$0-1$ Knapsack summary

- The problem: We are given a knapsack of capacity W and a set of n items; an each item i, with $1 \leq i \leq n$, is worth $v[i]$ and has weight $w[i]$ pounds. Assume that weights $w[i]$ and the total weight W are integers. The goal is to fill the knapsack so that the value of all items in the knapsack is maximized.

- Choice of subproblem: Let $optknapsack(k, w)$ return the maximal value obtainable when filling a knapsack of capacity w using items among items 1 through k. To solve our problem we call $optknapsack(n, W)$.

- Recursive definition of $optknapsack(k, w)$:

OPTKNAPSACK(k, w)
// returns the max value to pack a knapsack ofcapacity w using items 1 through k.
if $(w == 0)$: return 0
if $(k \leq 0)$: return 0
IF $(weight[k] \leq w)$: $with = value[k] +$ OPTKNAPSACK $(k-1, w - weight[k])$
ELSE: $with = 0$
$without =$ OPTKNAPSACK$(k-1, w)$
RETURN max $\{ \, with, without \, \}$

- Correctness: tries both possibilities for an item (take or not) and recurses on the rest (which is correct bec. of optimal substructure).

- Dynamic programming solution, recursive (top-down) with memoization: We create a table $table[1..n][1..W]$, where $table[i][w]$ will store the result of optknapsack(i, w). We initialize all entries in the table as 0. To solve the problem, we call $optknapsackDP(n, W)$.

OPTKNAPSACKDP(k, w)
// global variable $table[1..n][1..W]$ initialized to 0. Also global $v[1..n]$ and $w[1..n]$.
// returns the max value to pack a knapsack of capacity w using items 1 through k.
if $(w == 0)$: return 0
if $(k \leq 0)$: return 0
IF $(table[k][w] \neq 0)$: RETURN $table[k][w]$
IF $(w[k] \leq w)$: $with = v[k] +$ OPTKNAPSACKDP$(k-1, w - w[k])$
ELSE: with = 0
$without =$ OPTKNAPSACKDP$(k-1, w)$
$table[k][w] = \max \{ \, with, without \, \}$
RETURN $table[k][w]$

Running time: $O(n \cdot W)$

- Dynamic programming, iterative (bottom-up):

OPTKNAPSACKDP_ITERATIVE()
create table[0..n][0..W] and initialize all entries to 0
for $k = 1; k \leq n; k++$
    for $w = 1; w \leq W; w++$
        with $= v[k] + table[k-1][w - w[k]]$
        without $= table[k-1][w]$
        $table[k][w] = \max \{$ with, without $\}$
RETURN $table[n][W]$

Running time: $O(n \cdot W)$

- Computing full solution:

Input: The table $table[1..n][1..W]$ as computed above, where $table[i][x]$ stores the max value to pack a knapsack of weight x using items $1..i$.
Output: the set of items corresponding to $table[n][W]$

$i = n, w = W$
while $(i > 0)$ do:

    //is the value $table[i][w]$ achieved by including item i or not?

    if $table[i][w] == v[i] + table[i-1][w - w[i]]$:

        output item i

        $w = w - w[i], i = i - 1$

    else: $i = i - 1$

Running time: $\Theta(n \cdot W)$, with no extra space