Knapsack Problem Notes

V.S. Subrahmanian
University of Maryland

Knapsack Problem

- You have a knapsack that has capacity (weight) C.
- You have several items $I_1,\ldots,I_n$.
- Each item I_j has a weight w_j and a benefit b_j.
- You want to place a certain number of copies of each item I_j in the knapsack so that:
  - The knapsack weight capacity is not exceeded and
  - The total benefit is maximal.

Example

Item	**Weight**	**Benefit**
A	2	60
B	3	75
C	4	90

Capacity = 5

Key question

- Suppose f(w) represents the *maximal possible benefit* of a knapsack with weight w.
- We want to find (in the example) f(5).
- Is there anything we can say about f(w) for arbitrary w?

Key observation

- To fill a knapsack with items of weight w, we must have added items into the knapsack in some order.
- Suppose the last such item was I_i with weight w_i and benefit b_i.
- Consider the knapsack with weight $(w- w_i)$. Clearly, we chose to add I_i to this knapsack because of all items with weight w_i or less, I_i had the max benefit b_i.

Key observation

- Thus, $f(w) = MAX \{ b_j + f(w-w_j) \mid I_j \text{ is an item}\}$.
- This gives rise to an immediate recursive algorithm to determine how to fill a knapsack.

Example

Item	**Weight**	**Benefit**
A	2	60
B	3	75
C	4	90

f(0), f(1)

- f(0) = 0. Why? The knapsack with capacity 0 can have nothing in it.
- f(1) = 0. There is no item with weight 1.

f(2)

- f(2) = 60. There is only one item with weight 60.
- **Choose A.**

f(3)

- $f(3) = \text{MAX} \{ b_j + f(w-w_j) \mid I_j \text{ is an item}\}$.

$= \text{MAX} \{ 60+f(3-2), 75 + f(3-3)\}$

$= \text{MAX} \{ 60 + 0, 75 + 0 \}$

$= 75$.

Choose B.

f(4)

- $f(4) = \text{MAX} \{ b_j + f(w-w_j) \mid I_j \text{ is an item}\}$.

$= \text{MAX} \{ 60 + f(4-2), 75 + f(4-3), 90+f(4-4)\}$

$= \text{MAX} \{ 60 + 60, 75 + f(1), 90 + f(0)\}$

$= \text{MAX} \{ 120, 75, 90\}$

$=120$.

Choose A.

f(5)

- $f(5) = \text{MAX} \{ b_j + f(w-w_j) \mid I_j \text{ is an item}\}$.

$= \text{MAX} \{ 60 + f(5-2), 75 + f(5-3), 90+f(5-4)\}$

$= \text{MAX} \{ 60 + f(3), 75 + f(2), 90 + f(1)\}$

$= \text{MAX} \{ 60 + 75, 75 + 60, 90+0\}$

$= 135$.

Choose A or B.

Result

- Optimal knapsack weight is 135.
- Two possible optimal solutions:
  - Choose A during computation of f(5). Choose B in computation of f(3).
  - Choose B during computation of f(5). Choose A in computation of f(2).
- Both solutions coincide. Take A and B.

Another example

- Knapsack of capacity 50.
- 3 items
  - Item 1 has weight 10, benefit 60
  - Item 2 has weight 20,benefit 100
  - Item 3 has weight 30, benefit 120.

f(0),..,f(9)

- All have value 0.

f(10),..,f(19)

- All have value 60.
- **Choose Item 1.**

f(20),..,f(29)

- F(20) = MAX { 60 + f(10), 100 + f(0) }

= MAX { 60+60, 100+0}

=120.

Choose Item 1.

f(30),…,f(39)

- f(30) = MAX { 60 + f(20), 100 + f(10), 120 + f(0) }

= MAX { 60 + 120, 100+60, 120+0}

= 180

Choose item 1.

$f(40),\ldots,f(49)$

- $F(40) = \text{MAX}\ \{ 60 + f(30), 100 + f(20), 120 + f(10)\}$

$= \text{MAX}\ \{ 60 + 180, 100+120, 120 + 60\}$

$= 240.$

Choose item 1.

f(50)

- f(50) = MAX { 60 + f(40), 100 + f(30), 120 + f(20) }

= MAX { 60 + 240, 100+180, 120 + 120}

= 300.

Choose item 1.

Knapsack Problem Variants

- 0/1 Knapsack problem: Similar to the knapsack problem except that for each item, only 1 copy is available (not an unlimited number as we have been assuming so far).
- Fractional knapsack problem: You can take a fractional number of items. Has the same constraint as 0/1 knapsack. Can solve using a greedy algorithm.

0/1 Knapsack Problem

- [Credit: next few slides use notation from Wikipedia]
- Assume all the objects are ordered in some order $O_1,..,O_n$.
- Let *m(i,w)* denote the best value we can get for a knapsack of weight w with only items selected from the first *i* objects.

"Initialization" Steps

- m(0,w) = 0. *If we can select nothing, then the value we get is 0.*
- m(i,0) = 0. *If the total weight allowed is zero, then we can put nothing in the knapsack and so get 0 benefit.*

"Recursive" Steps

- m(i,w) = m(i-1,w) if $w_i > w$. *If the new item (item O_i)* has a weight that exceeds the weight limit, then the best solution you have is the solution you had before.

