pack problem

Problem description:

N items

1, …, N

Weight: W(1), …, W(N)

Value: V(1), …, V(N)

Now a bag with max weight M, then how to pick up items to maximize the value?

a. Simple 0-1 pack problem

For each item, it can be chosen only once.

Solution 1:

Define F(i, w) as the maximum value for first i items in less than weight w.

F(i, w) = max{F(i-1, w-W(i)) + V(i), F(i-1, w)}

Meanings:

When i'th item comes, it can be chosen, or not be chosen. If chosen, the value is F(i-1, w-W(i)) + V(i). If not, the value is F(i-1, w).

Examples:

Weight: 3, 4, 2, 6, 1

Value: 5, 8, 9, 5, 3

Max Weight: 10

1. initial time

	1	2	3	4	5	6	7	8	9	10
0	0	0	0	0	0	0	0	0	0	0
1										
2										
3										
4										
5										

2. first round

	1	2	3	4	5	6	7	8	9	10
0	0	0	0	0	0	0	0	0	0	0
1 (weight 3)	0	0	5	5	5	5	5	5	5	5
2										

3										
4										
5										

3. second round

	1	2	3	4	5	6	7	8	9	10
0	0	0	0	0	0	0	0	0	0	0
1	0	0	5	5	5	5	5	5	5	5
2 (weight 4)	0	0	5	8	8	8	13	13	13	13
3										
4										
5										

4. third round

	1	2	3	4	5	6	7	8	9	10
0	0	0	0	0	0	0	0	0	0	0
1	0	0	5	5	5	5	5	5	5	5
2	0	0	5	8	8	8	13	13	13	13
3 (weight 2)	0	9	9	9	14	17	17	17	21	21
4										
5										

5. 4th round

	1	2	3	4	5	6	7	8	9	10
0	0	0	0	0	0	0	0	0	0	0
1	0	0	5	5	5	5	5	5	5	5
2	0	0	5	8	8	8	13	13	13	13
3	0	9	9	9	14	17	17	17	21	21
4 (weight 6)	0	9	9	9	14	17	17	17	21	21
5										

6. 5th round

	1	2	3	4	5	6	7	8	9	10
0	0	0	0	0	0	0	0	0	0	0
1	0	0	5	5	5	5	5	5	5	5
2	0	0	5	8	8	8	13	13	13	13
3	0	9	9	9	14	17	17	17	21	21
4	0	9	9	9	14	17	17	17	21	21
5 (weight 1)	3	9	12	12	14	17	20	20	21	24

If we want to be exactly same as weight M, how to do?

1. initial time

	0	1	2	3	4	5	6	7	8	9	10
0	0	-Inf	-Inf	-Inf	-Inf	-Inf	-Inf	-Inf	-Inf	-Inf	-Inf
1											
2											

3											
4											
5											

2. first round

	0	1	2	3	4	5	6	7	8	9	10
0	0	-Inf	-Inf	-Inf	-Inf	-Inf	-Inf	-Inf	-Inf	-Inf	-Inf
1 (weight 3)	0	-Inf	-Inf	5	-Inf	-Inf	-Inf	-Inf	-Inf	-Inf	-Inf
2											
3											
4											
5											

3. second round

	0	1	2	3	4	5	6	7	8	9	10
0	0	-Inf	-Inf	-Inf	-Inf	-Inf	-Inf	-Inf	-Inf	-Inf	-Inf
1	0	-Inf	-Inf	5	-Inf	-Inf	-Inf	-Inf	-Inf	-Inf	-Inf
2 (weight 4)	0	0	0	5	8	-Inf	-Inf	13	-Inf	-Inf	-Inf
3											
4											
5											

4. third round

	0	1	2	3	4	5	6	7	8	9	10
0	0	-Inf	-Inf	-Inf	-Inf	-Inf	-Inf	-Inf	-Inf	-Inf	-Inf
1	0	-Inf	-Inf	5	-Inf	-Inf	-Inf	-Inf	-Inf	-Inf	-Inf
2	0	0	0	5	8	-Inf	-Inf	13	-Inf	-Inf	-Inf
3 (weight 2)	0	0	9	9	9	14	17	-Inf	-Inf	21	-Inf
4	…										
5	…										

Finally, if the value is not -Inf, then it can be with exactly weight M.

