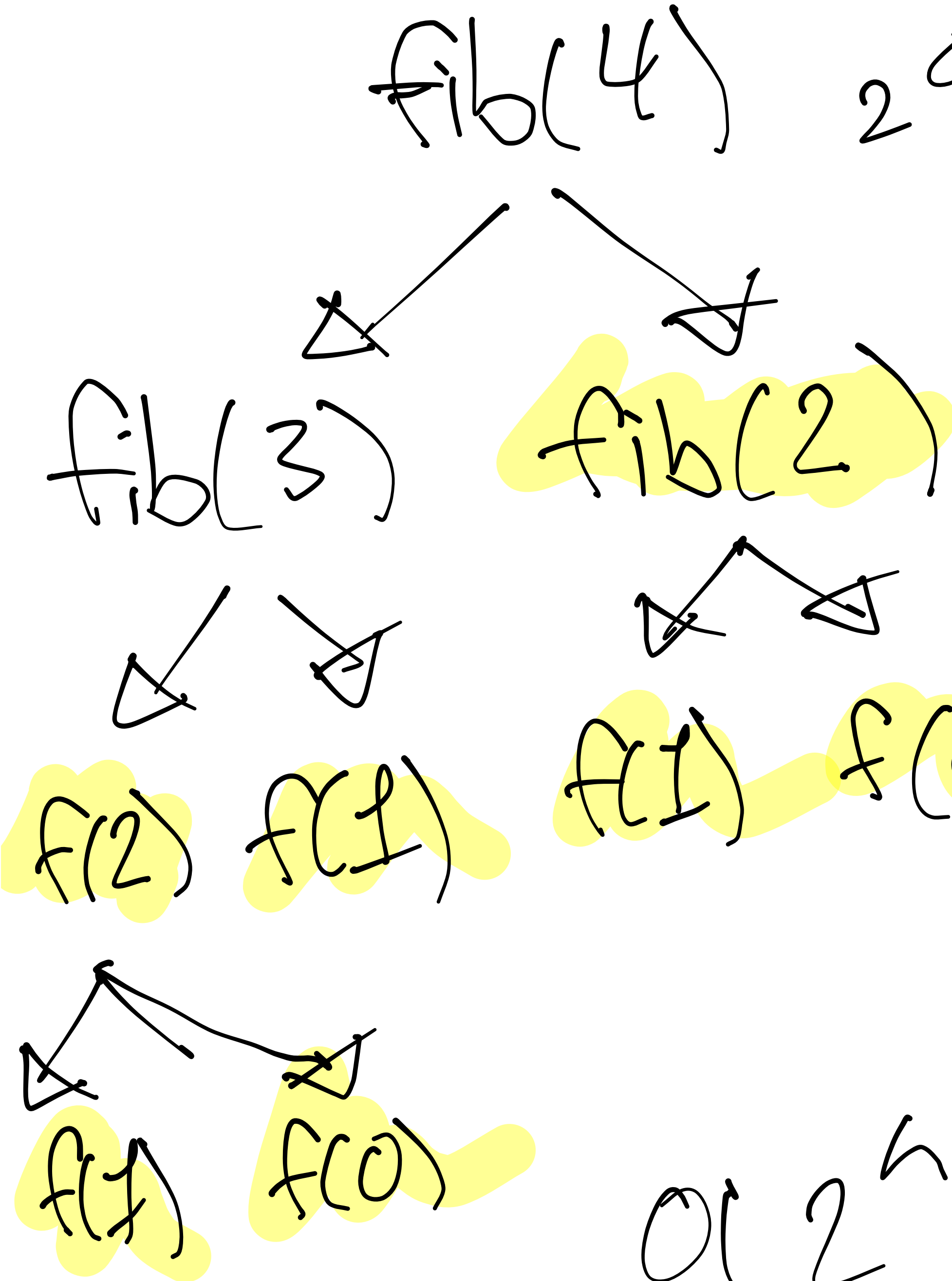
Recursion

- function that will calls itself repeatedly.
- Base case
- way to break complex problems into smaller problems.

