K. J. Somaiya College of Engineering
(A Constituent College of Somaiya Vidyavihar University)

Batch: A1 **Roll No.:16010120015**

Experiment No.10

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 Describe various algorithm design strategies to solve different problems and analyse Complexity.

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**
3. **http://www.math.utah.edu/~alfeld/queens/queens.html**

4. **http://www-isl.ece.arizona.edu/ece175/assignments275/assignment4a/Solving%208%20queen%20problem.pdf**
5. **http://www.slideshare.net/Tech_MX/8-queens-problem-using-back-tracking**
6. **http://www.mathcs.emory.edu/~cheung/Courses/170.2010/Syllabus/Backtracking/8queens.html**
7. **http://www.geeksforgeeks.org/backtracking-set-3-n-queen-problem/**
8. **http://www.hbmeyer.de/backtrack/achtdamen/eight.htm**

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.

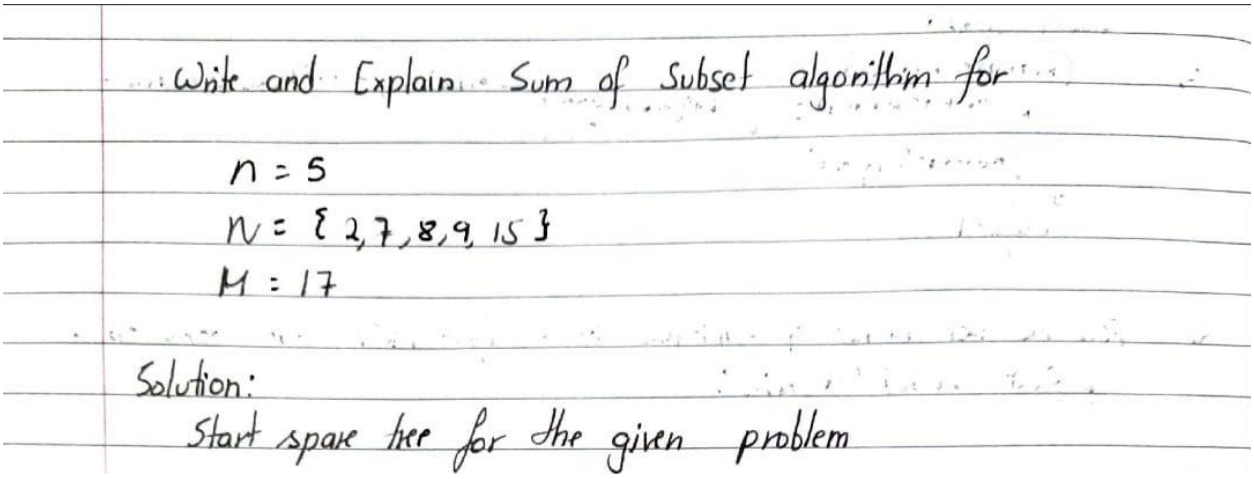
Similarly, assume the array is presorted and we found one subset. We can generate next node excluding the present node only when inclusion of next node satisfies the constraints.

