The Knapsack Problem (0/1 version)

Description

- You have a knapsack with capacity W
- You have a set S of items. $S = \{0, 1, 2, \ldots, n\}$
- Each item i has a value and a weight, V_i and W_i
- Goal: Get as much value as possible into your knapsack without exceeding the capacity W.
- An item can either be fully taken or not taken, i.e. it's not allowed to take fractions of items.
- Goal in fancy notation:

$$\text{maximize} \sum_{i \in S} V_i \quad \text{subject to} \sum_{i \in S} W_i \leq W$$

Solution using Dynamic Programming

- We have the array `values[]` that stores all the item's values
- We have the array `weights[]` that stores all the item's weights
- We have the knapsack's capacity `W`
- We have the total number of (distinct) items `N`
- `i` means item number, V_i is i's value, W_i is i's weight &
- `W` is the current weight limit.
- And most importantly: We have a 2D-array, the memo-table `memo[i, W]` that stores the maximum value that can be attained with a weight less than or equal to `W` using items up to `i`, that is, the first `i` items.

continuation →