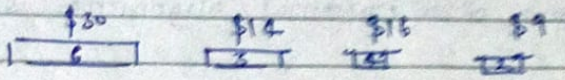
Knapsack Problem

Knapsack Problem:

goal: maximize value ($)
while limiting total weight (kg)

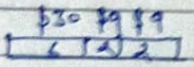
greedy { fractional knapsack: can take fraction of items

greedy does not work { discrete knapsack: each item is either taken or not
(without repetitions / with Repetition)

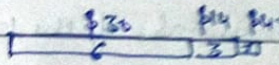
ex 

w/o repeats 

total: $46

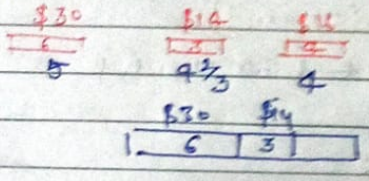
w repeats: 

total $48

fraction: 

total $105

why greedy fails for discrete

ex 

taking an element of maximum value per unit of weight is not safe

knapsack with Repetition:

Input: weight $w_1 \ldots w_n$ and values $v_1 \ldots v_n$ of n items; total weight W (v_i's, w_i's, and W are non-negative integers)

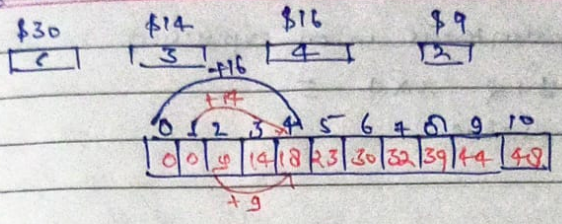
Output: The maximum value of items whose weight does not exceed W. Each item can be used any number of times

Pseudocode:

```
knapsack(w)
value(0) ← 0
for w from 1 to W:
    value(w) ← 0
    for i from 1 to n:
        if w_i < w:
            val ← value(w - w_i) + v_i
            if val > value(w)
                value(w) ← val
return value(W)
```

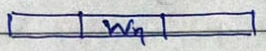
O(n·W)

eg W = 10



Knapsack without Repetation

- If the n-th term is taken into an optimal solution



then what is left is an optimal solution for a knapsack of total weight $W - w_n$ using items $1, 2 \cdots n-1$

- If the n-th item is not used, then the whole knapsack must be filled in optimally with item $1, 2 \cdots n-1$

```
Knapsack(W)

initialize all value(0, j) ← 0
initialize all value(w, 0) ← 0                          O(nW)
for i from 1 to n:                                      n iterations
    for w from 1 to W:                                  W iterations
        value(w, i) ← value(w, i-1)
        if w_i ≤ w:                                     O(1)
            val ← value(w - w_i, i-1) + v_i
            if value(w, i) < val
                value(w, i) ← val
return value(W, n)
```

Problem Overview

- How to place parenthesis in an expression

$$1+2-3\times4-5$$

to maximize it value.

Placing Parenthesis:

Input: A sequence of digit $d_1 \cdots d_n$ and a sequence of operation $op_1 \cdots op_{n-1} \in \{+, -, \times\}$

Output: An order of applying these operations that maximize the value of the expression.

$$d_1 op_1 d_2 op_2 \cdots op_{n-1} d_n.$$

toy eg

- Assume that the last operation in an optimal parenthesizing $5-0+7\times4-0+9$ is $\times$:

$$(5-0+7) \times (4-8+9).$$

- It would help to know optimal values for subexpressions $(5-0+7)$ and $4-0+9$

Example $(5-0+7) \times (4-0+9)$

$\min(5-0+7) = (5-(0+7)) = -10$

$\max(5-0+7) = ((5-0)+7) = 4$

$\min(4-0+9) = (4-(0+9)) = -13$

$\max(4-0+9) = ((4-0)+9) = 5$

$\max((5-0+7) \times (4-0+9)) = 130$

Algorithm

MinAndMax (i, j)
min ← +∞
max ← −∞
for k from i to j − 1:
    a ← M(i,k) op_k M(k+1, j)
    b ← M(i,k) op_k m(k+1, j)
    c ← m(i,k) op_k M(k+1, j)
    d ← m(i,k) op_k m(k+1, j)
    min ← min(min, a, b, c, d)
    max ← max(max, a, b, c, d)
return (min, max)

Parentheses ($d_1, op_1, d_2, op_2, \ldots, d_n$)

for i from 1 to n:
    m(i,i) ← d_i, M(i,i) ← d_i
for s from 1 to n−1:
    for i from 1 to n−s:
        j ← i+s
        m(i,j), M(i,j) ← MinAndMax(i,j)
return M(1, n)

$O(n^3)$

Reconstructing Solution: