K. J. Somaiya College of Engineering, Mumbai-77
(A Constituent College of Somaiya Vidyavihar University)
Department of Computer Engineering

Batch: C2	**Roll No.: 16010122257**
Experiment No. __8____	
Grade: AA / AB / BB / BC / CC / CD /DD	
Signature of the Staff In-charge with date	

Title: Implementation of sum of subset Algorithm

Objective: To learn the Backtracking strategy of problem solving for Sum of subset

CO to be achieved:

CO 2 Analyze and solve problems for divide and conquer strategy, greedy method, dynamic programming approach and backtracking and branch & bound policies.

Books/ Journals/ Websites referred:

1. **Ellis horowitz, Sarataj Sahni, S.Rajsekaran," Fundamentals of computer algorithm", University Press**
2. **T.H.Cormen ,C.E.Leiserson,R.L.Rivest and C.Stein," Introduction to algortihtms",2nd Edition ,MIT press/McGraw Hill,2001**

Pre Lab/ Prior Concepts:
Data structures, Concepts of algorithm analysis

Historical Profile:
Subset sum problem is to find subset of elements that are selected from a given set whose sum adds up to a given number K. We are considering the set contains non-negative values. It is assumed that the input set is unique (no duplicates are presented).
One way to find subsets that sum to K is to consider all possible subsets. A power set contains all those subsets generated from a given set. The size of such a power set is 2N.

Input:

A vector X={x1,x2… xn} for all n elements in the set where Xi=0 (element not added) or xi=1 (element added in the solution tuple).

Output:
Summation of the chosen numbers must be equal to given number M and one number can be used only once.

BACKTRACKING CONDITION

$$B_k(x_1, \ldots, x_k) = true \text{ iff } \sum_{i=1}^{k} w_i x_i + \sum_{i=k+1}^{n} w_i \geq m$$

$$\text{and } \sum_{i=1}^{k} w_i x_i + w_{k+1} \leq m$$

New Concepts to be learned:
Application of algorithmic design strategy to any problem, Backtracking method of problem solving Vs other methods of problem solving problem sum of subset and its applications.

Algorithm:

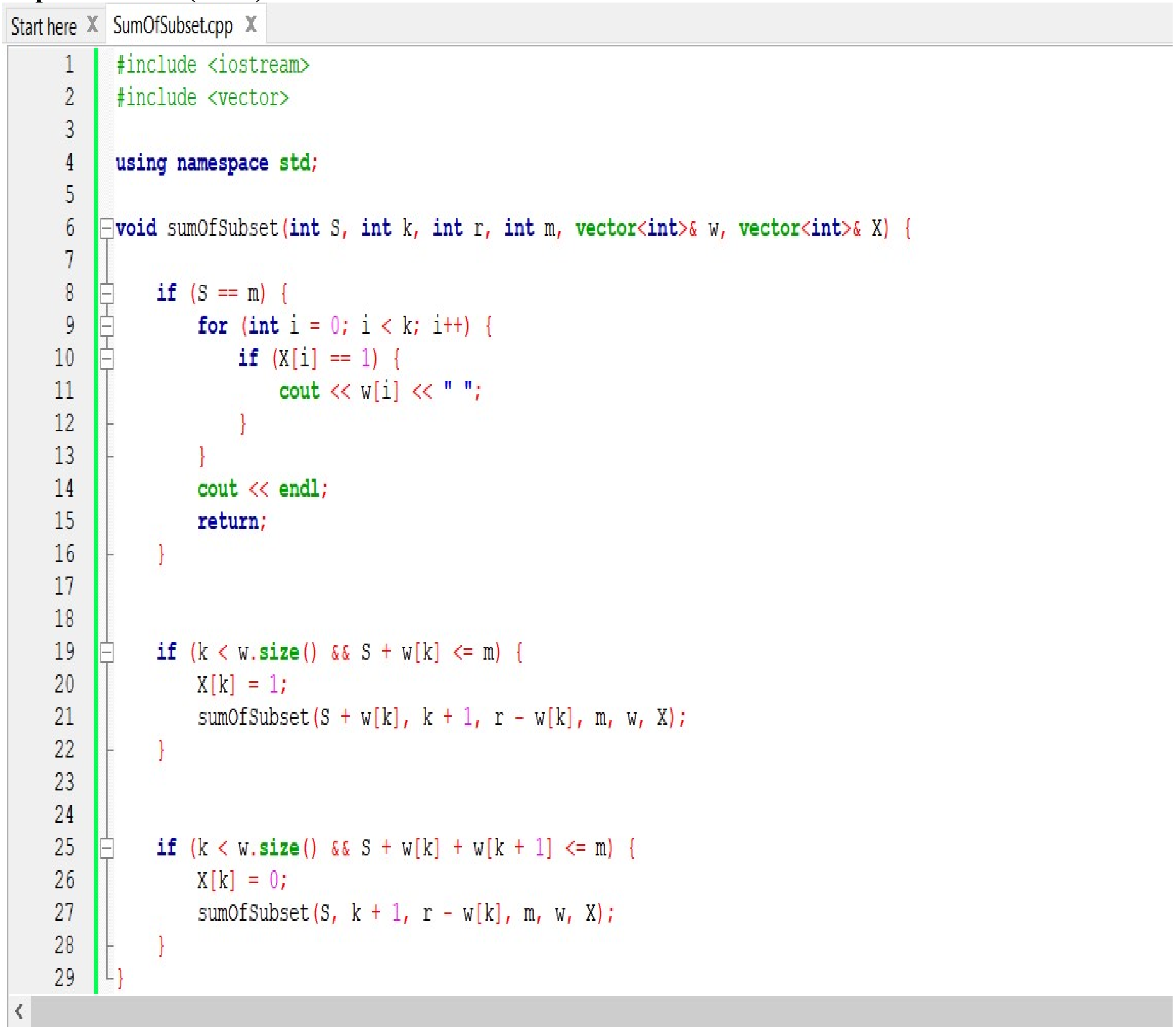
```
Algorithm sumOfSub(s, k, r)

{//It is assumed w[1]<=m and Sigma(i=1 to m)w[i]>=m

//generate the left child. Note: s+w(k)<=M since Bk-1 is true.


X{k]=1;


if (S+W[k]=m) then write(X[1:k]); //Subset found. there is no recursive call here
as W[j]>0,1<=j<=n.

else if (S+W[k]+W[k+1]<=m) then sumOfSub(S+W[k], k+1,r- W[k]); //moving to
next sub-problem.
```

Similarly, assume the array is presorted and we found one subset. We can generate next node excluding the present node only when inclusion of next node satisfies the constraints.

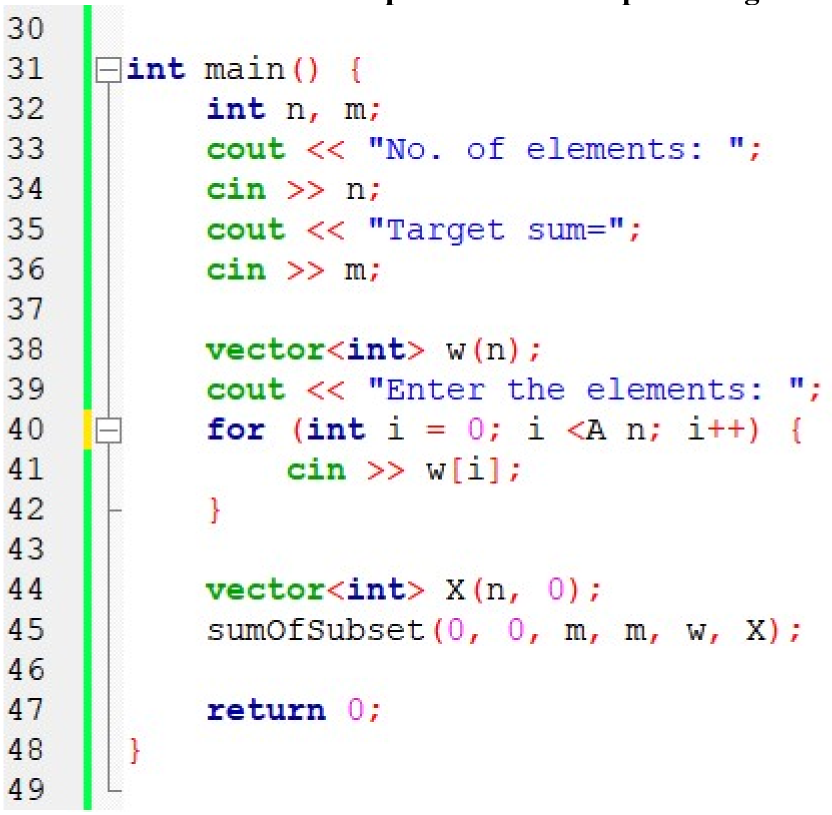
```
if ((S+ r- W[k]>=m)and (S+ W[k+1]<=m)) then//generate right {
```

```
//child and those satisfying 2 bounding functions

X{k]=0;

sumOfSub (S, k+1, r- W[k]);
}}
```

