**Experiment No: 12**

**AIM:** Implementation of sum of subset problem using Backtracking

**THEORY:**

Given *n* distinct positive numbers(called weights), we need to find all combinations of these numbers whose sums are *m.* This is called the *sum of subsets problem.*

The problem is efficiently solved in programming using a concept called Backtracking.

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.

The following bounding functions are used:

**ALGORITHM:**

**Algorithm** SumOfSub(s,k,r)

// Find all subsets of w[1 :n] that sum to m. The values of x[j],

// 1<j < k have already been determined.

// The w[j]'s are in nondecreasing order.

// It is assumed that w[1] < m

{

// Generate 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) 12

thenSumOfSub(s+w[k],k+1,r-w[k]);

// Generate right child and evaluate Bk.

if ((s + r – w[k] > m) and (s + wk+1 < m)) then

{

x[k]=0;

SumOfSub(s,k+1,r-w[k]);

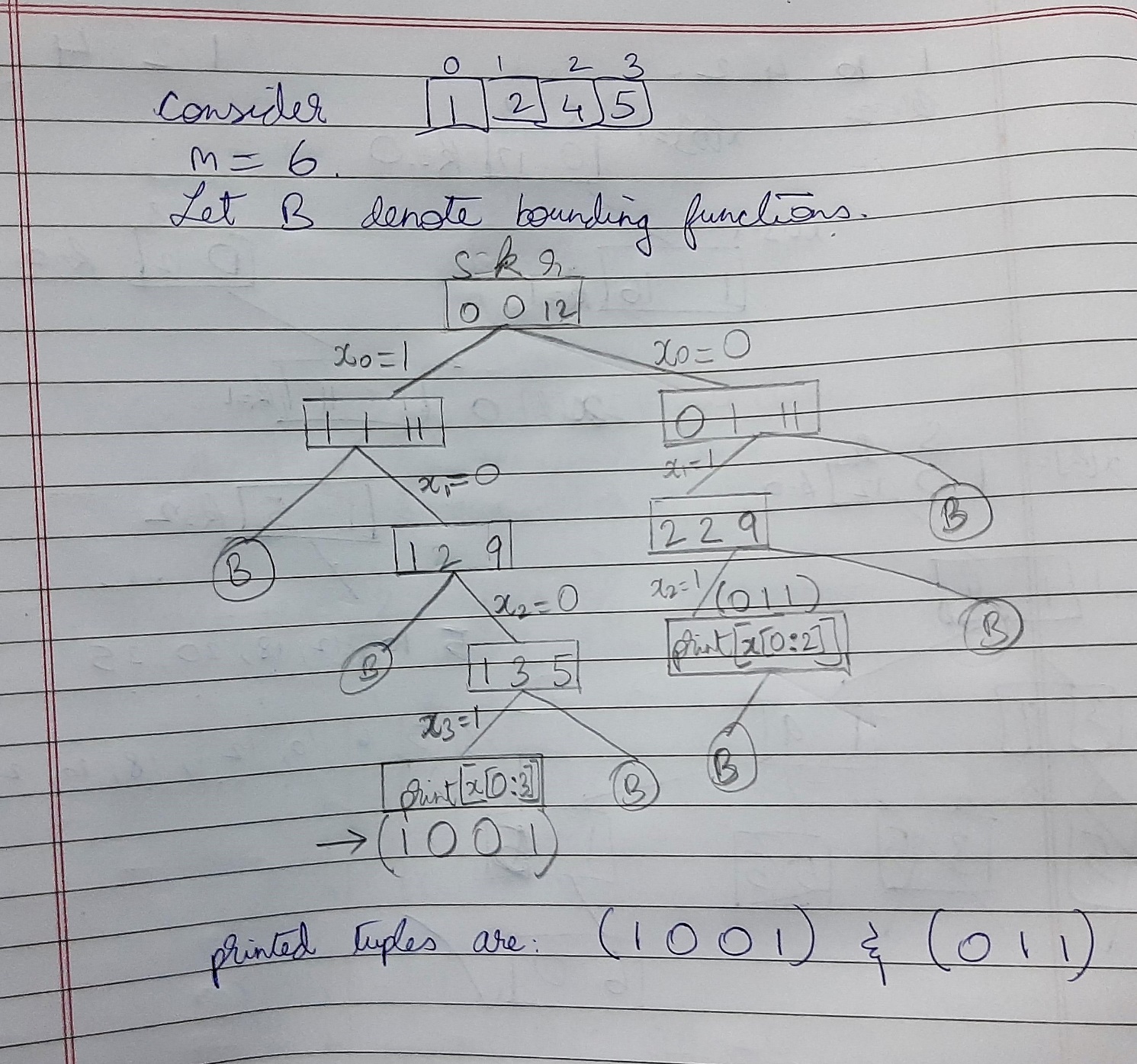
}

}

*Time Complexity*

• The time complexity of this algorithm is O (2^n).

*Problem Tracing*



PROGRAM IMPLEMENTATION:

#include<iostream>

using namespace std;

int n,m,\*w,\*x; //soln vector

void show\_set(int k)

{

cout<<"( ";

for(int i=0;i<=k;i++)

cout<<x[i]<<" ";

cout<<")"<<endl;

}

void sum\_of\_sub(int s, int k, int r)

{

x[k] = 1;

if(s+w[k] == m)

show\_set(k);

else if(s + w[k] + w[k+1] <=m) //current sum is less than or equal to m

sum\_of\_sub(s+w[k],k+1,r-w[k]);

if( (s + r - w[k] >= m) && (s + w[k+1] <=m)) //bounding function says that

{ //w obtained so far + remaining w

x[k] = 0; //must be >=m

sum\_of\_sub(s,k+1,r-w[k]);

} //and the w obtained so far + next

//w must be <=m

}

int main()

{

int r=0; //temp sum

cout<<"Enter number of elements:\n";

cin>>n;

cout<<"Enter "<<n<<" elements: ";

w=new int[n];

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

cin>>w[i],r+=w[i];

x = new int[n];

cout<<"\nEnter the sum: ";

cin>>m;

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

x[i]=0;

int s=0,k=0;

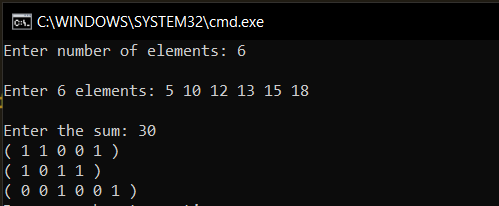
sum\_of\_sub(s,k,r);

return 0;

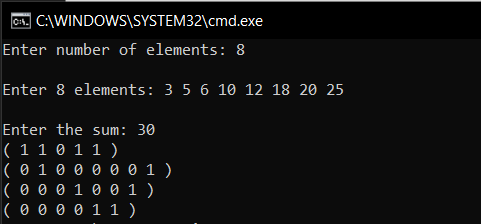
}

OUTPUTS:

1. When n=6



1. When n=8

****

**Conclusion**:

* **This algorithm works when the elements are arranged in non-decreasing order of their weights.**
* **Time complexity of the algorithm is O(2^n)**