**Bharatiya Vidya Bhavan's**
**Sardar Patel Institute of Technology**
Bhavan's Campus, Munshi Nagar, Andheri (West), Mumbai-400058-India
(Autonomous College Affiliated to University of Mumbai)

**Computer Engineering Department &**
**Information Technology Engineering Department**

**Academic Year:** 2021-2022

**Class:** S.Y.B.Tech **Sem.:** 4 **Course:** DAA

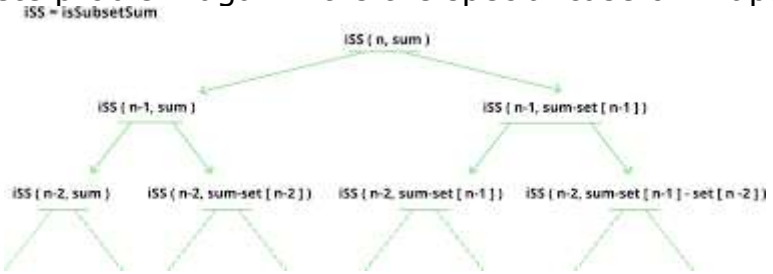
| **Name** | **Pratik Pujari** | | |
|---|---|---|---|
| **UID no.** | **2020300054** | **Class:** | **Comps C Batch** |
| **Experiment No.** | 7 | | |

| **AIM:** | To implement the concept of backtracking in subset sum problem |
|---|---|
| **THEORY:** | **What is Backtracking?**<br>Backtracking is an algorithmic-technique for solving problems recursively by trying to build a solution incrementally, one piece at a time, removing those solutions that fail to satisfy the constraints of the problem at any point of time (by time, here, is referred to the time elapsed till reaching any level of the search tree).<br><br>There are three types of problems in backtracking –<br>1. Decision Problem – In this, we search for a feasible solution.<br>2. Optimization Problem – In this, we search for the best solution.<br>3. Enumeration Problem – In this, we find all feasible solutions.<br><br>**When to Use a Backtracking Algorithm**<br>The backtracking algorithm is applied to some specific types of problems. For instance, we can use it to find a feasible solution to a decision problem. It was also found to be very effective for optimization problems.<br><br>For some cases, a backtracking algorithm is used for the enumeration problem in order to find the set of all feasible solutions for the problem.<br><br>On the other hand, backtracking is not considered an optimized technique to solve a problem. It finds its |

application when the solution needed for a problem is not time-bounded.

# Subset Sum Problem

It is one of the most important problems in complexity theory. The problem is given an A set of integers a1, a2,…., an upto n integers. The question arises that is there a non-empty subset such that the sum of the subset is given as M integer?. For example, the set is given as [5, 2, 1, 3, 9], and the sum of the subset is 9; the answer is YES as the sum of the subset [5, 3, 1] is equal to 9. This is an NP-complete problem again. It is the special case of knapsack




**Example**

Consider the following array/ list of integers:
{1, 3, 2}
We want to find if there is a subset with sum 3.
Note that there are two such subsets **{1, 2}** and **{3}**. We will follow our backtracking approach.
Consider our empty set **{}**
We add 1 to it **{1}** (sum = 1, 1 < 3)
We add 2 to it **{1, 3}** (sum = 3, 3 == 3, found)
We remove 3 from it **{1}** (sum = 1, 1 < 3)
We add 2 to it **{1, 2}** (sum = 3, 3 == 3, found)
We remove 2 and see that all elements have been considered.
Following diagram captures the idea:

**{} -> {1} -> {1, 3} (found) -> {1} -> {1, 2} (found) -> {1} (end)**

Bharatiya Vidya Bhavan's
**Sardar Patel Institute of Technology**
Bhavan's Campus, Munshi Nagar, Andheri (West), Mumbai-400058-India
(Autonomous College Affiliated to University of Mumbai)

**Computer Engineering Department &**
**Information Technology Engineering Department**

**Academic Year:** 2021-2022

**Class:** S.Y.B.Tech **Sem.:** 4 **Course:** DAA

| **PSEUDOCODE:** | subset_sum()<br>if(subset is satisfying the constraint)<br>print the subset<br>exclude the current element and consider next element<br>else<br>generate the nodes of present level along breadth of tree and<br>recur for next levels |
|---|---|
| **EXPERIMENT 1** | |
| **CODE:** | #include <stdio.h><br>#include <stdlib.h><br>static int total_nodes;<br>void printValues(int A[], int size){<br>//prints the array<br>printf("\n\n");<br>for (int i = 0; i < size; i++) {<br>printf("%*d", 5, A[i]);<br>}<br>}<br>void subset_sum(int s[], int t[], int s_size,<br>int t_size, int sum, int ite, int const target_sum){<br>//increments the total node count<br>total_nodes++;<br>if (target_sum == sum) {<br>// target sum found and printing the list.<br>printValues(t, t_size);<br>//finding all other valid pairs<br>subset_sum(s, t, s_size, t_size - 1, sum<br>- s[ite], ite + 1, target_sum);<br>return;<br>}<br>else {<br>//checking for all possible combinations<br>for (int i = ite; i < s_size; i++) {<br>t[t_size] = s[i]; |

