The 0/1 Knapsack Problem
(No fractions)

* Each item is represented by a pair <Value, weight> (weight $\in \mathbb{Z}^+$)

* The knapsack can accommodate items with a total weight of at most capacity W

Let, I be a vector of length n, representing the set of 'n' available item to be selected from.
Also, let V be a vector of length n to indicate whether an item ends up in the knapsack or not?
i.e
if V[i]=1 then I[i] is in the knapsack

The 0/1 knapsack problem is thus, finding a collection of items that maximizes

$$K(W) = \sum_{i=0}^{n-1} V[i] \times I[i]$$

Subject to the constraint $\sum_{i=0}^{n-1} V[i] \times I[i] \leq W$

where, W is the capacity of the knapsack.

Solving 0/1 Knapsack with Brute-Force

1) Enumerate all possible combinations of items (exponential!).
2) Remove all such combinations whose total weight exceeds W (knapsack's capacity)
3) From the remaining combinations, choose the one that maximizes the total value K(W)

Recall: Brute-force guarantees an optimal solution but at an exponential cost! Intractable when 'n' and 'W' are Large.

Date

0/1 knapsack : Two Possible Cases

- With Repitition (Unlimited quantity)
- Without Repitition (each item to be taken only once)

a binary decision tree

For example, Consider the following items with $W = 10$

	Weight	Value
a	6	30
b	3	14
c	4	16
d	2	9

1) 0/1 knapsack with Repitition:

Pick 'a' and two 'd's.

$\sum w_i = a.wt + d.wt + d.wt$
$= 6 + 2 + 2 = 10$

$K(w) = 30 + 2(9) = 48$
Total value

(2) 0/1 knapsack without Repitition:

Pick 'a' and 'c':

$\sum w_i = a.wt + c.wt$
$= 6 + 4 = 10$

Total value $K(w) = 30 + 16 = 46$

Recursive Optimal Substructure for 0/1 knapsack

Let $K(w)$ be the optimal solution, i.e, the maximum value with a given capacity of w.

Can we express $K(w)$ in terms of optimal subproblems?

Claim: The 0/1 knapsack has an optimal solution in Dynamic Programming in $O(nW)$.

pseudopolynomial

Signature

RC

No.

Recursive Optimal Subproblems: Given $K(w)$ the optimal solution.

Now, the input to the problem can be shortened in two ways:

1) Look at the knapsack with smaller capacity?
   i.e., $w_i \leq w$, $1 \leq \underbrace{w_i \leq w}_{\text{a smaller capacity knapsack.}}$

2) Look at fewer items (given 'n' items) i.e., 1, 2, 3, .., j; $j \leq n$

Let i be an item that is in the optimal solution, i.e., $K(w)$.

If we remove 'i' from the knapsack, the remaining items in the knapsack will constitute the optimal solution for a knapsack of weight (capacity) $w - w_i$ and will have a total value of $K(w - w_i)$, which will be optimal for a subproblem of size $w - w_i$.

Then,

$$\underbrace{K(w)}_{\text{the optimal solution}} = \underbrace{K(w - w_i)}_{\text{optimal soln for the subproblem}} + \underbrace{v_i}_{\text{value of item } i} \quad \text{; for some } i$$

Now, we don't know which item 'i' is to be removed.
So, we try all possibilities :-

i.e.,

$$K(w) = \max_{i : w_i \leq w} K(w - w_i) + v_i$$ { With Repitition allowed !

Bad recursion! But since we'll fill-in the table, there is no need to worry.

Base Case: $K(0) = 0$ { Smallest subproblem ($w = 0$). The optimal solution for $K(0) = 0$.

