KNAPSACK PROBLEM

A dynamic approach

Knapsack Problem

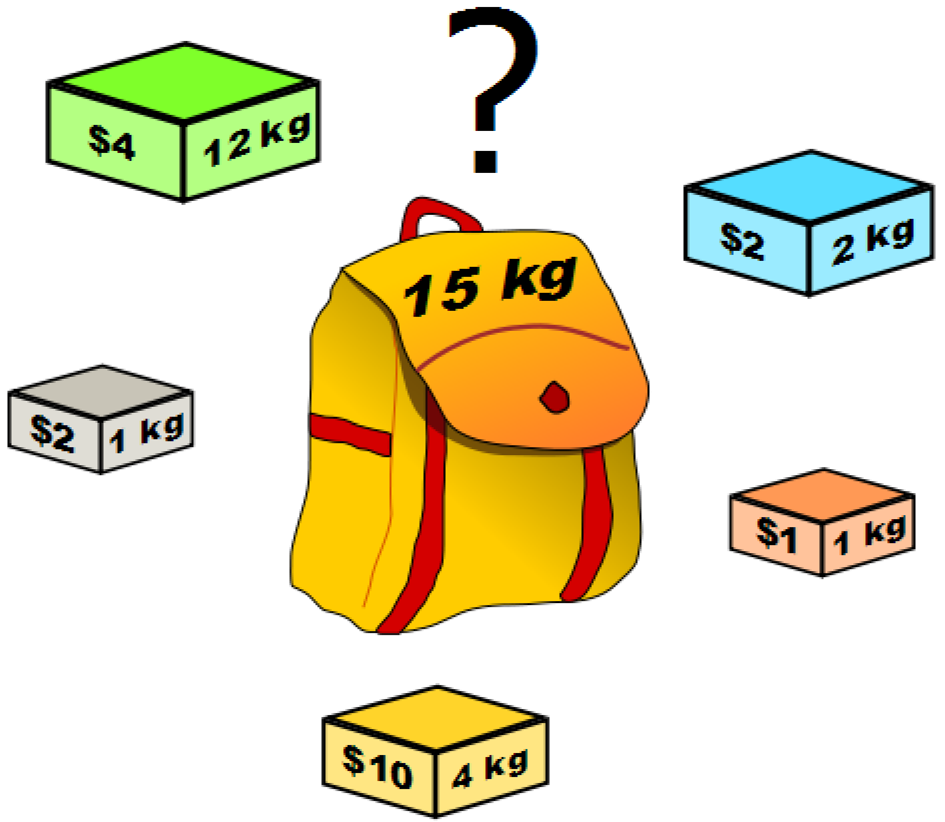
- Given a sack, able to hold **W** kg
- Given a list of objects
  - Each has a weight and a value
- Try to pack the object in the sack so that the total value is maximized

Variation

- Rational Knapsack
  - Object is like a gold bar, we can cut it in to piece with the same value/weight
- 0-1 Knapsack
  - Object cannot be broken, we have to choose to take (1) or leave (0) the object
  - E.g.
    - K = 50
    - Objects = (60,10) (100,20) (120,30)
    - Best solution = second and third

The Problem

- Input
  - A number W, the capacity of the sack
  - n pairs of weight and price $((w_1,p_1),(w_2,p_2),\ldots,(w_n,p_n))$
    - w_i = weight of the i^{th} items
    - p_i = price of the i^{th} item
- Output
  - A subset S of {1,2,3,…,n} such that
    - $\sum_{i \in S} p_i$ is maximum
    - $\sum_{i \in S} w_i < W$



from http://en.wikipedia.org/wiki/File:Knapsack.svg

Naïve approach

- Try every possible combination of {1,2,3,…n}
- Test whether a combination satisfies the weight constraint
  - If so, remember the best one
- This gives $O(2^n)$

Dynamic Approach

- Let us assume that W (the maximum weight) and w_i are integers
- Let us assume that we just want to know "the best total price", i.e., $\sum_{i \in S} p_i$
  - (well, soon we will see that this also leads to the actual solution
- The problem can be solved by a dynamic programming
  - How?
  - What should be the subproblem?
  - Is it overlapping?

