



Experiment-6

Student Name: Tejinderpal Singh

Branch: BE-CSE

Semester: 5th

Subject Name: DAA

UID: 23BCS80343

Section/Group: 23BCS_KRG_3A

Subject Code: 23CSH-301

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

2. Objective: To implement the Subset Sum Problem using Dynamic Programming in C++ and understand how dynamic programming efficiently solves problems involving subsets and target sums.

3. Procedure:

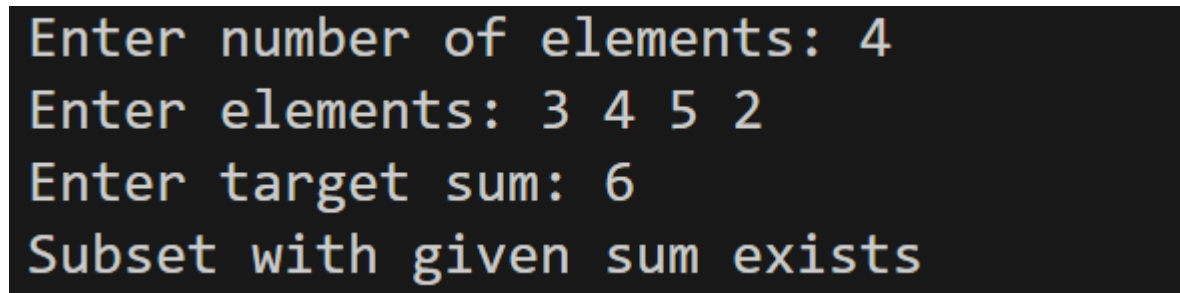
1. Start
2. Input the number of elements n and the array $arr[]$.
3. Input the target sum sum .
4. Create a 2D boolean DP table $dp[n+1][sum+1]$.
5. Initialize $dp[i][0] = \text{true}$ for all i and $dp[0][j] = \text{false}$ for all $j > 0$.
6. For each element i from 1 to n :
 For each sum j from 1 to sum :
 If $arr[i-1] > j$, then $dp[i][j] = dp[i-1][j]$.
 Else $dp[i][j] = dp[i-1][j] \parallel dp[i-1][j - arr[i-1]]$.
7. The final answer is $dp[n][sum]$.
8. If true, print subset exists; otherwise, it does not.

4. Code:

```
#include <iostream>
#include <vector>
using namespace std;
int main() {
    int n, sum;
    cout << "Enter number of elements: ";
    cin >> n;
    vector<int> arr(n);
    cout << "Enter elements: ";
    for (int i = 0; i < n; i++) cin >> arr[i];
    cout << "Enter target sum: ";
```

```
cin >> sum;
vector<vector<bool>> dp(n + 1, vector<bool>(sum + 1, false));
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];
        else dp[i][j] = dp[i - 1][j] || dp[i - 1][j - arr[i - 1]];
    }
}
if (dp[n][sum]) cout << "Subset with given sum exists";
else cout << "Subset with given sum does not exist";
return 0;
}
```

5. Observations:



```
Enter number of elements: 4
Enter elements: 3 4 5 2
Enter target sum: 6
Subset with given sum exists
```

6. Time Complexity:

- $O(n \times \text{sum})$ in every case

7. Learning Outcome:

- ❖ Learned how to implement the Subset Sum Problem using Dynamic Programming in C++.
- ❖ Understood the concept of overlapping subproblems and optimal substructure in DP.
- ❖ Gained practical experience in constructing and filling a DP table for subset evaluation.
- ❖ Strengthened understanding of problem-solving using recursive and iterative DP approaches.
- ❖ Learned to analyze the time and space complexity of dynamic programming solutions.