Implementation(Code):

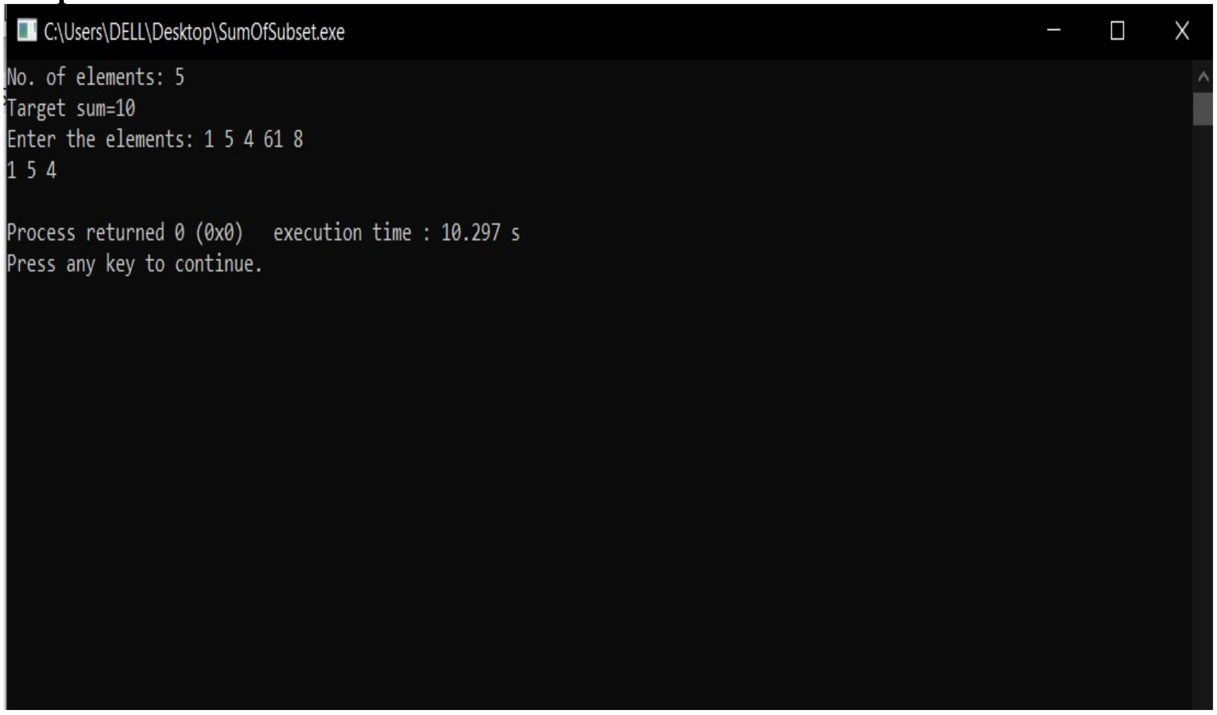


```cpp
#include <iostream>
#include <vector>

using namespace std;

void sumOfSubset(int S, int k, int r, int m, vector<int>& w, vector<int>& X) {

    if (S == m) {
        for (int i = 0; i < k; i++) {
            if (X[i] == 1) {
                cout << w[i] << " ";
            }
        }
        cout << endl;
        return;
    }


    if (k < w.size() && S + w[k] <= m) {
        X[k] = 1;
        sumOfSubset(S + w[k], k + 1, r - w[k], m, w, X);
    }


    if (k < w.size() && S + w[k] + w[k + 1] <= m) {
        X[k] = 0;
        sumOfSubset(S, k + 1, r - w[k], m, w, X);
    }
}
```