The Sub Problem

- What shall we divide?
  - The number of items?
    - Let’s try half of the items?

  - what about the weight?

The Optimal Solution

- Assume that we know the actual optimal solution to the problem
  - The solution consist of item {2,5,6,7}
  - What if we takes the item number 7 out?
    - What can we say about the set of {2,5,6}
    - Is it an optimal solution of any particular problem?

The Optimal Solution

- Let $K(b)$ be the "best total value" when W equals to b
- If the i^th item is in the best solution
  - $K(W) = K(W - w_i) + p_i$
- But, we don't really know that the i^th item is in the optimal solution
  - So, we try everything
  - $K(W) = max_{1 \leq i \leq n}(K(W - w_i) + p_i)$
- Is this our algorithm?

  - Yes, if and only if we allow each item to be selected multiple times (that is not true for this problem)

Solution

- We need to keep track whether the item is used
- What if we know the optimal solution when i^{th} items is not being used?
  - Also for every size of knapsack from 0 to W

- Then, with additional i^{th} item, we have only two choices, use it or not use it

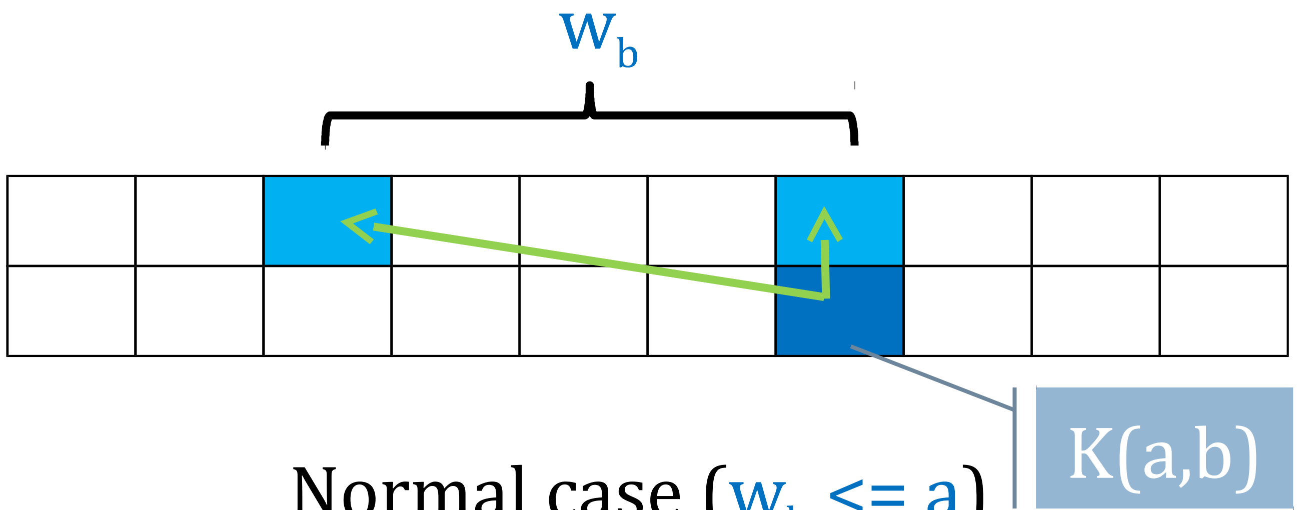
The Recurrence

- $K(a,b)$ = the best total price when the knapsack is of size a and only item number 1 to number b is co nsidered