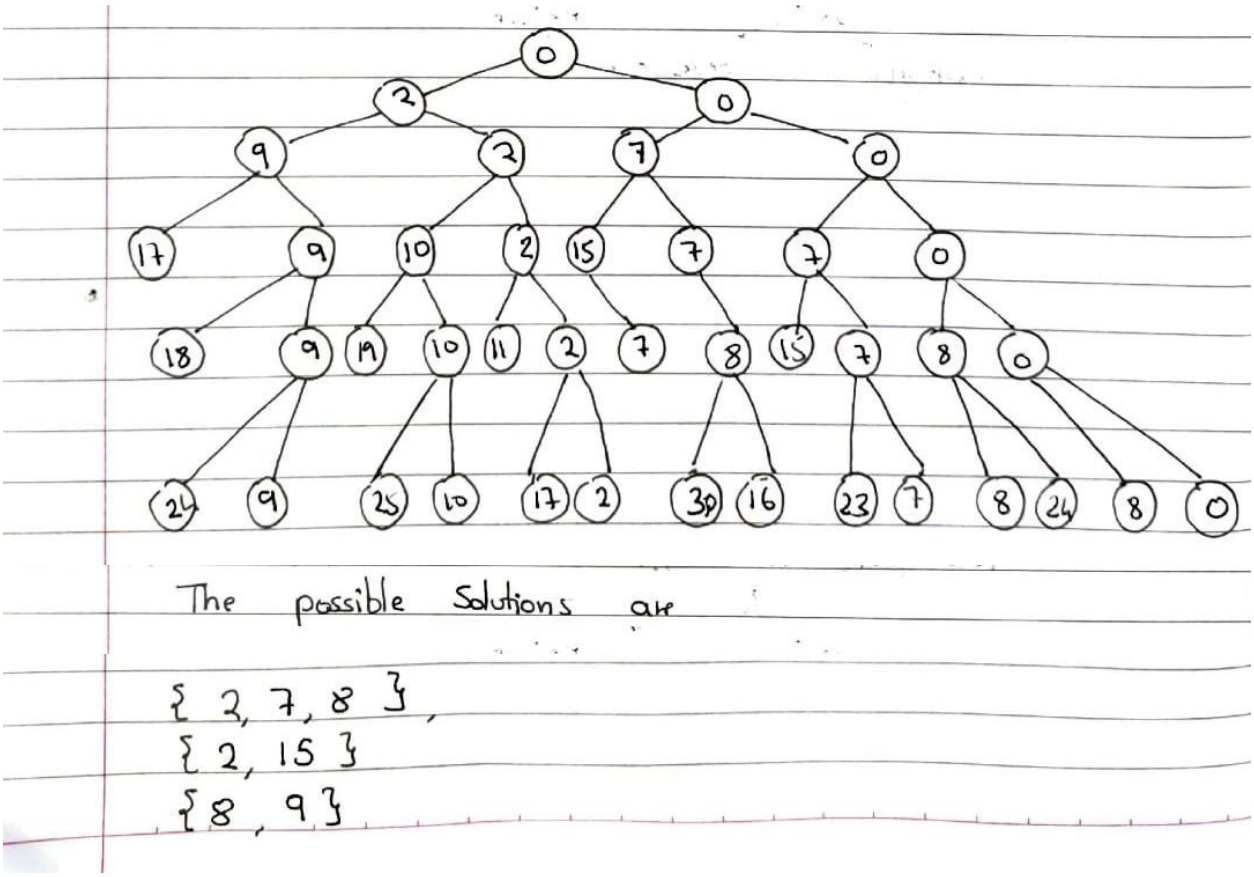
if ((S+ r- W[k]>=m)and (S+ W[k+1]<=m)) then//generate right {
//child and those satisfying 2 bounding functions

X{k]=0;

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

EXAMPLE SUM OF SUBSET PROBLEM ALONG WITH STATE SPACE TREE





CODE

```
import java.util.*;

class SubSet
{
  int set[];
  int sum;

  Stack<Integer> solutionSet;

  boolean hasSolution;

  SubSet(int set[], int sum){
      this.set = set;
      this.sum = sum;
      this.solutionSet = new Stack<>();
```

```
      hasSolution = false;
  }

  public void solve(int s, int idx)

{

    if(s>sum)
        return;


    if(s==sum){
        hasSolution = true;


        displaySolutionSet();

        return;
    }



    for(int i=idx; i<set.length; i++)
{

        solutionSet.push(set[i]);
        solve(s+set[i],i+1);

        solutionSet.pop();

    }


  }

  private void displaySolutionSet()
{

    for (Integer item: solutionSet)
{
      System.out.print(item+" ");
```

```
    }

        System.out.println();
  }

}


public class Main
{
    public static void main(String[] args) {

        Scanner s = new Scanner(System.in);

        System.out.print("Enter the number of elements in the set : ");
        int []set = new int[s.nextInt()];
    for(int i=0;i<set.length;i++){
            System.out.printf("Enter the value %d : ",(i+1));
            set[i]=s.nextInt();
    }

    System.out.print("\nYour set : ");
       for(int i=0;i<set.length;i++){
            System.out.printf("\t%d ",(set[i]));
    }
      System.out.print("\nEnter the Size : ");
       int size = s.nextInt();
       SubSet ss = new SubSet(set, size);
       ss.solve(0,0);

       if(ss.hasSolution == false)
      System.out.print("No Solution");
    }
}
```