“Recursive” Steps

- $m(i,w) = MAX(m(i-1,w),$
  $m(i-1,w-w_i)+b_i)$
  if $w_i \le w$.
- When the i’th element’s weight is less than w, then there are two possibilities:
  – O_i is not in the knapsack [case 1]
  – O_i is in the knapsack [case 2]

"Recursive" Step: Case 1

- $m(i,w) = MAX(m(i-1,w), m(i-1,w-w_i)+b_i)$

  if $w_i \leq w$.
- O_i is not in the knapsack [red case]
- If it is not in the knapsack, then all elements in it are from the first (i-1) elements, so their best benefit is $m(i-1,w)$.

"Recursive" Step: Case 2

- $m(i,w) = \text{MAX}(m(i\text{-}1,w),$
  $m(i\text{-}1,w\text{-}w_i)+b_i)$

  if $w_i \leq w$.
- O_i is in the knapsack [green case]
- If it is in the knapsack, then it occupies w_i weight. This means there is $(w\text{-}w_i)$ weight left in the knapsack which must be filled with the first (i-1) elements.
- That yields a benefit of $m(i\text{-}1,w\text{-}w_i)+b_i$

0/1 Knapsack Problem

- The above leads to an immediate recursive algorithm for the 0/1 knapsack problem.
- Given objects O1,..,On, and a weight bound *W*, just write the natural recursive algorithm using the definition.

0/1 Knapsack Iterative Algorithm with n objects and weight w.

Set m(0,w') = 0 for all $w' \leq w$.

Set m(i,0) = 0 for all i=1,..,n.

For i=1 to n do

    For w' = 0 to w do

        if $w_i \leq w$ then (* item O_i could be in the solution *)

            if $b_i + m(i-1,w- w_i) > m(i-1,w)$ then

                $m(i,w) = b_i + m(i-1,w- w_i)$

            else m(i.w) = m(i-1,w).

Means adding O_i to solution is best

Complexity is O(n*w).

Example

- 6 Objects O1,..,On. W=35.
- Weights and benefits are as shown in the table below.

Obj	O1	O2	O3	O4	O5	O6
Weight	5	10	20	15	10	20
Benefit	1	5	8	10	20	25

Initialization

- m(0,w')=0 for all w' < w.
- M(i,o) = 0 for all i=1,..,n.

Obj	**O1**	**O2**	**O3**	**O4**	**O5**	**O6**
Weight	5	10	20	15	10	20
Benefit	1	5	8	10	20	25

Processing O1

- m(0,w')=0 for all w' < w.
- M(i,o) = 0 for all i=1,..,n.
- Pick O1.
- m(1,w) =1 by putting O1 in the knapsack.

Obj	O1	O2	O3	O4	O5	O6
Weight	5	10	20	15	10	20
Benefit	1	5	8	10	20	25

Processing O2

- m(0,w')=0 for all w' < w.
- M(i,o) = 0 for all i=1,..,n.
- m(1,w)=1 by putting O1 in the knapsack.
- Add O2 to the knapsack.
- m(2,w)=6.

Obj	O1	O2	O3	O4	O5	O6
Weight	5	10	20	15	10	20
Benefit	1	5	8	10	20	25

Processing O3

- m(0,w')=0 for all w' < w.
- M(i,o) = 0 for all i=1,..,n.
- m(1,w)=1. m(2,w)=6. {O1,O2} are in.
- m(3,w)=8+5=13. Put {O2,O3} in knapsack.

Obj	O1	O2	O3	O4	O5	O6
Weight	5	10	20	15	10	20
Benefit	1	5	8	10	20	25

Processing O4

- m(0,w')=0 for all w' < w.
- m(i,o) = 0 for all i=1,..,n.
- m(1,w)=1. m(2,w)=6.
- m(3,w)=8+5=13.
- m(4,w) = 10 + m(3,35-15) = 10 + 8 =18. Put {O3,O4} in knapsack.

Obj	O1	O2	O3	O4	O5	O6
Weight	5	10	20	15	10	20
Benefit	1	5	8	10	20	25

Processing O5

- m(0,w')=0 for all w' < w.
- m(i,o) = 0 for all i=1,..,n.
- m(1,w)=1. m(2,w)=6.
- m(3,w)=8+5=13.
- m(4,w) = 10 + m(3,35-15) = 10 + 8 =18. Put {O3,O4} in knapsack.
- m(5,w) = 20 + m(35-10) = 20 + m(4,25) = 20+15 = 35. Put {O5,O2,O4} in knapsack.

Obj	O1	O2	O3	O4	O5	O6
Weight	5	10	20	15	10	20
Benefit	1	5	8	10	20	25

Processing O6

- m(0,w')=0 for all w' < w.
- m(i,o) = 0 for all i=1,..,n.
- m(1,w)=1. m(2,w)=6.
- m(3,w)=8+5=13.
- m(4,w) = 10 + m(3,35-15) = 10 + 8 =18. Put {O3,O4} in knapsack.
- m(5,w) = 20 + m(35-10) = 20 + m(4,25) = 20+15 = 35. Put {O5,O2,O4} in knapsack.
- M(6,w) = 25 + m(5,35-20) = 25 + m(5,15) = 25+25 = 50. Put {O6,O5,O2} in the knapsack. DONE/

Error here.
Refer to 01knapsack_revisited

Obj	O1	O2	O3	O4	O5	O6
Weight	5	10	20	15	10	20
Benefit	1	5	8	10	20	25

In-class exercise

- 6 Objects O1,..,O6. W=10.
- Weights and benefits are as shown in the table below.

Obj	O1	O2	O3	O4	O5	O6
Weight	2	1	4	2	3	5
Benefit	3	4	2	5	7	11