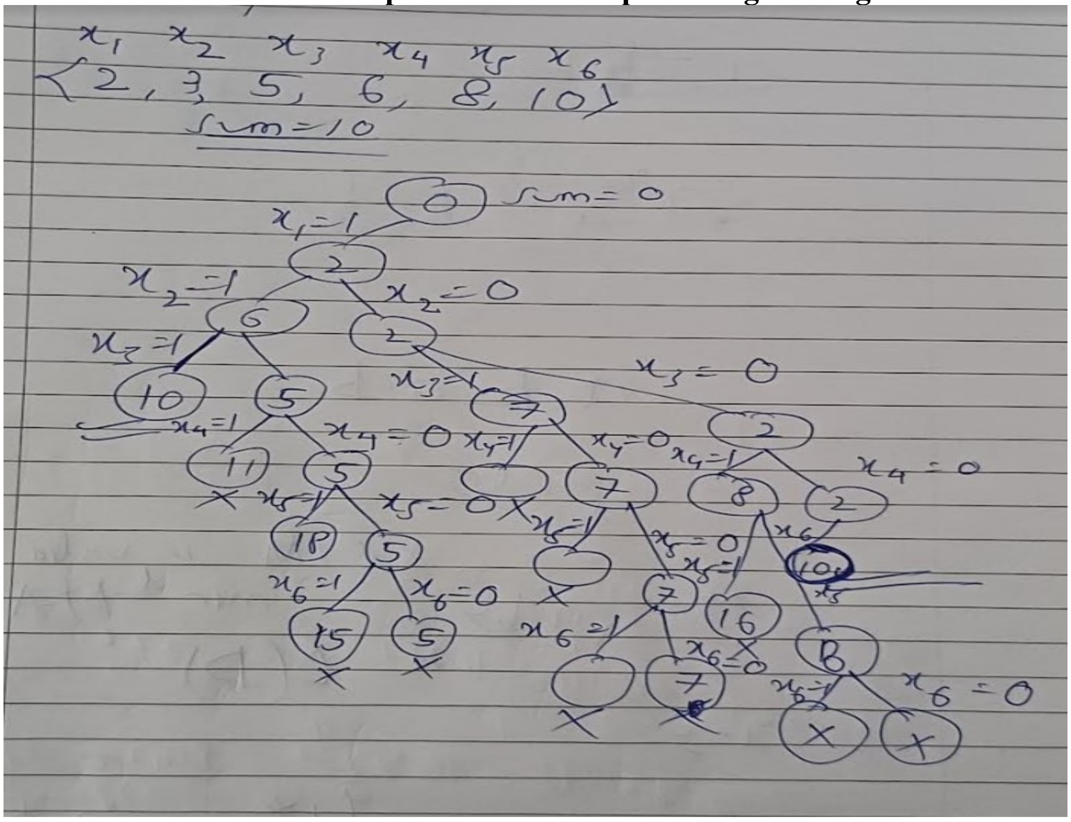
```
int main() {
    int n, m;
    cout << "No. of elements: ";
    cin >> n;
    cout << "Target sum=";
    cin >> m;

    vector<int> w(n);
    cout << "Enter the elements: ";
    for (int i = 0; i <A n; i++) {
        cin >> w[i];
    }

    vector<int> X(n, 0);
    sumOfSubset(0, 0, m, m, w, X);

    return 0;
}
```

Output:



```
No. of elements: 5
Target sum=10
Enter the elements: 1 5 4 61 8
1 5 4

Process returned 0 (0x0)   execution time : 10.297 s
Press any key to continue.
```

Example sum of subset Problem along with state space tree:



Analysis of Backtracking solution for sum of subset Problem:

Time Complexity= O(2^n), where n is the number of elements in the set. This is because, in the worst-case scenario, the algorithm may need to explore all possible subsets of the given set. Each element can either be included or excluded in a subset, leading to 2^n possible combinations.
Space Complexity= O(n), where n is the number of elements in the set. This is because the algorithm uses a recursive call stack to keep track of the current subset being explored, and the maximum depth of this stack is equal to the number of elements in the set.
Practical Considerations
Its exponential time complexity makes it impractical for large sets.

Conclusion: Thus,from this experiment we learnt about sum of subset algorithm and implemented it using backtracking analysis.It finds analysis in fields like finance,resource allocation and scheduling,etc.