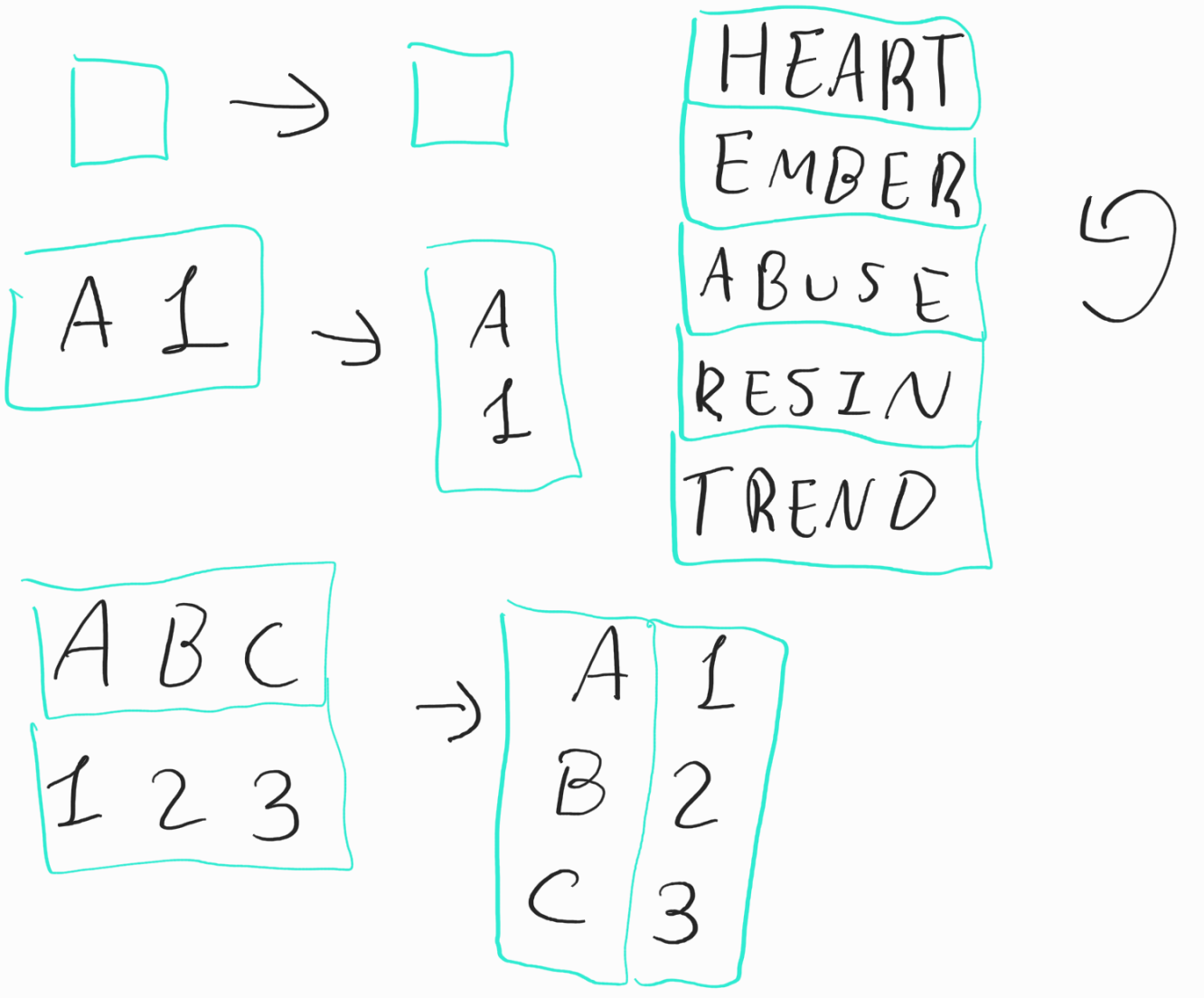

Reader
1 2 3
1 A B C
2 D E F
1 2
1 A D
2 B E
3 C F
1 2 3
1 A B C
2 D E None
1 A D
2 B E
3 C None
1 2 3
A B None
D E F
1 A D
2 B E
3 - F



Program needs to know max row length
Then it needs to fix rows with no full max row length and iterates over fixed matrix to get the new one.