Solution 2:

Define F(w) as the maximum value for first i items in less than weight w.

F(w) = max{F(w-W(i)) + V(i), F(w)} where w from M to W(i) in decrease order

Meanings:

Compared with equation in Solution 1: F(i, w) = max{F(i-1, w-W(i)) + V(i), F(i-1, w)}. If we move from M to W(i), the F(w-W(i)) + V(i) can replace F(i-1, w-W(i)) + V(i). Then we can use 1 * M table in the loop.

Examples:

Weight: 3, 4, 2, 6, 1

Value: 5, 8, 9, 5, 3

Max Weight: 10

1. initial time

	1	2	3	4	5	6	7	8	9	10
0	0	0	0	0	0	0	0	0	0	0
1										
2										
3										
4										
5										

2. first round

	1	2	3	4	5	6	7	8	9	10
0	0	0	0	0	0	0	0	0	0	0
1 (weight 3)	0	0	5	5	5	5	5	5	5	5(start point)
2										
3										
4										
5										

3. second round

	1	2	3	4	5	6	7	8	9	10
0	0	0	0	0	0	0	0	0	0	0
1	0	0	5	5	5	5	5	5	5	5
2 (weight 4)	0	0	5	8	8	8	13	13	13	13(start point)
3										
4										
5										

4. third round

	1	2	3	4	5	6	7	8	9	10
0	0	0	0	0	0	0	0	0	0	0
1	0	0	5	5	5	5	5	5	5	5
2	0	0	5	8	8	8	13	13	13	13
3 (weight 2)	0	9	9	9	14	17	17	17	21	21(start point)

4										
5										

5. 4th round

	1	2	3	4	5	6	7	8	9	10
0	0	0	0	0	0	0	0	0	0	0
1	0	0	5	5	5	5	5	5	5	5
2	0	0	5	8	8	8	13	13	13	13
3	0	9	9	9	14	17	17	17	21	21
4 (weight 6)	0	9	9	9	14	17	17	17	21	21(start point)
5										

6. 5th round

	1	2	3	4	5	6	7	8	9	10
0	0	0	0	0	0	0	0	0	0	0
1	0	0	5	5	5	5	5	5	5	5
2	0	0	5	8	8	8	13	13	13	13
3	0	9	9	9	14	17	17	17	21	21
4	0	9	9	9	14	17	17	17	21	21
5 (weight 1)	3	9	12	12	14	17	20	20	21	24(start point)

The table is same as Solution 1. However, we start to fill the table from right rather than that from left in Solution 2. In this method, we can only keep 1*M matrix. However, we need to keep N*M matrix in Solution 1.

b. Complete pack problem

For each item, it can be chosen any times or limited times T.

Solution 1:

Define F(i, w) as the maximum value for first i types of items in less than weight w.

F(i, w) = max{F(i-1, w-k*W(i)) + k*V(i) | 0<=k*W(i)<=w or k <=T}

Meanings:

When i'th type of item comes, it can be chosen k times where 0<=k*W(i)<=w or k<=T.

Examples:

Weight: 3, 4, 2, 6, 1

Value: 5, 8, 9, 5, 3

Max Weight: 10

1. initial time

	1	2	3	4	5	6	7	8	9	10
0	0	0	0	0	0	0	0	0	0	0
1										
2										
3										
4										
5										

2. first round

	1	2	3	4	5	6	7	8	9	10
0	0	0	0	0	0	0	0	0	0	0
1 (weight 3)	0	0	5	5	5	10	10	10	15	15
2										
3										
4										
5										

3. second round

	1	2	3	4	5	6	7	8	9	10
0	0	0	0	0	0	0	0	0	0	0
1	0	0	5	5	5	10	10	10	15	15
2 (weight 4)	0	0	5	8	8	8	13	16	16	16
3										
4										
5										

4. third round

	1	2	3	4	5	6	7	8	9	10
0	0	0	0	0	0	0	0	0	0	0
1	0	0	5	5	5	10	10	10	15	15
2	0	0	5	8	8	8	13	16	16	16
3 (weight 2)	0	9	9	18	18	27	27	36	36	45
4										
5										

