Name Chirag Rana
SE COMPS
2018130043

Experiment 1

Aim: Sum of Subsets (C, B) using a Backtracking strategy.

Theory:

General Backtracking method:

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 (by time, here, is referred to the time elapsed till reaching any level of the search tree).

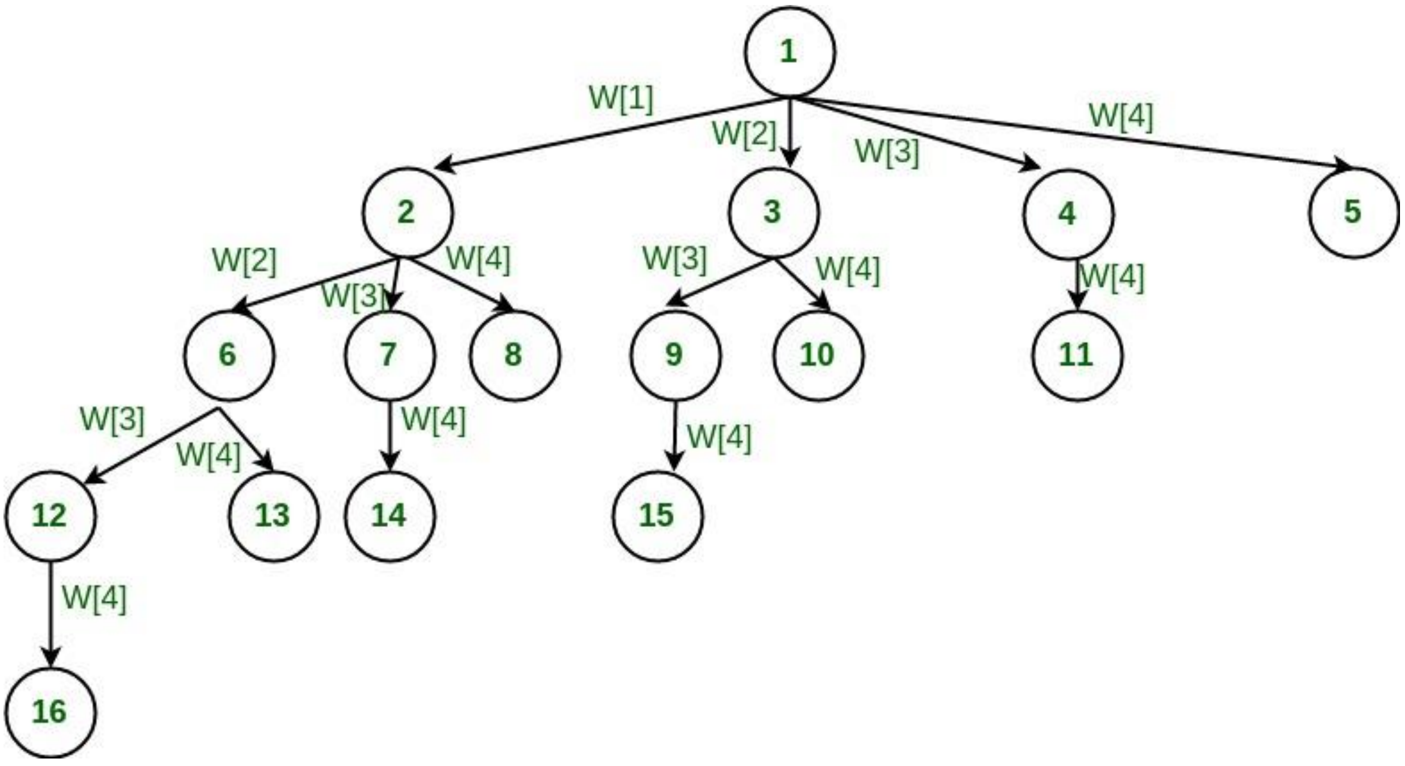
According to the wiki definition,

Backtracking can be defined as a general algorithmic technique that considers searching every possible combination in order to solve a computational problem.

Backtracking Algorithm for Subset Sum

Using exhaustive search, we consider all subsets irrespective of whether they satisfy given constraints or not. Backtracking can be used to make a systematic consideration of the elements to be selected.