OUTPUT

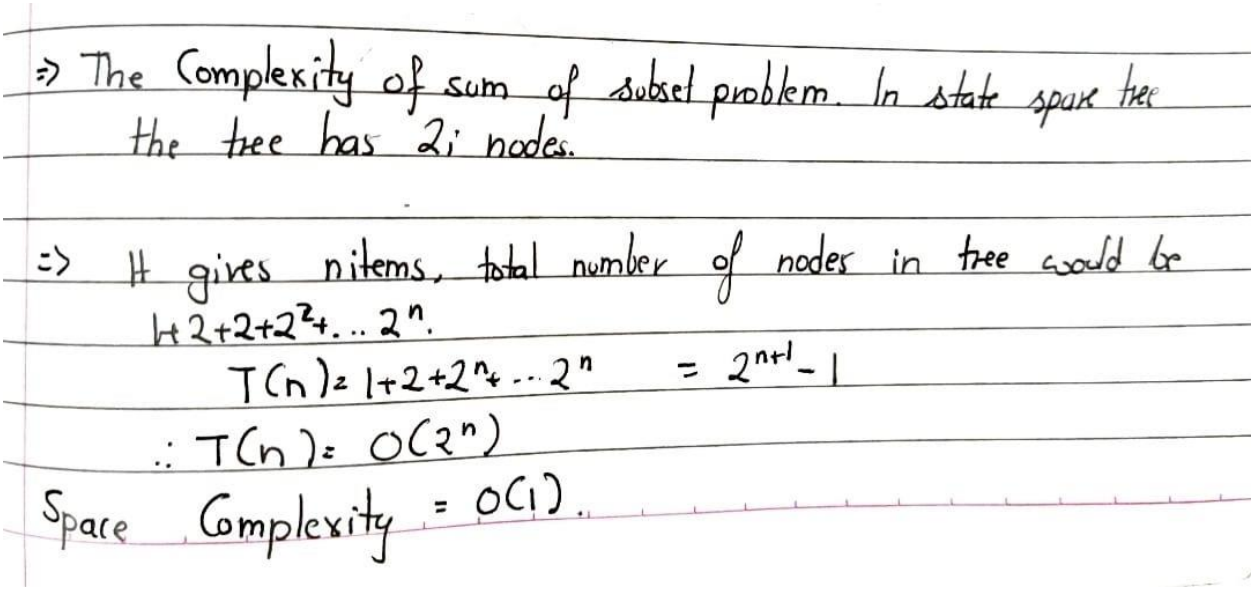
```
Enter the number of elements in the set : 5
Enter the value 1 : 2
Enter the value 2 : 7
Enter the value 3 : 8
Enter the value 4 : 9
Enter the value 5 : 15

Your set :      2       7       8       9       15
Enter the Size : 17
2 7 8
2 15
8 9
```

```
Enter the number of elements in the set : 5
Enter the value 1 : 2
Enter the value 2 : 5
Enter the value 3 : 7
Enter the value 4 : 11
Enter the value 5 : 15

Your set :      2       5       7       11      15
Enter the Size : 21
No Solution
```

ANALYSIS OF BACKTRACKING SOLUTION FOR SUM OF SUBSET PROBLEM



CONCLUSION

With the help of this experiment we were able to learn, understand and implement the following concepts:

- ✓ Implementation of sum of subset problem using Backtracking in JAVA Programming language
- ✓ Complexity Analysis of sum of subset problem.