UNIVERSITY INSTITUTE of COMPUTING
Asia's Fastest Growing University

Worksheet 6

Student Name: Rahul Saxena **UID:** 24MCI10204

Branch: MCA(AI&ML) **Section/Group:** 3-B

Semester: 1st semester **Date of Performance:** 14/10/2024

Subject Name: Design and analysis of Algorithm Lab **Subject Code:** 24CAP-612

AIM:

Find a subset of a given set S={sl,s2,....sn} of n positive integers whose sum is equal to a given positive integer d. For example, if S={1,2,5,6,8}and d =9 there are two solutions{1,2,6}and{1,8}.A suitable message is to be displayed if the given problem instance doesn't have a solution.

Task To be Done:

- **Understand the Problem Requirements:**
  - Analyse the subset sum problem, which involves finding subsets of a given set that sum up to a target integer.
  - Determine and handle cases where no subset matches the desired sum by displaying an appropriate message.
- **Implement the Solution in Java:**
  - Create a Java program that recursively finds all subsets of a set SSS that sum to a given integer d.
  - Use recursion to explore both choices (including and excluding elements) for each element in the set.
  - Print each subset that sums to the target value.
  - Include base cases to handle when the subset sum matches the target, the target becomes negative, or the subset is empty.

Source Code:

```
import java.util.ArrayList;
import java.util.List;
public class SubsetSum {
   public static void findSubsets(int[] S, int index, int target, List<Integer> currentSubset) {
      if (target == 0) {
         System.out.println("Subset found: " + currentSubset);
         return;
      }
      if (target < 0 || index == S.length) {
         return;
      }
      currentSubset.add(S[index]);
      findSubsets(S, index + 1, target - S[index], currentSubset);
      currentSubset.removeLast();
```

```
        findSubsets(S, index + 1, target, currentSubset);
    }
    public static void findSubsetsThatSumToTarget(int[] S, int target) {
        List<Integer> currentSubset = new ArrayList<>();
        findSubsets(S, 0, target, currentSubset);

        if (currentSubset.isEmpty()) {
            System.out.println("No subset found that sums up to " + target);
        }
    }
    public static void main(String[] args) {
        int[] S = {1, 2, 5, 6, 8};
        int target = 9;
        System.out.println("Finding subsets in the set " + java.util.Arrays.toString(S) + " that sum up to " + target +
":");
        findSubsetsThatSumToTarget(S, target);
    }
}
```

Output:

```
"C:\Program Files\Java\jdk-22\bin\java.exe" "-javaagent:C:\Program
Finding subsets in the set [1, 2, 5, 6, 8] that sum up to 9:
Subset found: [1, 2, 6]
Subset found: [1, 8]
No subset found that sums up to 9

Process finished with exit code 0
```

Learning Outcome:

- **Understanding Recursion**:
  - Gain a deeper understanding of recursive programming techniques, especially for problems that involve exploring multiple choices and paths.
  - Learn to apply recursion for subset generation and backtracking, which is useful in many algorithmic problems.
- **Subset Sum Problem Solving**:
  - Develop problem-solving skills for the subset sum problem, a classic problem in computer science, and recognize its applications in other fields like combinatorial optimization.
  - Learn to implement and understand the logic behind exploring all subsets of a set and identifying specific conditions for successful subset matches.