**Batch: B1 Roll No.: 1611077**

**Experiment No. \_\_\_7\_\_\_**

**Grade: AA / AB / BB / BC / CC / CD /DD**

**Signature of the Staff In-charge with date**

|  |
| --- |
| **Title: Implementation of Backtracking Algorithm** |



**Objective:** To learn the Backtracking strategy of problem solving for SUM OF SUBSET 

**CO to be achieved:**

|  |  |
| --- | --- |
| Sr. No | Objective |
| CO 1 | Compare and demonstrate the efficiency of algorithms using asymptotic complexity notations. |
| CO 2 | Analyze and solve problems for divide and conquer strategy, greedy method, dynamic programming approach and backtracking and branch & bound policies. |
| CO 3 | Analyze and solve problems for   different string matching algorithms. |



**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:**

In computer science, the subset sum problem is an important problem in complexity theory and cryptography. The problem is this: given a set (or multiset) of integers, is there a non-empty subset whose sum is zero? For example, given the set {−7, −3, −2, 5, 8}, the answer is *yes* because the subset {−3, −2, 5} sums to zero. The problem is NP-complete, meaning roughly that while it is easy to confirm whether a proposed solution is valid, it may inherently be prohibitively difficult to determine in the first place whether any solution exists.



**New Concepts to be learned:**

Application of algorithmic design strategy to any problem, Backtracking method of problem solving vs. other methods of problem solving SUM OF SUBSET problem and its applications.

