**Batch: A1 Roll No.: 1911004**

**Experiment No 9**

**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:**

|  |  |
| --- | --- |
| Sr. No | Objective |
| CO 1 | Analyze the asymptotic running time and space complexity of algorithms. |
| CO 2 | Describe various algorithm design strategies to solve different problems and analyze  Complexity. |
| CO 3 | Develop string matching techniques |
| CO 4 | Describe the classes P, NP, and NP-Complete |

**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**](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**](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**](http://www.mathcs.emory.edu/~cheung/Courses/170.2010/Syllabus/Backtracking/8queens.html)
7. [**http://www.geeksforgeeks.org/backtracking-set-3-n-queen-problem/**](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](http://en.wikipedia.org/wiki/Power_set) contains all those subsets generated from a given set. The size of such a power set is 2N.

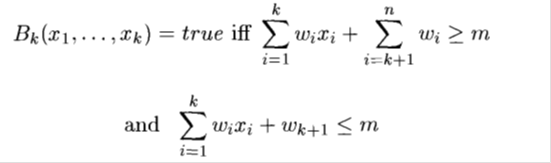
***Input:***

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



**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 {

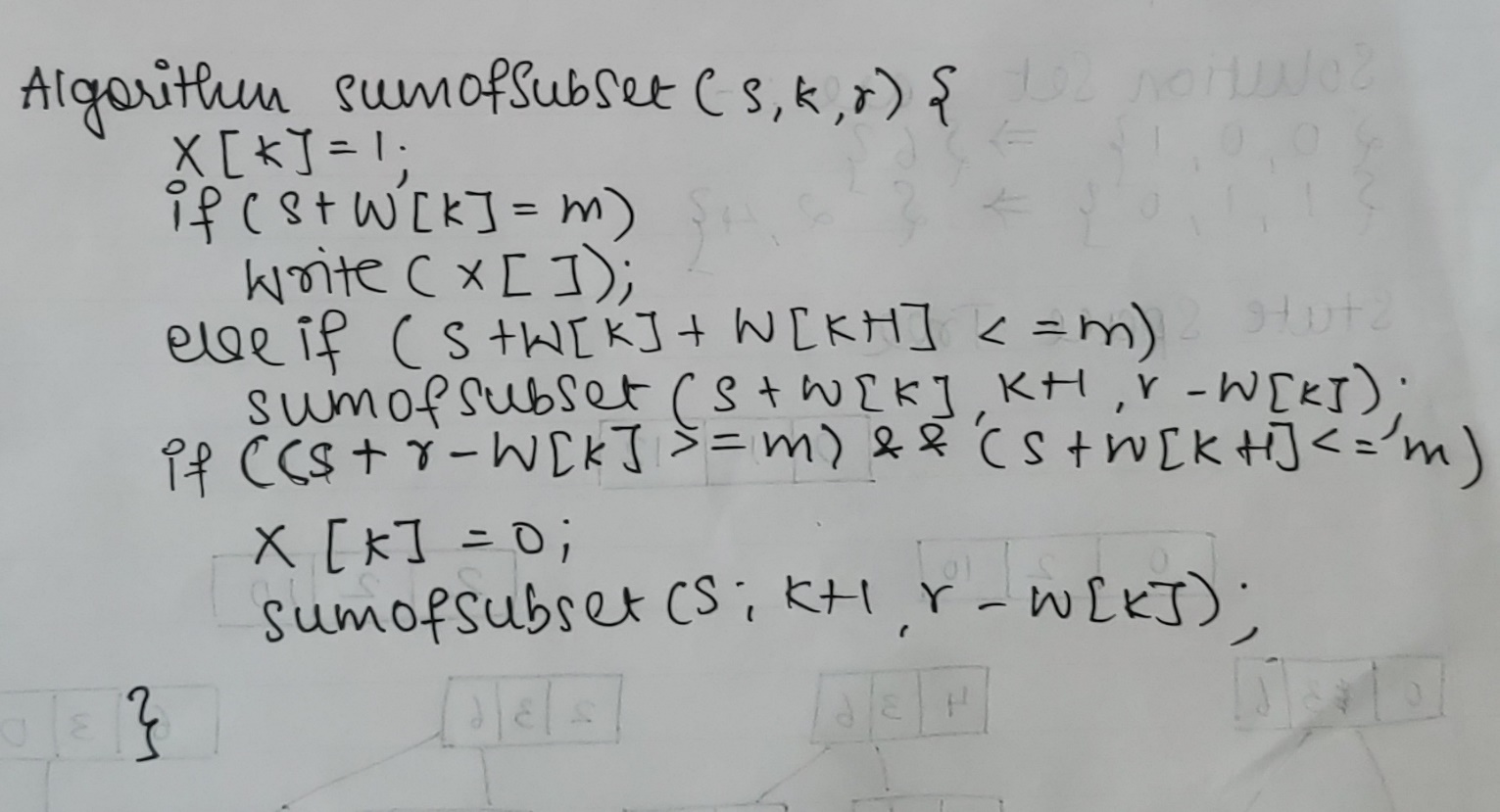
child and those satisfying 2 bounding functions