Assume given set of 4 elements, say **w [1] … w [4]**. Tree diagrams can be used to design backtracking algorithms. The following tree diagram depicts approach of generating variable sized tuple.



SUBSET-SUM PROBLEM: Subset-Sum Problem is finding a subset of a given set S = {s1,s2….sn} of n positive integers whose sum is equal to a given positive integer d.

For example, for S = {1, 2, 5, 6, 8) and d = 9, there are two solutions: {1, 2, 6} and {1, 8}.

Of course, some instances of this problem may have no solutions. It is convenient to sort the set‘s elements in increasing order.

So we will assume that s1 ≤ s2 ≤ ……. ≤ sn

The state-space tree can be constructed as a binary tree as that in the following figure for the instances S = (3, 5, 6, 7) and d = 15.

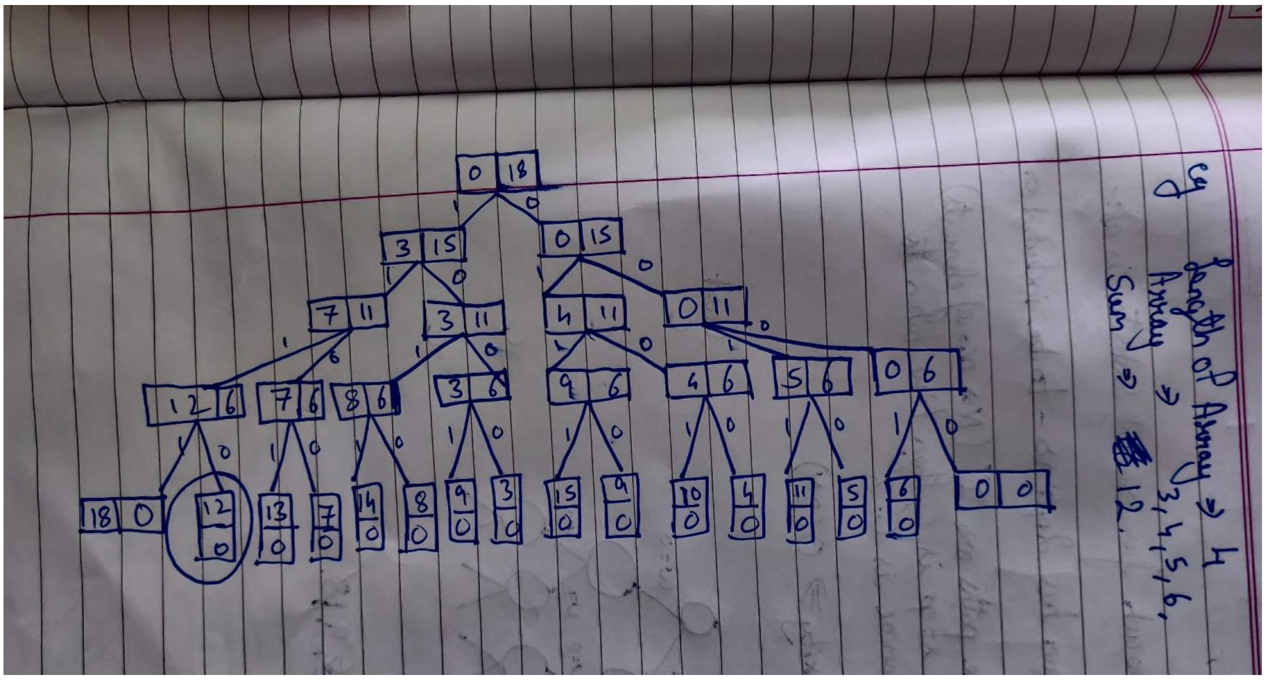
The root of the tree represents the starting point, with no decisions about the given elements made as yet. Its left and right children represent, respectively, inclusion and exclusion ofs1 in a set being sought. Similarly, going to the left from a node of the first level corresponds to inclusion of s2, while going to the right corresponds to its exclusion, and soon.

Thus, a path from the root to a node on the ith level of the tree indicates which of the first i numbers have been included in the subsets represented by that node. We record the value of s‘ the sum of these numbers, in the node, Ifs is equal to d. we have a solution to the problem.

We can either, report this result and stop or, if all the solutions need to he found, continue by backtracking to the node‘s parent.