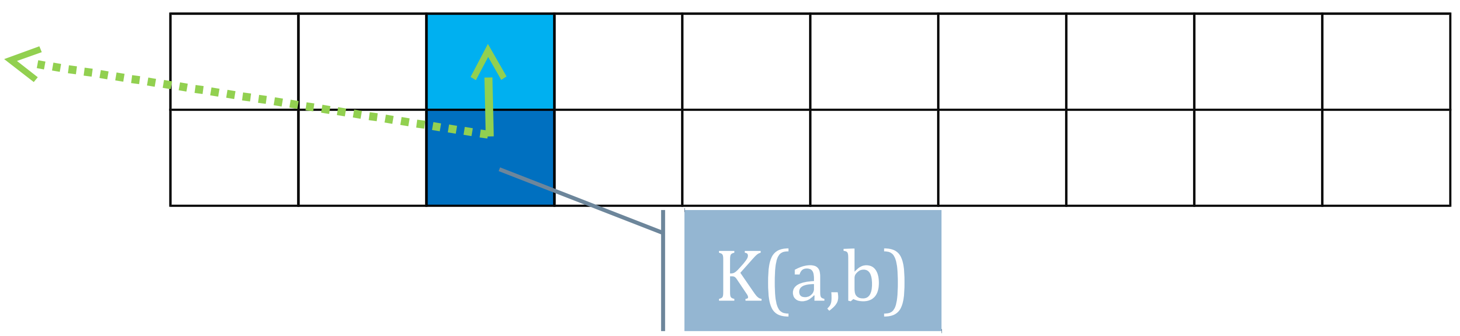
- $K(a,b) = \max(K(a - w_b, b - 1) + p_b, K(a, b - 1))$
- $K(a,b) = 0$ when $a = 0$ or $b = 0$
- $K(a,b) = K(a,b-1)$ when $w_b > a$

- The solution is at $K(W,n)$

The Recurrent



Normal case (w_b <= a)



Too much weight (w_b > a)

Example

p = {4, 2, 2, 1, 10}
w ={12, 2, 1, 1, 4}   W = 15

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0																
1																
2																
3																
4																
5																

Example

p = {4, 2, 2, 1, 10}
w ={12, 2, 1, 1, 4}   W = 15

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0															
2	0															
3	0															
4	0															
5	0															

$K(a,b) = 0$ when $a = 0$ or $b = 0$

Example

p = {4, 2, 2, 1, 10}
w ={12, 2, 1, 1, 4}   W = 15

Fill row 1   (p_1=4 w_1=12)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0															
2	0															
3	0															
4	0															
5	0															

Example

p = {4, 2, 2, 1, 10}
w ={12, 2, 1, 1, 4}   W = 15

Fill row 1   (p_1=4 w_1=12)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0				
2	0															
3	0															
4	0															
5	0															

K(a,b) = K(a,b-1)   when $w_b > a$

Example

p = {4, 2, 2, 1, 10}
w ={12, 2, 1, 1, 4}   W = 15

Fill row 1   (p_1=4 w_1=12)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	4	4	4	4
2	0															
3	0															
4	0															
5	0															

$K(a,b) = \max(\, K(a - w_b, b - 1) + p_b\,,\ K(a, b - 1)\,)$

Example

$p = \{4, 2, 2, 1, 10\}$
$w = \{12, 2, 1, 1, 4\}$ $W = 15$

Fill row 2 $(p_2=2 \quad w_2=2)$

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	4	4	4	4
2	0	0														
3	0															
4	0															
5	0															

$K(a,b) = K(a,b\text{-}1)$ when $w_b > a$

Example

$p = \{4, 2, 2, 1, 10\}$
$w = \{12, 2, 1, 1, 4\}$ $W = 15$

Fill row 2 ($p_2=2$ $w_2=2$)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	4	4	4	4
2	0	0														
3	0															
4	0															
5	0															

$K(a,b) = \max(K(a - w_b, b - 1) + p_b, K(a, b - 1))$

Example

p = {4, 2, 2, 1, 10}
w ={12, 2, 1, 1, 4}   W = 15

Fill row 2   (p_2=2 w_2=2)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	4	4	4	4
2	0	0	2	2	2	2	2	2	2	2	2	2	4	4	6	6
3	0															
4	0															
5	0															