X{k]=0;

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

}

}



**Implementation(Code):**

import java.util.\*;

class sumOfSubSets {

public static int sum=0,l=0,tS=0,subSetSolnC = 0,set[],m=0;

public static void main(String[] args) {

System.out.println("Sum Of Subsets");

Scanner ob = new Scanner(System.in);

System.out.print("Enter no of nos. ");

int n=ob.nextInt(),i,j;

set = new int[n];

int solnSubSet[][]=new int[n][n];

System.out.print("Enter weights ");

for(i=0;i<n;i++){

set[i]=ob.nextInt();

tS+=set[i];

}

System.out.print("Enter Max Sum "); m=ob.nextInt();

sumOfSubSets(solnSubSet, sum, l, tS);

System.out.println("Total soln sets = "+subSetSolnC);

System.out.println("Soln sets are");

System.out.print("\t");

for(j=0;j<n;j++){

System.out.print(" "+set[j]+" ");

}

System.out.println("");

ptr(solnSubSet,n);

System.out.println("Sub Sets are ");

for(i=0;i<n;i++){

System.out.print("{ ");

for(j=0;j<n;j++)

if (solnSubSet[i][j]==1)

System.out.print(set[j]+" ");

System.out.print("}\n");

}

}

public static void ptr(int s[][],int n){

for(int i=0;i<subSetSolnC;i++){

System.out.print((i+1)+" \t");

for(int j=0;j<n;j++)

System.out.print(" "+s[i][j]+" ");

System.out.println(" ");

}

}

public static void prt(int a[]){

for(int i=0;i<a.length;i++)

System.out.print(a[i]+" ");

System.out.println(" ");

}

static void sumOfSubSets(int x[][],int sum,int l,int tS){

x[subSetSolnC][l]=1;

if((sum+set[l])==m){

prt(x[subSetSolnC]); subSetSolnC++; System.out.println(" ");

return;

}

else if((sum+set[l]+set[l+1])<=m){

prt(x[subSetSolnC]);

sumOfSubSets(x, sum+set[l], l+1, tS-set[l]);

}

if(((sum+tS-set[l])>=m)&&((sum+set[l+1])<=m)){ //

prt(x[subSetSolnC]);

x[subSetSolnC][l]=0;

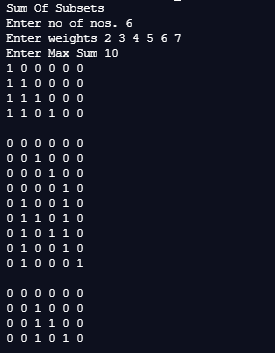
sumOfSubSets(x, sum, l+1, tS-set[l]);