fibonacci

$\underbrace{0+1}+\underbrace{1+1}+\underbrace{2+3}+\underbrace{5+8}\ldots\ldots$



$2^0 = 1$

$2^1 = 2$

$2^2 = 4$

$O(2^n)$

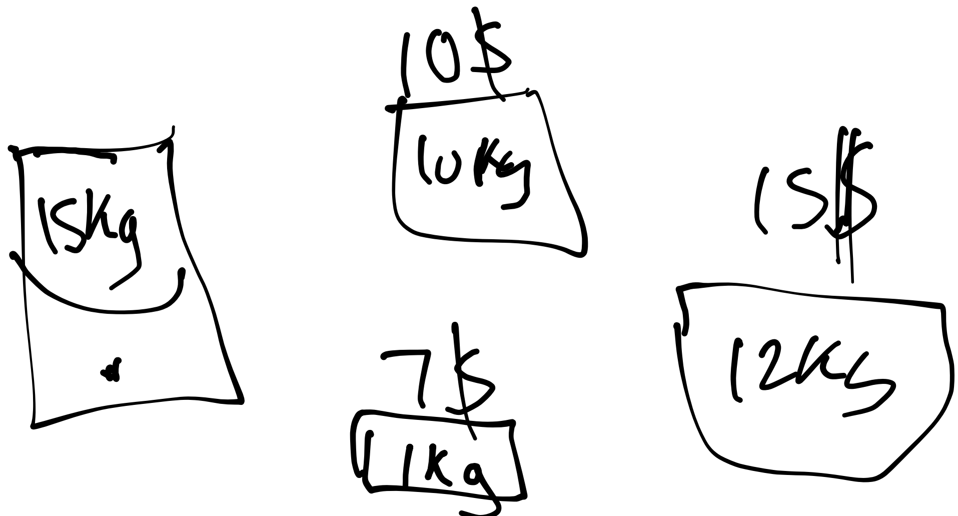
DP

* memoization (Top down)
- When we solve a subproblem we will store in our DS.
- Now when we see the same subproblem we will reference it.
~~O(1) == constant~~

* Tabulation (Bottom up)
- This more complex because requires an iterative sol'n

② Knapsack Problem

- To maximize the value to fit inside the knapsack



- given a set of N items usually from 1 - N

each of the items has mass 'w_i' and 'v_i'

0-1 Knapsack problem

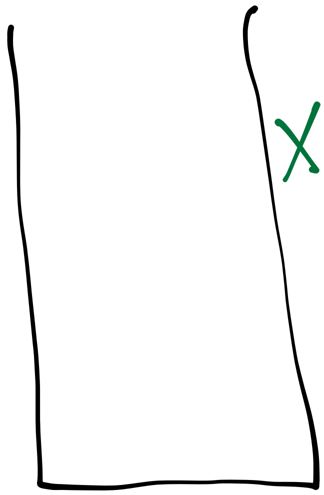
- we either take or do not take. 0 == no take, 1 = take
- Goal is to maximize the profit inside the knapsack

M = 15 Kg = capacity

Sum of 'w' items and and 'v' values must be

$\leq M$

- Figure out what to include & exclude



KS

$M = 15kg$

x_i
$$\begin{cases} w_1 = 5kg, & v_1 = 10\$ \\ w_2 = 7kg, & v_2 = 13\$ \\ w_3 = 9kg, & v_3 = 19\$ \\ w_4 = 2kg, & v_4 = 4\$ \end{cases}$$

x_i : take item 'i' or not

v_i : value of ith item

w_i : weight of ith-item

M : capacity of KS

$$\sum_{i=1}^{N} v_i * x_i \leq M$$

Formula

$$S[i][w] = \max(S[i][w], \; v_i + S[i-1][w-w_i])$$

S = DP Table

③ Knapsack problem Overview

step 1: Create DP table with rows = N+1 and columns need to be M+1. Every cell is considered a subproblem.