$K(a,b) = \max(K(a - w_b, b-1) + p_b, K(a, b-1))$

Example

p = {4, 2, 2, 1, 10}
w ={12, 2, 1, 1, 4}   W = 15

Fill row 2   (p_2=2 w_2=2)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	4	4	4	4
2	0	0	2	2	2	2	2	2	2	2	2	2	4	4	6	6
3	0															
4	0															
5	0															

$K(a,b) = \max(\, K(a - w_b, b - 1) + p_b\,,\; K(a, b - 1)\,)$

Example

$p = \{4, 2, 2, 1, 10\}$
$w = \{12, 2, 1, 1, 4\}$ $W = 15$

Fill row 3 ($p_3=2$ $w_3=1$)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	4	4	4	4
2	0	0	2	2	2	2	2	2	2	2	2	2	4	4	6	6
3	0	2	2	4	4	4	4	4	4	4	4	4	4	6	6	8
4	0															
5	0															

Example

$p = \{4, 2, 2, 1, 10\}$
$w = \{12, 2, 1, 1, 4\}$ $W = 15$

Fill row 3 ($p_3=2$ $w_3=1$)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	4	4	4	4
2	0	0	2	2	2	2	2	2	2	2	2	2	4	4	6	6
3	0	2	2	4	4	4	4	4	4	4	4	4	4	6	6	8
4	0															
5	0															

Example

$p = \{4, 2, 2, 1, 10\}$
$w = \{12, 2, 1, 1, 4\}$ $W = 15$

Fill row 4 ($p_4=1$ $w_4=1$)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	4	4	4	4
2	0	0	2	2	2	2	2	2	2	2	2	2	4	4	6	6
3	0	2	2	4	4	4	4	4	4	4	4	4	4	6	6	8
4	0	2	3	4	5	5	5	5	5	5	5	5	5	6	7	8
5	0															

Example

$p = \{4, 2, 2, 1, 10\}$
$w = \{12, 2, 1, 1, 4\}$ $W = 15$

Fill row 4 ($p_4=1$ $w_4=1$)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	4	4	4	4
2	0	0	2	2	2	2	2	2	2	2	2	2	4	4	6	6
3	0	2	2	4	4	4	4	4	4	4	4	4	4	6	6	8
4	0	2	3	4	5	5	5	5	5	5	5	5	5	6	7	8
5	0															

Example

p = {4, 2, 2, 1, 10}
w ={12, 2, 1, 1, 4}   W = 15

Fill row 5   (p_5=10 w_5=4)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	4	4	4	4
2	0	0	2	2	2	2	2	2	2	2	2	2	4	4	6	6
3	0	2	2	4	4	4	4	4	4	4	4	4	4	6	6	8
4	0	2	3	4	5	5	5	5	5	5	5	5	5	6	7	8
5	0	2	3	4	10	12	13	14	15	15	15	15	15	15	15	15

Example

p = {4, 2, 2, 1, 10}
w ={12, 2, 1, 1, 4}   W = 15

Fill row 5   (p_5=10 w_5=4)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	4	4	4	4
2	0	0	2	2	2	2	2	2	2	2	2	2	4	4	6	6
3	0	2	2	4	4	4	4	4	4	4	4	4	4	6	6	8
4	0	2	3	4	5	5	5	5	5	5	5	5	5	6	7	8
5	0	2	3	4	10	12	13	14	15	15	15	15	15	15	15	15

Example

p = {4, 2, 2, 1, 10}
w ={12, 2, 1, 1, 4}   W = 15

Trace the solution

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	4	4	4	4
2	0	0	2	2	2	2	2	2	2	2	2	2	4	4	6	6
3	0	2	2	4	4	4	4	4	4	4	4	4	4	6	6	8
4	0	2	3	4	5	5	5	5	5	5	5	5	5	6	7	8
5	0	2	3	4	10	12	13	14	15	15	15	15	15	15	15	15

The Code

```
set all K[0][j] = 0 and all K[w][0] = 0
for j = 1 to n
    for w = 1 to W
        if (w_j > W)
            K[w][j] = K[w][j - 1];
        else
            K[w][j] = max( K[w - w_i][j - 1] + p_i ,
                           K[W][j - 1] )
return K[W][n];
```