If s‘ is not equal to d, we can terminate the node as nonpromising if either of the two inequalities holds:

Design and implementation:

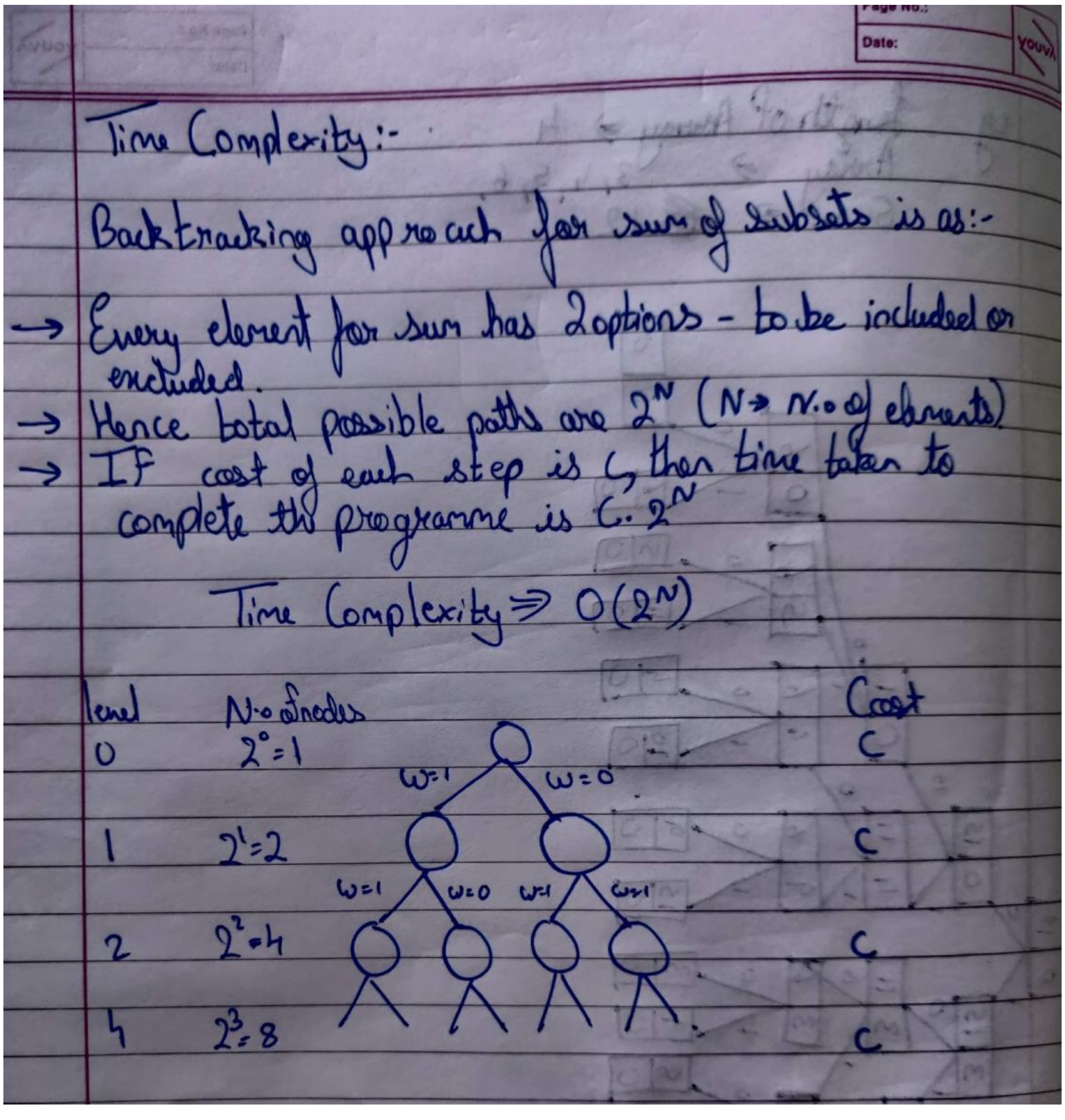


Algorithm:

Let, S is a set of elements and m is the expected sum of subsets. Then:

1. Start with an empty set.
2. Add to the subset, the next element from the list.
3. If the subset is having sum m then stop with that subset as solution.
4. If the subset is not feasible or if we have reached the end of the set then backtrack through the subset until we find the most suitable value.
5. If the subset is feasible then repeat step 2.
6. If we have visited all the elements without finding a suitable subset and if no backtracking is possible then stop without solution.