$$S[i][w] = \max(S[i-1][w], \; v_i + S[i-1][w-w_i])$$

```
N = 3 items      M = 5kg capacity of knapsack
item #1          w_1 = 4kg          v_1 = 10$
item #2          w_2 = 2kg          v_2 = 4$
item #3          w_3 = 3kg          v_3 = 7$
```

	0	1	2	3	4	5	w = weights [kg]
0	0	0	0	0	0	0	no of items
1	0	0	0	0	10	10	4 [item 1]
2	0	0	4	4	10	10	[item 1, item 2]
3	0	0	4	7	10	11	all items

$S[2][1] = \max(S[1][1];$ 4.1
$4\$ + S[1][1-2])$

4.2 $= \max(0, 0)$

$S[2][2] = \max(S[1][2];$
$4\$ + S[1][2-2])$

$= \max(0, 4)$ 4.3

$S[2][3] = \max(S[1][3];$

4\$ + S[1][3-2])

max(0, 4)

4.4 S[2][4] = max(S[1][4],
4\$ + S[1][4-2])

max(10, 0+4)

4.5 S[2][5] = max(S[1][5],
4\$ + S[1][5-2]

max(10, 4) = 10

* The last cell indicates the maximum profit we can make.

```
N = 3 items        M = 5kg capacity of knapsack
item #1            w_1 = 4kg            v_1 = 10$
item #2            w_2 = 2kg            v_2 = 4$
item #3            w_3 = 3kg            v_3 = 7$

     |   0   |   1   |   2   |   3   |   4   |   5   |       weights [kg]
    -------------------------------------------------
0    |       |       |       |       |       |       |       no of items
    -------------------------------------------------
1    |       |       |       |       |       |       |       [item 1]
    -------------------------------------------------
2    |       |       |       |       |       |       |       [item 1, item 2]
    -------------------------------------------------
3    |       |       |       |       |       |       |       all items
    -------------------------------------------------
```

5.1 Question:

How do we know which items to take?

- start w/ last item and compare it with item above
- If the 2 values are the same it means we have not included in the knapsack.

- otherwise we take 1 step upward and take as many steps to the left as the weight of that item.

Example 6

11 != 10 we will includ last item to KS

```
N = 3 items      M = 5kg capacity of knapsack
item #1          w_1 = 4kg          v_1 = 10$
item #2          w_2 = 2kg          v_2 = 4$
item #3          w_3 = 3kg          v_3 = 7$
```

	0	1	2	3	4	5	w = weights [kg]
0	0	0	0	0	0	0	no of items
1	0	0	0	0	10	10	4 [item 1]
2	0	0	4	4	10	10	[item 1, item 2]
3	0	0	4	7	10	11	all items

6.2

4 != 0

N = 3 items   M = 5kg capacity of knapsack
item #1 w_1 = 4kg v_1 = 10$
item #2 w_2 = 2kg v_2 = 4$
item #3 w_3 = 3kg v_3 = 7$

	0	1	2	3	4	5	w = weights [kg]
0	0	0	0	0	0	0	no of items
1	0	0	0	0	10	10	4 [item 1]
2	0	0	4	4	10	10	[item 1, item 2]
3	0	0	4	7	10	11	all items

When capacity is zero
algorithm ends

N = 3 items   M = 5kg capacity of knapsack
item #1 w_1 = 4kg v_1 = 10$
item #2 w_2 = 2kg v_2 = 4$
item #3 w_3 = 3kg v_3 = 7$

	0	1	2	3	4	5	w = weights [kg]
0	0	0	0	0	0	0	no of items
1	0	0	0	0	10	10	4 [item 1]
2	0	0	4	4	10	10	[item 1, item 2]
3	0	0	4	7	10	11	all items

total gain = 4 + 7 = \$11

weight = (2 + 3) kg = 5kg