A single book costs $8

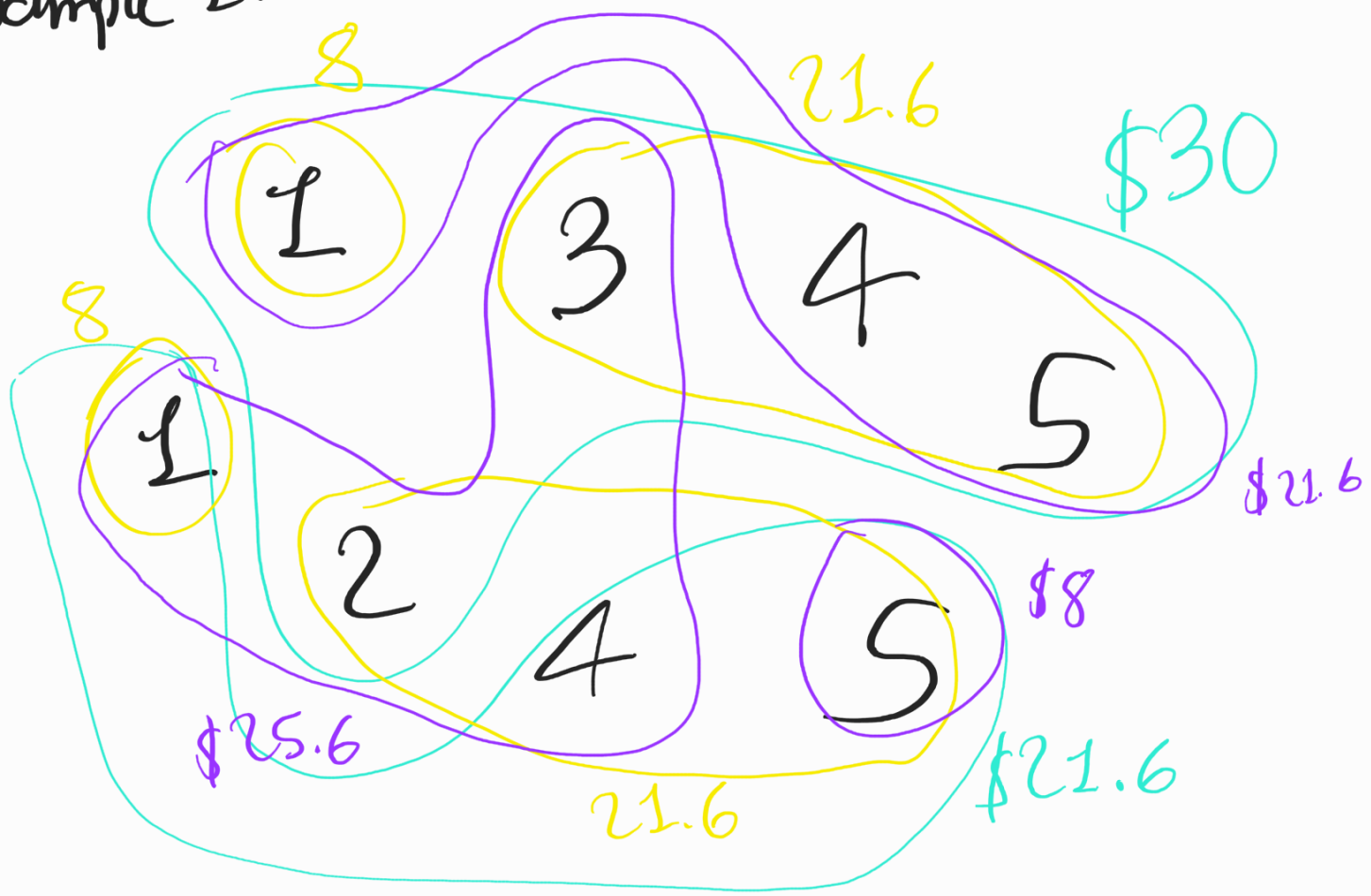
2 diff → 5% = 16 × 0.05 = 0.8 disc

3 diff → 10% = 24 × 0.1 = 2.4 disc