Time Complexity:



Page No.:
Date:

Time Complexity:-

Backtracking approach for sum of subsets is as:-

→ Every element for sum has 2 options - to be included or excluded.

→ Hence total possible paths are 2^N ($N \Rightarrow$ No. of elements)

→ IF cost of each step is C, then time taken to complete the programme is $C \cdot 2^N$

Time Complexity $\Rightarrow O(2^N)$

level	No. of nodes	Cost
0	$2^0 = 1$	C
1	$2^1 = 2$	C
2	$2^2 = 4$	C
4	$2^3 = 8$	C

w=1
w=0
w=1
w=0
w=1
w=1

Code:

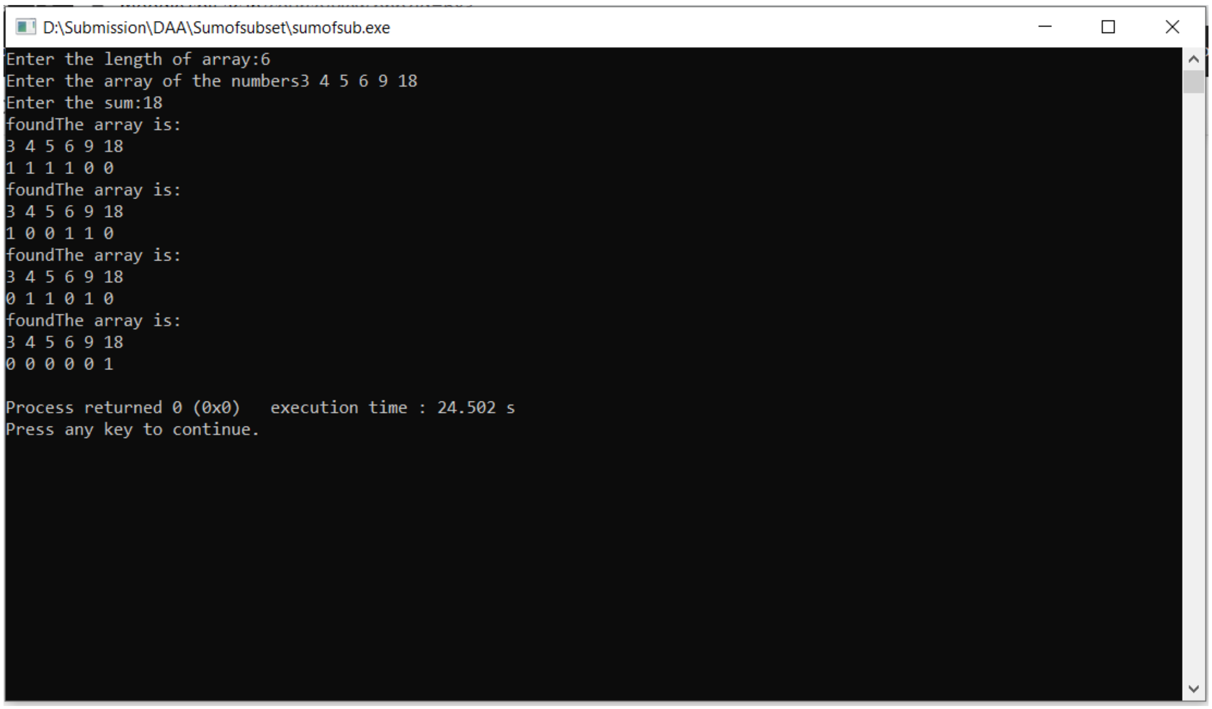
```
#include<stdio.h>
#include<stdlib.h>

void display(int array[],int binarray[],int length);
void SumofSubset(int array[],int binarray[],int target_sum,int index,int sum,int length);

int main(){

        int len,sumval = 0;
        int* array;
        int* binarray;

        printf("Enter the length of array:");
        scanf("%d",&len);

        array = (int*)malloc(len*sizeof(int));
   binarray = (int*)malloc(len*sizeof(int));
      printf("Enter the array of the numbers");
        for (int i = 0;i < len;i++){
                scanf("%d",&array[i]);
                binarray[i] = 0;
        }

        printf("Enter the sum:");
   scanf("%d",&sumval);
        SumofSubset(array,binarray,sumval,0,0,len);
        return 0;
}

void display(int array[],int binarray[],int length){
        printf("The array is:\n");
        for(int i = 0;i < length;i++){
                printf("%d ",array[i]);
        }
   printf("\n");
   for(int i = 0;i < length;i++){
                printf("%d ",binarray[i]);
        }
   printf("\n");
}
```

```
void SumofSubset(int array[],int binarray[],int target_sum,int index,int sum, int length){
         if(target_sum == sum){
            printf("found");
       display(array,binarray,length);
       binarray[index] = 1;
                   SumofSubset(array,binarray,target_sum,index,sum - array[index-1],length);
                   binarray[index] = 0;
         }else{
                   for (int i = index;i < length;i++){
          binarray[i] = 1;
                            SumofSubset(array,binarray,target_sum,i + 1,sum + array[i],length);
                            binarray[i] = 0;
          }
   }
}
```

