Worksheet-7

Student Name: Arpit Anand
UID: 23BCS12710
Branch: BE - CSE
Section/Group: KGR-3(A)
Semester: 5
Subject Name: DAA

Date of Performance: 23/09/2025
Subject Code: 23CSH-301

1. Aim:
Develop a program and analyze complexity to implement subset-sum problem using Dynamic Programming.

2. Objective:
To implement and analyze the Subset Sum problem using Dynamic Programming for efficient solution and complexity evaluation.

3. Requirements (Hardware/Software):
Online Java compiler.

4. Algorithm :
1. Input set of numbers and target sum.
2. Create a DP table dp[n+1][sum+1].
3. Initialize dp[i][0] = true for all i, and dp[0][j] = false for j > 0.
4. For each element i from 1 to n:
   For each sum j from 1 to target:
   a) If arr[i-1] > j, set dp[i][j] = dp[i-1][j].
   b) Else set dp[i][j] = dp[i-1][j] OR dp[i-1][j-arr[i-1]].
5. Output dp[n][sum] as the result.
6. End.

5.Procedure:

```
import java.util.*;
class SubsetSumDP {
    static boolean subsetSum(int[] arr, int n, int sum) {
        boolean dp[][] = new boolean[n + 1][sum + 1];
        for (int i = 0; i <= n; i++) dp[i][0] = true;
        for (int i = 1; i <= n; i++) {
            for (int j = 1; j <= sum; j++) {
                if (arr[i - 1] <= j)
                    dp[i][j] = dp[i - 1][j] || dp[i - 1][j - arr[i - 1]];
                else
                    dp[i][j] = dp[i - 1][j];
            }
        }
        return dp[n][sum];
    }

    public static void main(String[] args) {
        int[] arr = {3, 34, 4, 12, 5, 2};
        int sum = 9;
        System.out.println(subsetSum(arr, arr.length, sum) ? "Subset exists"
: "No subset found");
    }
}
```

Time Complexity : Best Case: O(n*sum)

Space complexity : O(n*sum)

Output:

Learning Outcomes :

1. Learn to apply Dynamic Programming for solving the Subset Sum problem.
2. Analyze the time and space complexity of the DP approach
3. Understand optimization techniques to improve space efficiency.