$0-1$ **Knapsack**

- 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.

- Notation and choice of subproblem: Denote by $optknapsack(k, w)$ 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)

  if $(w \leq 0)$ or $(k \leq 0)$: return 0 //basecase

  IF $(weight[k] \leq w)$: $with = value[k] + \texttt{optknapsack}(k-1, w - weight[k])$

  ELSE: $with = 0$

  $without = \texttt{optknapsack}(k-1, w)$

  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)

  if $(w \leq 0)$ or $(k \leq 0)$:: return 0

  IF $(table[k][w] \neq 0)$: RETURN $table[k][w]$

  IF $(w[k] \leq w)$: $with = v[k] + \texttt{optknapsackDP}(k-1, w - w[k])$

  ELSE: with $= 0$

  $without = \texttt{optknapsackDP}(k-1, w)$

  $table[k][w] =$ 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 $(k = 1; k < n; k + +)$

for $(w = 1; w < W; w + +)$

$with = v[k] + table[k-1][w - w[k]]$

$without = table[k-1][w]$

$table[k][w] =$ max { with, without }

RETURN $table[n][W]$

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

- Computing full solution: