Program 2.8: Dynamic Programming Maximum Knapsack

input Set X of n items, for each $x_i \in X$, values p_i, a_i, positive integer b;
output Subset $Y \subseteq X$ such that $\sum_{x_i \in Y} a_i \leq b$;
begin
  for $p := 0$ **to** $\sum_{i=1}^{n} p_i$ **do**
  begin
    $M^*(1,p) :=$ undefined;
    $S^*(1,p) := 1 + \sum_{i=1}^{n} a_i$;
  end;
  $M^*(1,0) := \emptyset$; $S^*(1,0) := 0$;
  $M^*(1,p_1) := \{x_1\}$; $S^*(1,p_1) := a_1$;
  for $k := 2$ **to** n **do**
    for $p := 0$ **to** $\sum_{i=1}^{n} p_i$ **do**
    begin
      if $(p_k \leq p)$ **and** $(M^*(k-1, p-p_k) \neq$ undefined$)$
        and $(S^*(k-1,p-p_k) + a_k \leq S^*(k-1,p))$
        and $(S^*(k-1,p-p_k) + a_k \leq b)$ **then**
      begin
        $M^*(k,p) := M^*(k-1,p-p_k) \cup \{x_k\}$;
        $S^*(k,p) := S^*(k-1,p-p_k) + a_k$
      end
      else
      begin
        $M^*(k,p) := M^*(k-1,p)$;
        $S^*(k,p) := S^*(k-1,p)$
      end
    end;
  $p^* :=$ maximum p such that $M^*(n,p) \neq$ undefined;
  return $M^*(n,p^*)$
end.

of $\{x_1, \ldots, x_k\}$ that has total profit p must either contain x_k or not, one of these two choices must be the right one.

From the above relationship, it is now possible to derive an algorithm that, for any instance of MAXIMUM KNAPSACK, computes an optimal solution: this algorithm is shown in Program 2.8.

◂ Theorem 2.17

Given an instance x of MAXIMUM KNAPSACK *with n items, Program 2.8 finds an optimal solution of x in time $O(n \sum_{i=1}^{n} p_i)$ where p_i denotes the profit of the i-th item.*

PROOF

The correctness of the algorithm is implied by the principle of optimality in the case of MAXIMUM KNAPSACK. In order to bound the running time