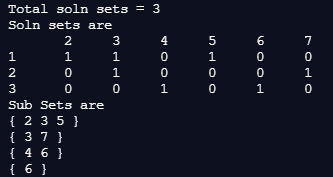
}

}

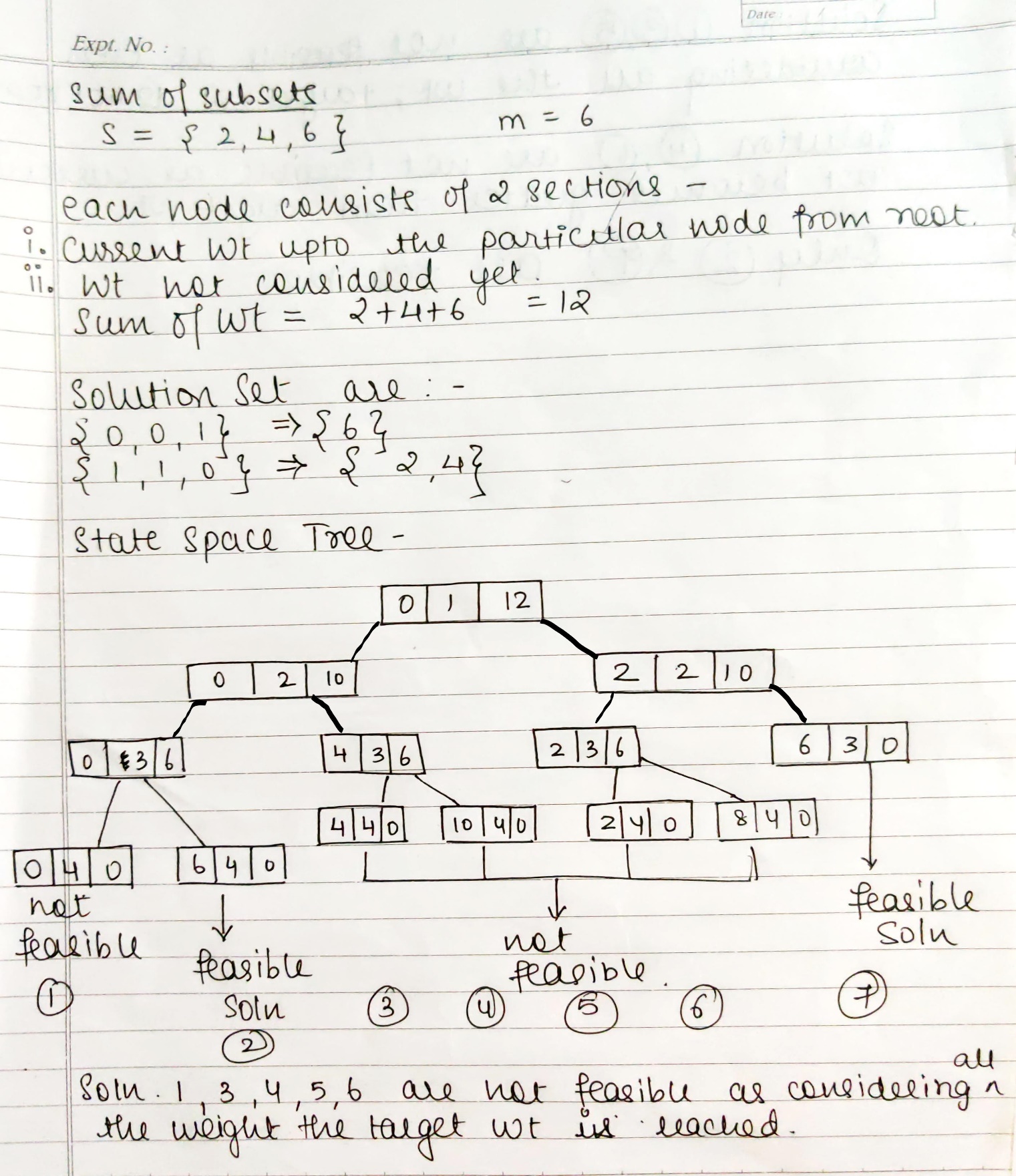
}

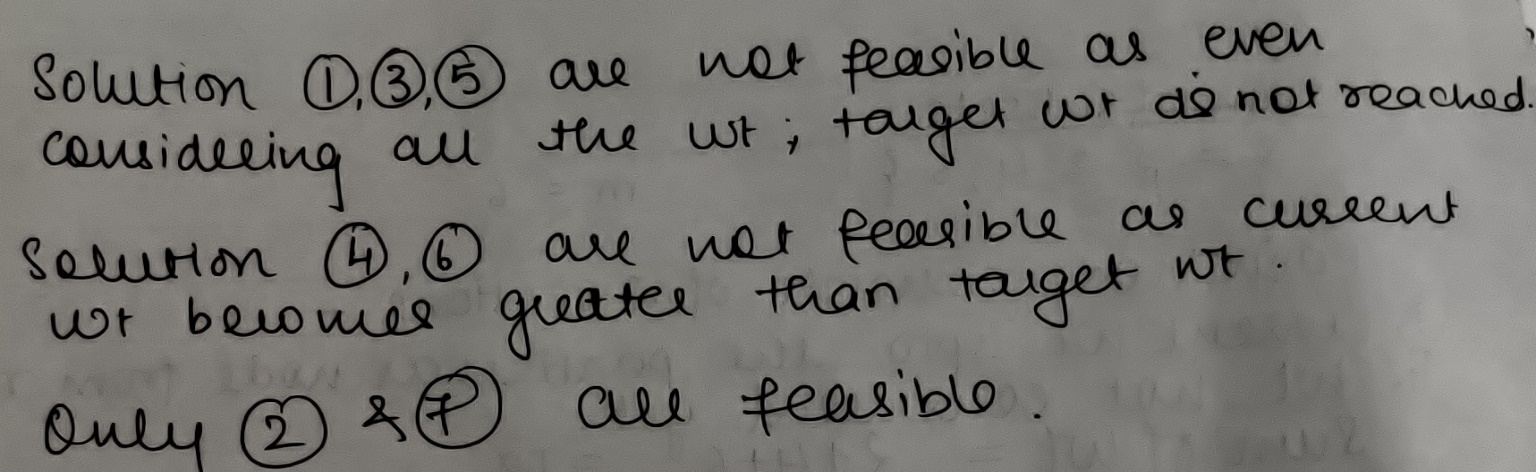
**Output:**



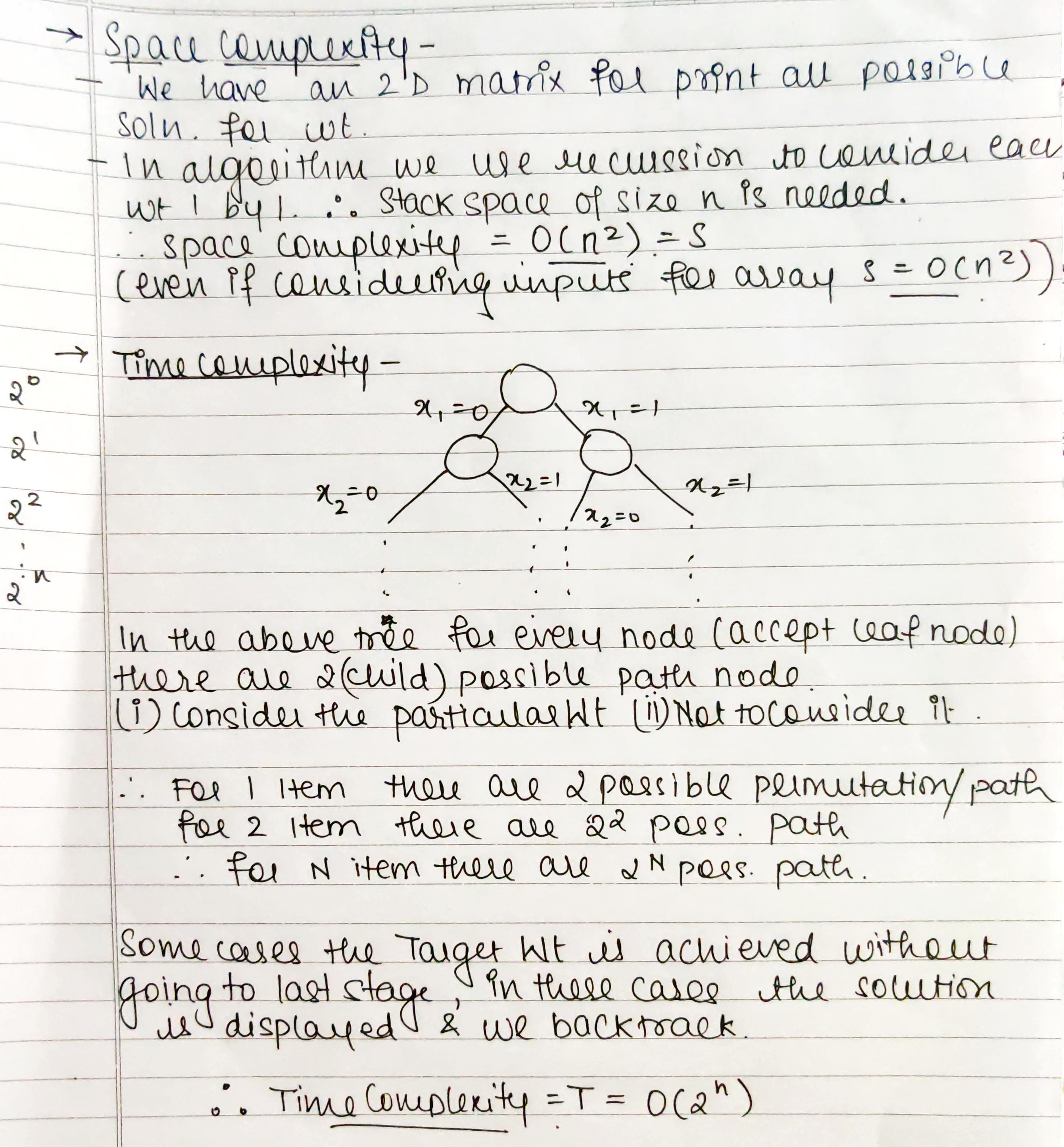


**Example sum of subset Problem along with state space tree:**





**Analysis of Backtracking solution for sum of subset Problem:**



**Conclusion:**

We successfully understood & implemented Sum of SubSets using backtracking ; thus obtained all possible Solution Set satisfying the Target Weight using Backtracking Algorithm in java.