Khemis Miliana University
Faculty of Science and Technology
Mathematics and Computer Science Department

جامعة جيلالي بونعامة– خميس مليانة
كلية العلوم والتكنولوجيا
قسم الرياضيات و الاعلام الالي

Module : Operations Research 1

Responsible: Dr. I. Ait Abderrahim

Tutorial sheet 1

Problem: Thief optimization

A thief robs a jewellery shop. With a backpack of fixed capacity he attempts to rob the valuables. Each item he can take has a profit and a weight. When filling his backpack he must respect its total capacity (i.e., the sum of the item sizes should be less than the capacity). His goal is to maximize the total profit of the items he steals but he cannot carry too much weight.

Task: Solve the problem using python programming language.

Solution

```
from itertools import combinations

def ThiefProb_bruteforce(values, weights, capacity):
    n = len(values)

    # Generate all possible combinations of items
    all_combinations = []
    for r in range(1, n + 1):
        all_combinations.extend(combinations(range(n), r))

    # Initialize variables to store the best solution
    best_value = 0
    best_selection = []

    # Iterate through all combinations and find the best solution
    for comb in all_combinations:
        total_value = sum(values[i] for i in comb)
        total_weight = sum(weights[i] for i in comb)

        if total_weight <= capacity and total_value > best_value:
            best_value = total_value
            best_selection = list(comb)

    return best_value, best_selection

# Example usage
values = [60, 100, 120]
weights = [10, 20, 30]
capacity = 50

optimal_value, selected_items = ThiefProb _bruteforce(values, weights, capacity)
print("Optimal Value:", optimal_value)
print("Selected Items:", [values[i] for i in selected_items])
```

Correct answer: