# 0/1 Knapsack Problem

## 1. Introduction

The **0/1 Knapsack Problem** is a classic **combinatorial optimization problem** in which each item can either be selected completely or not selected at all. Fractional selection of items is not allowed.

## 2. Problem Statement

Given:

- A set of $n$ items
- Each item has a weight $w_i$ and value $v_i$
- A knapsack with maximum capacity $W$

The objective is to maximize the total value of selected items such that the total weight does not exceed the knapsack capacity.

## 3. Mathematical Formulation

$$\text{Maximize } \sum_{i=1}^{n} v_i x_i$$

Subject to:

$$\sum_{i=1}^{n} w_i x_i \leq W$$

Where:

$$x_i \in \{0, 1\}$$

## 4. Why Greedy Approach Fails

Greedy methods based on value, weight, or value-to-weight ratio do not always produce optimal solutions for the 0/1 Knapsack problem. Hence, **Dynamic Programming** is used.

## 5. Dynamic Programming Approach

The problem is solved by breaking it into smaller overlapping subproblems and storing their results.

## 6. DP State Definition

Let:

$$dp[i][w]$$

represent the maximum value that can be obtained using the first $i$ items with knapsack capacity $w$.

## 7. Recurrence Relation

If $w_i \leq w$:

$$dp[i][w] = \max\left(dp[i-1][w],\ v_i + dp[i-1][w-w_i]\right)$$

If $w_i > w$:

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

## 8. Base Conditions

$$dp[0][w] = 0 \quad \forall w$$
$$dp[i][0] = 0 \quad \forall i$$

## 9. Algorithm Steps

1. Create a DP table of size $(n+1) \times (W+1)$
2. Initialize the first row and first column with zeros
3. Fill the table using the recurrence relation
4. The optimal solution is found at $dp[n][W]$

## 10. Time and Space Complexity

- Time Complexity: $O(nW)$
- Space Complexity: $O(nW)$
- Optimized Space Complexity: $O(W)$ using 1D DP

## 11. Advantages

- Guarantees optimal solution
- Systematic and reliable approach
- Suitable for moderate problem sizes

## 12. Disadvantages

- Pseudo-polynomial time complexity
- High memory usage for large $W$
- Not suitable for very large inputs

## 13. Applications

- Resource allocation
- Budget planning
- Cargo loading
- Project selection
- Memory management

## 14. Comparison with Fractional Knapsack

| Feature | 0/1 Knapsack | Fractional Knapsack |
|---|---|---|
| Item selection | Whole or none | Fraction allowed |
| Approach | Dynamic Programming | Greedy |
| Optimal solution | Yes | Yes |
| Time complexity | $O(nW)$ | $O(n \log n)$ |

## 15. Conclusion

The 0/1 Knapsack problem is an important optimization problem that is efficiently solved using dynamic programming, ensuring an optimal solution under given constraints.