01 Knapsack Problem Code:  
  
import time

def knapsack(weights, profits, capacity):

    """

    Solves the 0/1 Knapsack problem using dynamic programming.

    Args:

    weights (list): List of item weights.

    profits (list): List of item profits.

    capacity (int): Maximum weight capacity of the knapsack.

    Returns:

    int: Maximum profit that can be obtained with the given capacity.

    """

    n = len(profits)  # Number of items

    # Create a 2D array 'memo' where memo[i][j] represents the maximum profit

    # with the first i items and a knapsack capacity of j.

    memo = [[0 for \_ in range(capacity + 1)] for \_ in range(n + 1)]

    # Build the memoization table in a bottom-up manner

    for i in range(n + 1):

        for j in range(capacity + 1):

            if i == 0 or j == 0:

                memo[i][j] = 0  # No items or no capacity

            elif weights[i - 1] <= j:

                # Option 1: Include the current item

                # Option 2: Exclude the current item

                memo[i][j] = max(profits[i - 1] + memo[i - 1][j - weights[i - 1]], memo[i - 1][j])

            else:

                memo[i][j] = memo[i - 1][j]  # Item can't be included due to weight constraint

    return memo[n][capacity]

def test\_time\_complexity():

    """

    Tests the time complexity of the knapsack algorithm by measuring the execution time

    with different input sizes. Time complexity is O(n \* c), where n is the number of items

    and c is the knapsack capacity.

    """

    # Sample test cases with increasing problem size

    test\_cases = [

        ([10, 20, 30], [60, 100, 120], 50),       # Small input

        ([10, 20, 30, 40, 50], [60, 100, 120, 200, 250], 100),  # Medium input

        ([i for i in range(1, 101)], [i \* 10 for i in range(1, 101)], 500),  # Large input

    ]

    for idx, (weights, profits, capacity) in enumerate(test\_cases):

        start\_time = time.time()

        max\_profit = knapsack(weights, profits, capacity)

        end\_time = time.time()

        elapsed\_time = end\_time - start\_time

        print(f"Test Case {idx + 1}:")

        print(f"Items: {len(weights)}, Capacity: {capacity}")

        print(f"Maximum Profit: {max\_profit}")

        print(f"Execution Time: {elapsed\_time:.6f} seconds\n")

if \_\_name\_\_ == "\_\_main\_\_":

    weights = [10, 20, 30, 40]  # List of weights

    profits = [50, 120, 100, 200]  # Corresponding profits

    capacity = 60  # Maximum capacity of knapsack

    max\_profit = knapsack(weights, profits, capacity)

    print(f"The maximum profit for a knapsack with capacity {capacity} is: {max\_profit}")

    # Test time complexity

    print("\nTesting time complexity of the knapsack algorithm:")

    test\_time\_complexity()

# OUTPUT:

The maximum profit for a knapsack with capacity 60 is: 320

Testing time complexity of the knapsack algorithm:

Test Case 1:

Items: 3, Capacity: 50

Maximum Profit: 220

Execution Time: 0.000000 seconds

Test Case 2:

Items: 5, Capacity: 100

Maximum Profit: 510

Execution Time: 0.000000 seconds

Test Case 3:

Items: 100, Capacity: 500

Maximum Profit: 5000

Execution Time: 0.011280 seconds