UNIVERSITY INSTITUTE of COMPUTING
Asia's Fastest Growing University

NAAC GRADE A+ ACCREDITED UNIVERSITY

Worksheet 6

Student Name: Ramesh Kumar **UID:**25MCI10304
Branch: MCA (AI & ML) **Section/Group:** 25MAM-5A
Semester: 1st **Date of Performance:**27-10-2025
Subject Name: Design and analysis of algorithms **Subject Code:** 25CAP-612

1. **Aim/Overview of the practical:**
   To design and implement an algorithm to find a subset of a given set S = {5, 7, 9, 11, 13} of positive integers whose sum is equal to a given positive integer d = 20 using the Backtracking technique.
2. **Task to be done:**
   - Start the program.
   - Input a set of positive integers and a target sum d.
   - Create a recursive function to generate all possible subsets of the given set.
   - Add elements one by one to the current subset.
   - Calculate the sum of the subset elements at each recursive step.
   - If the sum equals the target value d, display that subset.
   - If no subset equals the desired sum after checking all combinations, display "No subset found with the given sum."

3.Algorithm:

Step 1: Start.
Step 2: Initialize the given set $S = \{5,7,9,11,13\}$and the target sum $d = 20$.
Step 3: Define a recursive function subset_sum(S, target, current, index) that:
Step 4: Include the current element S[index] in the subset and call recursively with target - S[index].
Step 5: Exclude the current element and call recursively with the same target.
Step 6: Repeat Steps 4 and 5 until all elements are checked.
Step 7: If no valid subset found, print "No subset found with the given sum."
Step 8: Stop.

4.Code:

```
def find_subsets(S, target):
    n = len(S)
    result = []

    def backtrack(start, path, current_sum):
        if current_sum == target:
            result.append(path[:])
            return
        if current_sum > target:
            return
        for i in range(start, n):
            path.append(S[i])
            backtrack(i + 1, path, current_sum + S[i])
            path.pop()

    backtrack(0, [], 0)
    return result

# Input Data
S = [5, 7, 9, 11, 13]
d = 20

# Find all subsets
solutions = find_subsets(S, d)

# Display Results
if solutions:
    print(f"\nSubsets of {S} with sum {d}:")
    for subset in solutions:
        print(subset)
else:
    print("\nNo subset found with the given sum.")
```

5.Output:

```
PS R:\Desktop\DAA> & C:/Python313/python.exe r:/De
sktop/DAA/worksheet6.py

Subsets of [5, 7, 9, 11, 13] with sum 20:
[7, 13]
[9, 11]
PS R:\Desktop\DAA>
```

6.Learning Outcomes: -

- Understood the working of Backtracking in solving recursive problems.
- Learned to generate all possible subsets of a given set using recursion.
- Gained experience in using base cases to stop recursion efficiently.
- Developed logic for checking sum conditions dynamically.
- Learned how to display multiple possible solutions for a problem.
- Understood how to handle no-solution cases gracefully in programs.
- Strengthened knowledge of recursive problem-solving and combinatorial logic in Python.