

(1) 入力例

入力は最初の 1 行で品物数 n を整数で与え、続く n 行で各品物 i の価値 v_i と重さ w_i を半角空白区切りで与える形式であり、価値は選択したときに得られる利得、重さは容量制約に消費される量として扱うものとし、実装では $dp[51]$ で容量 $0 \sim 50$ のみを扱い初期化も $w=0..50$ で行うため入力仕様として w_i は $0 \leq w_i \leq 50$ に制限し、 v_i は int 型の範囲の任意の整数としてよく、 n は 0 以上の整数で品物数分だけ行が続くことが前提であるが品物情報を配列に保持しないためメモリは n に依存せず、区切りは空白または改行のいずれでも `scanf` が読み取る一方で数値以外の文字列や行数不足がある場合は途中で `return` して以降を処理しない仕様であり、入力例として $n=5$ のとき「5 20 5 30 10 50 20 60 25 90 45」のように与えると容量 50 の最適化問題のデータとなる。

(2) 出力例

出力は容量 50 以下で選択した品物集合の価値総和の最大値、すなわち動的計画法配列 dp の最終結果 $dp[50]$ を 1 行に符号付き十進整数で表示するものであり、`printf("%d\n", dp[50])` により数値以外の装飾や説明文、余分な空白、複数行を付けず、行末は改行 1 回のみとし、容量値 50 自体は入力されずプログラム内で固定されている点が出力仕様の前提であり、最終的に $dp[50]$ のみを表示するため途中経過の dp 配列や選択した品物番号の列挙は行わず、標準出力に対して 1 行だけを返す。例えば入力「5 20 5 30 10 50 20 60 25 90 45」では重さ 5, 25, 20 の 3 品物を選ぶと総重量 50 で価値 130 となりこれが最大であるため「130」と出力され、もし全ての品物が重すぎて入らない場合、あるいは価値が負のみである場合でも初期値 0 が保持されるため出力は 0 となり、この挙動は空集合を選べるという一般的な定義と整合する。

(3) 入力に対する出力結果の妥当性の説明

本プログラムは $dp[w]$ を容量 w における最大価値として 0 で初期化し、各品物 (v_i, w_i) について w を 50 から w_i へ逆順に走査して $\text{cand} = dp[w - w_i] + v_i$ を計算し $dp[w]$ と比較して大きい場合のみ更新することで、同一品物を同一周回内で複数回使うことを防ぎ、品物を高々 1 回だけ選ぶ 0/1 ナップサックの要件を満たすアルゴリズムとなっており、 $w_i=0$ の場合でも逆順走査により各容量で v_i が高々 1 回だけ加算されるため仕様に反しないことが確認でき、また $w_i > 50$ を与えると更新が一切起きず無視されるため入力条件で $w_i \leq 50$ を課す必要があり、例として入力「5 20 5 30 10 50 20 60 25 90 45」では全探索の最適値 130 が得られるのに対し出力も 130 となり想定値と一致し、さらに計算量 $O(50n)$ と固定メモリ $O(51)$ で制約に適合しているため、重量制約 50 の下で最大価値を求めるという問題要件を満たしていると判断できる。