4 diff → 20% = 32 × 0.2 = 6.4 disc

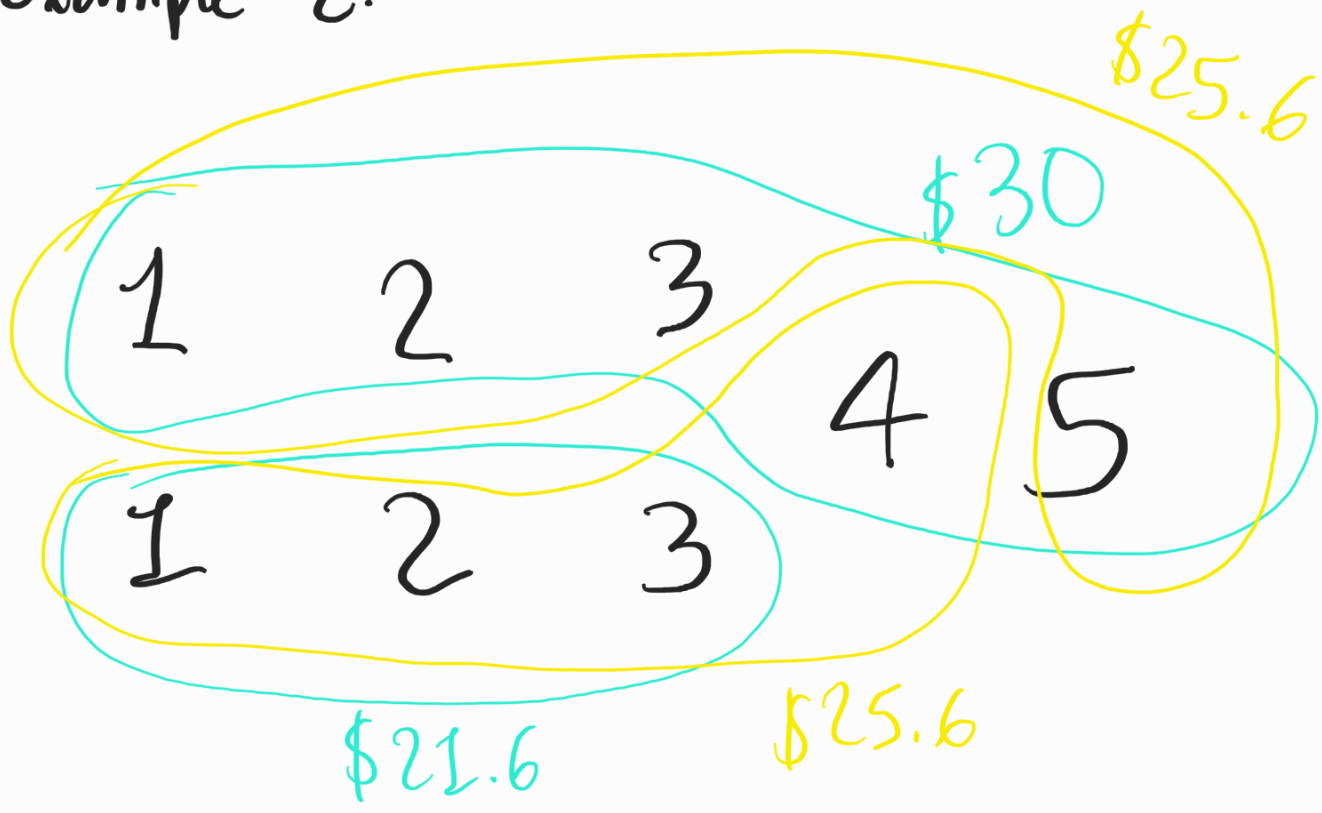
5 diff → 25% = 40 × 0.25 = 10 disc

Example 1:



$51.6 $59.2 $55.2 ...