5. fourth round

	1	2	3	4	5	6	7	8	9	10
0	0	0	0	0	0	0	0	0	0	0
1	0	0	5	5	5	10	10	10	15	15
2	0	0	5	8	8	8	13	16	16	16
3	0	9	9	18	18	27	27	36	36	45
4 (weight 6)	0	9	9	18	18	27	27	36	36	45
5										

6. fifth round

	1	2	3	4	5	6	7	8	9	10
0	0	0	0	0	0	0	0	0	0	0
1	0	0	5	5	5	10	10	10	15	15
2	0	0	5	8	8	8	13	16	16	16
3	0	9	9	18	18	27	27	36	36	45
4	0	9	9	18	18	27	27	36	36	45
5 (weight 1)	3	9	9	18	18	27	27	36	36	45

Solution 2:

Define F(w) as the maximum value less than weight w.

F(w) = max{F(w), F(w-Wi)+Vi}

Meanings:

When one item in i'th type comes, it can be chosen or chosen. If chosen, F(w-Wi)+Vi. If not, F(w).

Hard to understand. The best way to understand theories is to using examples.

Examples:

Weight: 3, 4, 2, 6, 1

Value: 5, 8, 9, 5, 3

Max Weight: 10

1. initial time

	1	2	3	4	5	6	7	8	9	10
0	0	0	0	0	0	0	0	0	0	0
1										
2										
3										
4										
5										

2. first round

	1	2	3	4	5	6	7	8	9	10
0	0	0	0	0	0	0	0	0	0	0
1 (weight 3)	0	0	5	5	5	10	10	10	15	15
2										
3										
4										
5										

3. second round

	1	2	3	4	5	6	7	8	9	10
0	0	0	0	0	0	0	0	0	0	0
1	0	0	5	5	5	10	10	10	15	15
2 (weight 4)	0	0	5	8	8	8	13	16	16	16
3										
4										
5										

4. third round

	1	2	3	4	5	6	7	8	9	10
0	0	0	0	0	0	0	0	0	0	0
1	0	0	5	5	5	10	10	10	15	15
2	0	0	5	8	8	8	13	16	16	16
3 (weight 2)	0	9	9	18	18	27	27	36	36	45
4										
5										

5. fourth round

	1	2	3	4	5	6	7	8	9	10
0	0	0	0	0	0	0	0	0	0	0
1	0	0	5	5	5	10	10	10	15	15
2	0	0	5	8	8	8	13	16	16	16
3	0	9	9	18	18	27	27	36	36	45
4 (weight 6)	0	9	9	18	18	27	27	36	36	45
5										

6. fifth round

	1	2	3	4	5	6	7	8	9	10
0	0	0	0	0	0	0	0	0	0	0
1	0	0	5	5	5	10	10	10	15	15
2	0	0	5	8	8	8	13	16	16	16
3	0	9	9	18	18	27	27	36	36	45
4	0	9	9	18	18	27	27	36	36	45
5 (weight 1)	3	9	9	18	18	27	27	36	36	45

This tables in Solution 2 is same as that in Solution 1. But Solution 2 is simpler in computation complexity as it will not calculate the max among k choices.

c. Pack problem with dependence

For item j, it only depends on item i, that means if we want to choose j, item i must be chosen ahead. And at same time, no other items depend on item j.

Solution:

We group items with dependences into K groups. Define F(k, w) as the maximum value for first k group of items in less than weight w.

F(k, w) = max{F(k-1, w-Wk)+Vk}

Where Wk, Vk are combinations in group k. For example, in group k, 3 items W ={2, 3, 4} V= {30, 40 ,50}, the first item is master and left 2 items are relied on the first item. {Wk, Vk} can be {0, 0}, {2, 30}, {5, 70}, {6, 80}, {9, 120}

Examples:

Items: weight, value, dependency

Item 1: 4 2 0

Item 2: 4 5 1

Item 3: 3 5 1

Item 4: 2 3 0

Item 5: 3 3 4

Item 6: 5 2 0

If dependency is 0, it is independent item. If not 0, it depends on the item {dependency}

Max weight: 10

	0	1	2	3	4	5	6	7	8	9	10
0	0	0	0	0	0	0	0	0	0	0	0
K=1	0	0	0	0	2	2	2	7	7	7	7
K=2	0	0	3	3	3	6	6	7	7	10	10
K=3	0	0	3	3	3	6	6	7	7	10	10