**Computer Engineering Department &**
**Information Technology Engineering Department**

**Academic Year:** 2021-2022

**Class:** S.Y.B.Tech **Sem.:** 4 **Course:** DAA

```
        subset_sum(s, t, s_size, t_size + 1, sum
+ s[i], i + 1, target_sum);
      }
   }
}
void generateSubsets(int s[], int size, int target_sum){
//generating subsets of the initial array
   int* tuplet_vector = (int*)malloc(size * sizeof(int));
   subset_sum(s, tuplet_vector, size, 0, 0, 0, target_sum);
   free(tuplet_vector);
}
int main(){
   int size;
   int target_sum;
//user input like array sze and array and target sum
   printf("\n-------------------------------");
   printf("\nEnter the size of the set: ");
   scanf("%d", &size);
   int set[size];
   printf("\n-------------------------------");
   printf("\nEnter the elements of the set: ");
   for (int i = 0; i < size; i++) {
      scanf("%d", &set[i]);
   }
   printf("The set is ");
   printValues(set , size);
   printf("\n-------------------------------");
   printf("\nEnter the target sum: ");
   scanf("%d", &target_sum);

//calling the functions and getting the result
   generateSubsets(set, size, target_sum);
//printing the total nodes.
   printf("\n\nTotal Nodes generated %d\n", total_nodes);
   return 0;
}
```

**Bharatiya Vidya Bhavan's**
# Sardar Patel Institute of Technology
Bhavan's Campus, Munshi Nagar, Andheri (West), Mumbai-400058-India
(Autonomous College Affiliated to University of Mumbai)

## Computer Engineering Department & Information Technology Engineering Department

**Academic Year:** 2021-2022

**Class:** S.Y.B.Tech **Sem.:** 4 **Course:** DAA

**OUTPUT:**

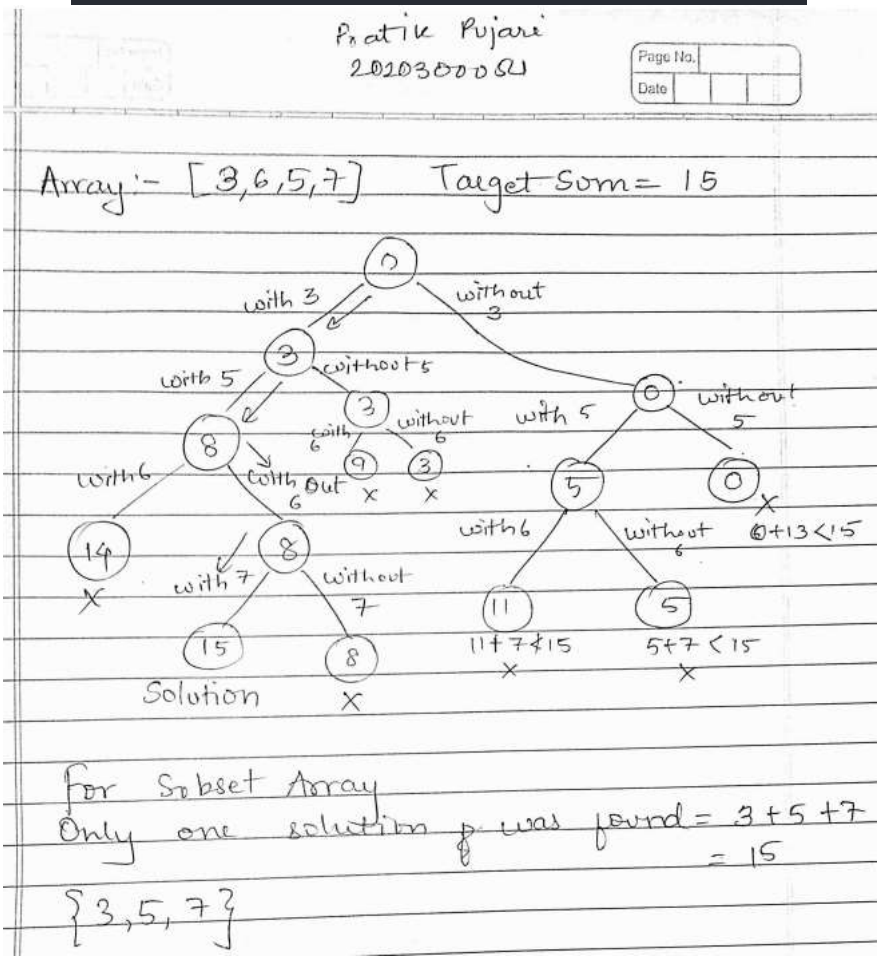
```
--------------------------------
Enter the size of the set: 6

--------------------------------
Enter the elements of the set: 12 1 61 5 9 2
The set is

   12    1   61    5    9    2
--------------------------------
Enter the target sum: 24


   12    1    9    2

Total Nodes generated 65
```



**Bharatiya Vidya Bhavan's**
# Sardar Patel Institute of Technology
Bhavan's Campus, Munshi Nagar, Andheri (West), Mumbai-400058-India
(Autonomous College Affiliated to University of Mumbai)

## Computer Engineering Department & Information Technology Engineering Department

**Academic Year:** 2021-2022

**Class:** S.Y.B.Tech **Sem.:** 4 **Course:** DAA

| | |
|---|---|
| **TIME COMPLEXITY:** | The time complexity of the sum of subset problem is O(2^n) using backtracking approach ,where n is the number of elements in the array<br><br>Since there will be two branches of each node and every element is atleast traversed once making it a binary tree.<br><br>The total number of nodes can be calculated by the formula 2^n-1 where is the number of elements in the array. |

**CONCLUSION:** Things learnt during the procedural solving of the program.

- Learnt how to use backtracking for solving sums that use a binary tree structure.
- Learnt how to analyse the time complexity to find the number of nodes in diagram
- Learnt to implement backtracking solution to the subset problem.
- Learnt how to solve subset problem using the backtracking method using the tree way.