



**DEPARTMENT OF**

**COMPUTER SCIENCE & ENGINEERING**

Discover. Learn. Empower.

## **Experiment 2.2**

**Student Name: UTKARSH JOSHI**

**Branch: CSE**

**Semester: 5<sup>th</sup>**

**Subject Name: DAA LAB**

**UID: 21BCS9158**

**Section/Group: ST\_802A**

**Date of Performance: 28/09/2023**

**Subject Code: 21CSP-311**

### **1. Aim/Overview of the practical:**

To implement subset-sum problem using Dynamic Programming.

### **2. Task to be done/which logistics used:**

To implement subset sum problem using dynamic programming and problemsolving skills with the designing of the algorithm.

### **3. Algorithm/ Flowchart:**

- A naive solution would be to cycle through all subsets of  $n$  numbers and, for every one of them, check if the subset sums to the right number. The running time is of order  $O(2^n \cdot n)$
- Since there are  $2^n$  subsets, and to check each subset, we need to sum at most  $n$  elements.
- A better exponential-time algorithm uses recursion. Subset sum can also be thought of as a special case of the 0–1.
- Knapsack problem. For each item, there are two possibilities:
  - a. Include the current item in the subset and recur for the remaining items with the remaining total.
  - b. Exclude the current item from the subset and recur for the remaining items.

- Finally, return true if we get a subset by including or excluding the current item; otherwise, return false.
- The recursion's base case would be when no items are left, or the sum becomes negative. Return true when the sum becomes 0, i.e., the subset is found.

#### 4. Steps for experiment/practical/Code:

```
//SUBSET SUM
#include<bits/stdc++.h>
using namespace std;
bool sumSet(int* arr,int sum,int n)
{ if
(sum==0)
{
return true;
}
if (n==0 && sum!=0)
{
return false;
}

{
return sumSet(arr,sum,n-1);
}

//Reverse Checking
//ACCORDING TO THE TOP ELEMENT IF TRUE THEN AS IT IS, IF
FALSE THEN REVEAL TILL THE Nth ELEMENT.

return sumSet(arr,sum,n-1)||sumSet(arr,sum-arr[n-1],n-1);

}

int main ()
```



**DEPARTMENT OF**

**COMPUTER SCIENCE & ENGINEERING**

Discover. Learn. Empower.

```
{
```

```
int size;
```

```
cout<<"Enter The Size Of The Array="; cin>>size;
```

```
int arr[size]; cout<<"Enter The Elements In  
The Array=";
```

```
for (int i=0;i<size;i++)  
{ cin>>arr[i];  
} int sum;
```

```
cout<<"Enter Targeted Sum="; cin>>sum;
```

```
if (sumSet(arr,sum,size)==true)  
{
```

```
cout<<"Sum Found."<<endl;
```

```
} else  
{
```

```
cout<<"Sum Not Found."<<endl;
```

```
}
```

```
return 0;
```

```
getchar();
```

}

## 5. Output:

```
C:\Users\win10\OneDrive\Desktop\C++\sum.exe
Enter The Size Of The Array=5
Enter The Elements In The Array=1 2 3 4 5
Enter Targeted Sum=23
Sum Not Found.

-----
Process exited after 6.947 seconds with return value 0
Press any key to continue . . .
```

## 6. Observations/Discussions/ Complexity Analysis:

- $O(N \cdot \text{sum})$  where,  $N$  is the size of the array.
- Space Complexity:  $O(N \cdot \text{sum})$ .

## 7. Learning Outcomes:

- a) Learnt about dynamic programming.
- b) Learnt about design and algorithm analysis.
- c) Learnt about calculating time complexity of the subset sum problem using dynamic programming.
- d) Learnt about recursion.
- e) Learnt about initiating the base case of the condition.