Example 2:

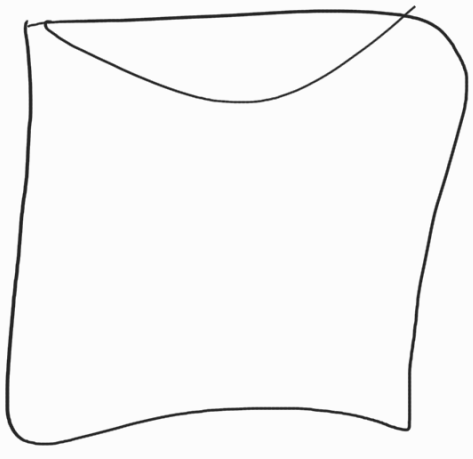


$51.6 $51.2 ...

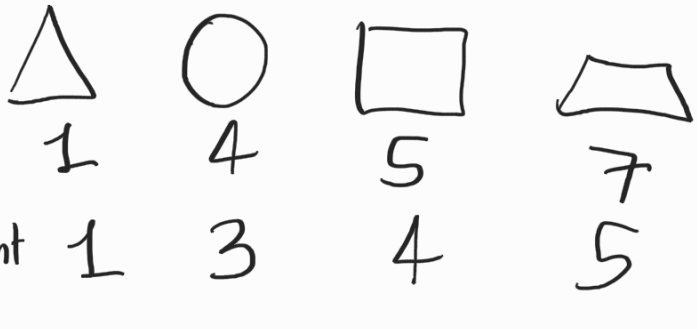
It could be implemented by using brute force but that is not the idea because of huge calculations and complexity...

Dynamic Programming Approach:

I think this problem is quite similar as knapsack problem...



W = 7



n = 4

Val	weight		max weight 0	1	2	3	4	5	6	7
0	0		0	0	0	0	0	0	0	0
1	1		0	1	1	1	1	1	1	1
4	3		0	1	1	4	5	5	5	5
5	4		0	1	1	4	5	6	6	9
7	5		0	1	1	4	5	7	8	9

Another approximation:

[1, 1, 2, 2, 3, 3, 4, 5]

L = Book G = Group of distincts

L1	L2	L3	L4	L5
1	1	1	1	1
1	1	1	0	0

= G1 0, G2 0, G3 1, G4 0, G5 1 = 51.6

X = 5 forever

Y = max numbers of a book
= 2

L1	L2	L3	L4	L5
1	1	1	0	1
1	1	1	1	0

= G1 0, G2 0, G3 0, G4 2, G5 0 = 51.2

[1, 1, 2, 2, 3, 3, 4, 4, 5, 5, 1]

L1	L2	L3	L4	L5			
1	1	1	1	1		G1	1
1	1	1	1	1	=	G2	0
1	0	0	0	0		G3	0
						G4	0
						G5	2

= 68

L1	L2	L3	L4	L5		
1	1	1	1	1		0
1	0	1	1	1	=	1
1	1	0	0	0		0
						1
						1

= 70.8

L1 L2 L3 L4 L5

1	1	1	1	1		0
1	0	0	1	1	=	0
1	1	1	0	0		2
						0
						1

$= 73.2$

It works but complexity is too high...

1 → 0
2 → 0.8
3 → 2.4
4 → 6.4
5 → 10

Maximize discount

Latest approach

① Permutate every row

② Try all combinations by selecting each possible row

Second DP approach:

A good approach could be maximize the discount, on each iteration.

The DP matrix could be thought like this:

X = max number of repeated books

Y = All possible groups $P(N)$