Output:

Input array : 3 4 5 6 18

Sum: 18

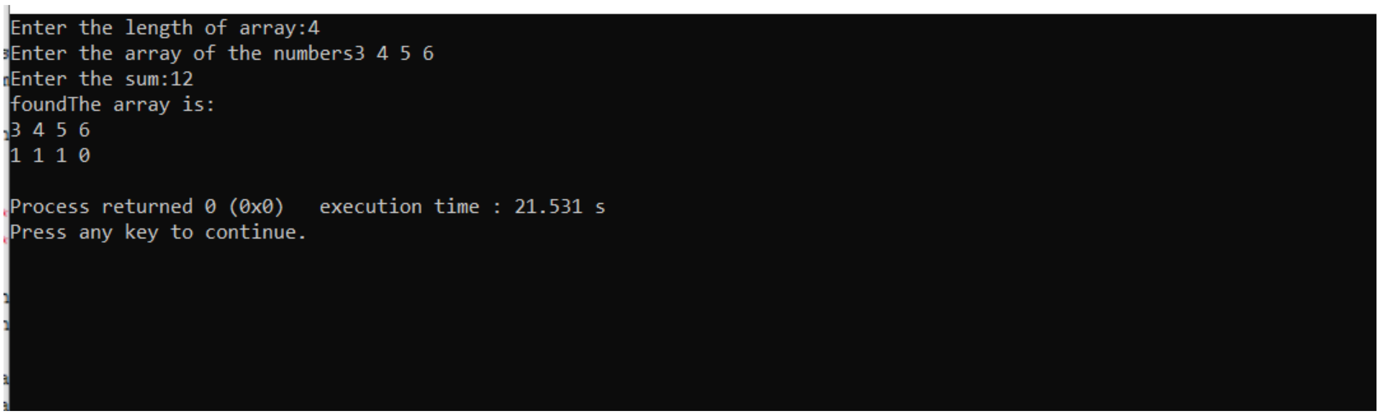


D:\Submission\DAA\Sumofsubset\sumofsub.exe

```
Enter the length of array:6
Enter the array of the numbers3 4 5 6 9 18
Enter the sum:18
foundThe array is:
3 4 5 6 9 18
1 1 1 1 0 0
foundThe array is:
3 4 5 6 9 18
1 0 0 1 1 0
foundThe array is:
3 4 5 6 9 18
0 1 1 0 1 0
foundThe array is:
3 4 5 6 9 18
0 0 0 0 0 1

Process returned 0 (0x0)   execution time : 24.502 s
Press any key to continue.
```

Input array:3 4 5 6

Sum: 12



```
Enter the length of array:4
Enter the array of the numbers3 4 5 6
Enter the sum:12
foundThe array is:
3 4 5 6
1 1 1 0

Process returned 0 (0x0)   execution time : 21.531 s
Press any key to continue.
```

Conclusion:

1] I implemented the Sum of Subset problem using Backtracking Algorithm.

2] I used two arrays one the actual array with input values and other consisting of the position of the answer found.