Signature

Filling-in the table $K(0) = 0$

For w_i : 1 to W

$$K(w) = \max_{w_i \leq w} \{ K(w - w_i) + V_i \}$$

∵ Items can be repeated here, the problem reduces to size by capacity only, so well, fill-in a 1D table

eg.:

w	0	1	2	3	4	5	6	7	8	9	10
$K(w)$	0	0	9	14	18	23	30	32	39	44	48

↑ starting with $K(w=0) = 0$

The items are

Item	Weight	Value
a	6	30
b	3	14
c	4	16
d	2	9

$K(w=1) = \max \{ K(w - w_i) + V_i \}$ for all $i : w_i \leq w$

$= \max \{ \underbrace{K(1-1)}_{=K(0)=0} + 0 \}$ ↑ No item of weight 1 $= 0$

$K(w=2) = \max \{ K(\underbrace{2 - w.d}_{2-2=0}) + V.d \} = \underbrace{K(0)}_{=0} + 9 = 9$

$K(w=3) = \max \{ K(\underbrace{3 - w.b}_{3-3=0}) + V.b, K(\underbrace{w - w.d}_{=K(1)=0}) + V.d \}$

$= \max \{ 0 + 14, 0 + 9 \} = 14$

Date

Similarly,

$$K(W=4) = \max\{K(4-w_b)+v_b, K(4-w_c)+v_c, K(4-w_d)+v_d\}$$

$$= \max\{\underbrace{K(4-3)}_{K(1)=0}+14, \underbrace{K(4-4)}_{K(0)=0}+16, \underbrace{K(4-2)}_{K(2)=9}+9\}$$

$$= \max\{14, 16, 18\} = 18$$

and so on. (Fill-in the rest of the table as HW)

HW

Use the dag from the table to find items in the final optimal soln for $W=10$

0/1 Knapsack Without Repetition (an item can be taken only once)

Recall, when repetitions were allowed, we only cared about $w_i \leq W$ [one-dimensional table]

Now, we need to keep track of $K(W)$ but also whether an item j $(1 \leq j \leq n)$ out of the given n items is already in the optimal solution or not! The problem is now two-dimensional (we'll fill in a 2D table of size $(n+1) \times (W+1)$

Let, $K(W, j)$ be the maximum value achievable for a knapsack with capacity W with items $1, 2, \ldots, j$.

How can we express the problem recursively?

The problem is $K(W, n)$:

Date

We have a Decision Problem :-

We're exploring the possibility whether a given item j is part of the optimal solution or not.

So, either the optimal solution exists when j is in the knapsack, or when it is not!

i.e.,

$$K(w,j) = \max \begin{cases} K(w - w_j, j-1) + v_j & \text{s.t. } w_j \leq w \\ K(w, j-1) \end{cases}$$

recall the LCS problem!

Let's take a concrete (toy) example from Dasgupta et al.

Item	Weight	Value
a	6	30
b	3	14
c	4	16
d	2	9

Smallest Subproblems (base-cases)

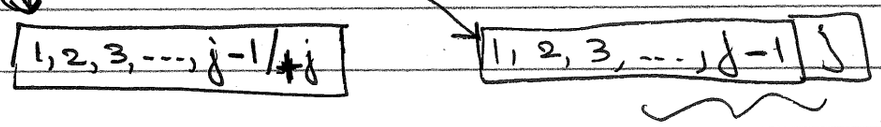
$K(0, n) = 0$

$K(w, 0) = 0$

$K(0, 0) = 0$

See,

$$K(w,j) = \max \begin{cases} K(w - w_j, j-1) + v_j & ; w_j \leq w \\ K(w, j-1) \end{cases}$$



Which j^{th} item are we talking about?

Signature

RC

No.

Date

Psuedocode to fill-in the 2D table:

1. Initialize all $K(0,j) = 0$ and $K(w,0) = 0$
2. For j: 1 to n
   For $w.j$: 1 to w
   If $w.j > w$
   $K(w,j) = K(w, j-1)$
   else
   $K(w,j) = \max \begin{Bmatrix} K(w-w.j, j-1) + v.j, \\ K(w, j-1) \end{Bmatrix}$

For our example:

	n	0	a	b	c	d
w	0	0	0	0	0	0
	1	0				
	2	0				
	3	0				
	4	0				
	5	0				
	6	0				
	7	0				
	8	0				
	9	0				
	10	0				

Hw: Fill-in the table!

(write a code!)

$O(nW)$

Pseudopolynomial!

[Next: Are we familiar with Pascal's Δ]

We'll return to Knapsack in Greedy Algorithms!

Signature

No.