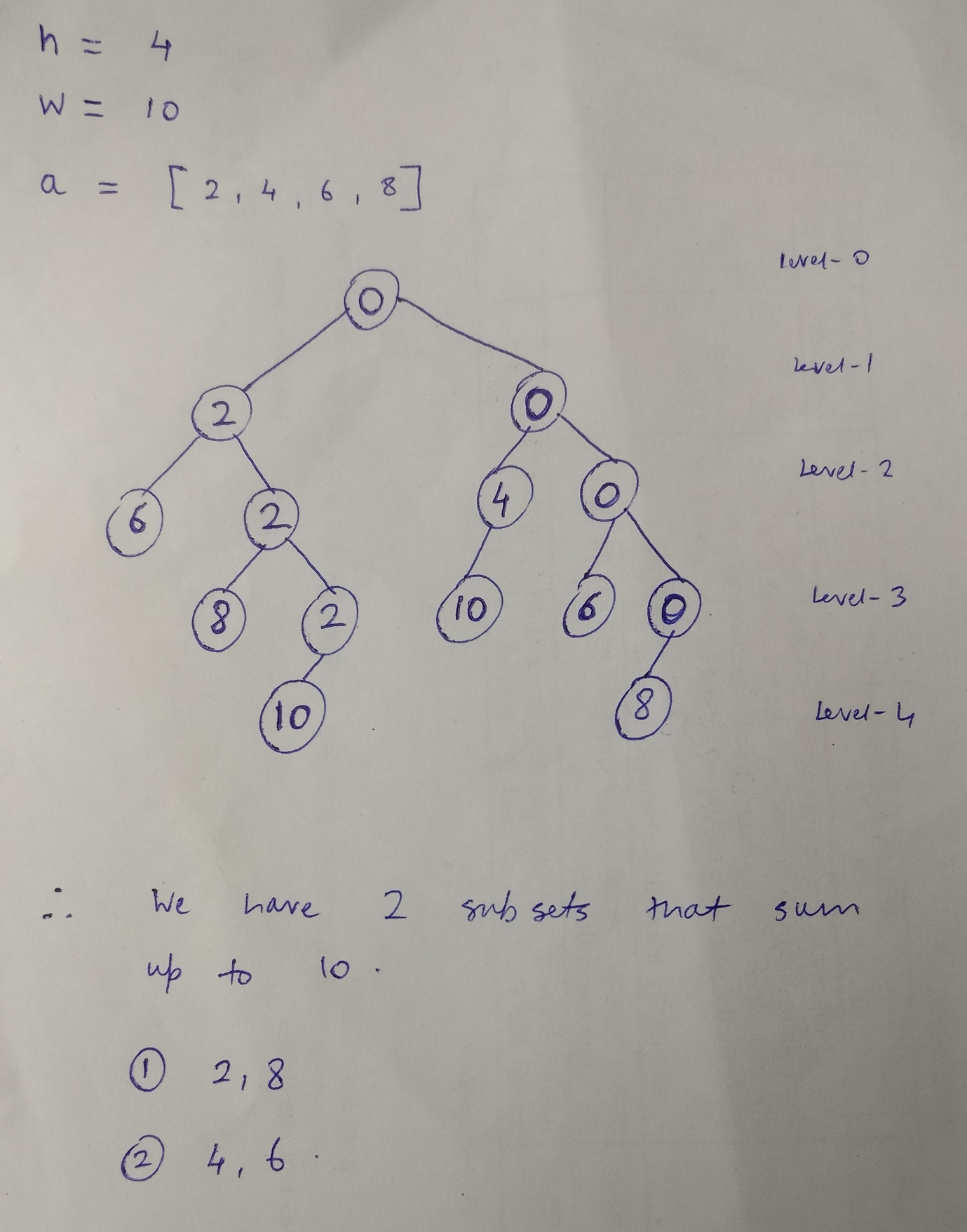


**Algorithm SUM OF SUBSET Problem:-**

The algorithm for the approximate subset sum problem is as follows:

1.Iinitialize a list *S* to contain one element 0.  
2. For each *i* from 1 to *N* do  
 let *T* be a list consisting of *xi* + *y*, for all *y* in *S*  
 let *U* be the union of *T* and *S*  
 sort *U*  
 make *S* empty   
 let *y* be the smallest element of *U*   
 add *y* to *S*   
 3. for each element *z* of *U* in increasing order do  
 //trim the list by eliminating numbers close to one another  
 //and throw out elements greater than *s*  
 if *y* + *cs*/*N* < *z* ≤ *s*, set *y* = *z* and add *z* to *S*   
4. if *S* contains a number between (1 − *c*)*s* and *s*, output *yes*, otherwise *no*

**Example sum of subset problem:**



**Time Complexity Analysis:**

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

**IMPLEMENTATION:**

import java.util.\*;

class Sum

{

static int total,n,v[],a[],sum;

public static void main(String[] args)

{

Sum m=new Sum();

Scanner sc =new Scanner(System.in);

System.out.println("Enter the Number of elements: ");

n=sc.nextInt();

a=new int[n];

v=new int[n];

System.out.println("Enter the sum to be obtained: ");

total=sc.nextInt();

System.out.println("Enter the elements: ");

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

{

a[i]=sc.nextInt();

sum=sum+a[i];

}

m.sub(0,0,sum);

System.out.print("\n\nProgram by Mihir Gandhi B1 1611077\n");

}

void sub(int s,int l,int rs)

{

v[l]=1;

if(a[l]+s==total)

{

System.out.print("The subset is: ");

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

{

if(v[k]==1)

{

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

}

}

System.out.println();

}

else if(s+a[l]+a[l+1]<=total)

{

sub(s+a[l],l+1,rs-a[l]);

}

if(s+rs-a[l]>=total && s+a[l+1]<=total)

{

v[l]=0;

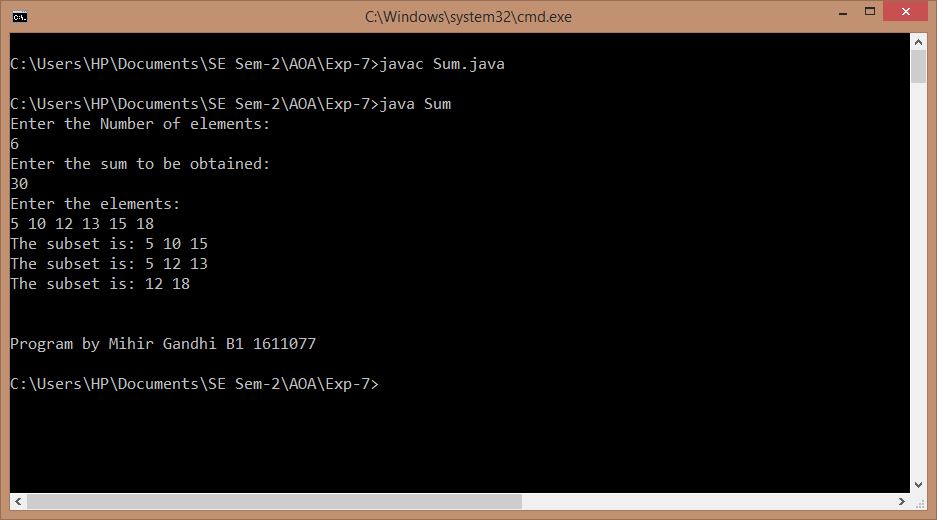
sub(s,l+1,rs-a[l]);

}

}

}

**OUTPUT:**

****

**Conclusion:**

Thus, by using method of backtracking, we have successfully solved the sum of subset problem and implemented the code for the same. The actual outcome matched with the expected outcome.