UNIVERSITY INSTITUTE OF COMPUTING

MASTER OF COMPUTER APPLICATIONS

Design and Analysis of Algorithms

23CAH511

UNIT-3

DISCOVER . LEARN . EMPOWER

SUM OF SUBSETS PROBLEM

- This is a simple algorithm, but it demonstrates that sometimes you need to return to a previous state and re-evaluate a previous decision in order to solve a problem.

  Backtracking is a general algorithmic technique that considers searching every possible combination in order to solve an optimization problem. Backtracking uses **depth-first search** approach. By inserting more knowledge of the problem, the search tree can be pruned to avoid considering cases that don't look promising. While backtracking is useful for hard problems to which we do not know more efficient solutions, it is a poor solution for the everyday problems that other techniques are much better at solving.

Sum of Subsets Using Backtracking

- Subset sum problem is to find subset of elements
- that are selected from a given set whose sum adds
- up to a given number .
- Ex: let A be a set
- A={5,7,10,12,15,18,20}
- and given sum m=35
- so we have the following subsets that add
- up to 35 are:
- {15,20},{18,7,10},{5,10,20}, and
- {18,12,5}

Assumption and considerations

- :

  - set contains non-negative values.
  - input set is unique (no duplicates are presented).

Solution Using NAÏVE APPROACH

Let $w_1, w_2, w_3, \ldots\ldots\ldots\ldots w_n$
be the given set of n numbers
Let $x_1, x_2, x_3, \ldots\ldots x_n$ belongs to $\{0,1\}$
If $x_i = 1$ then w_i is chosen
$x_i = 0$ then w_i is not chosen

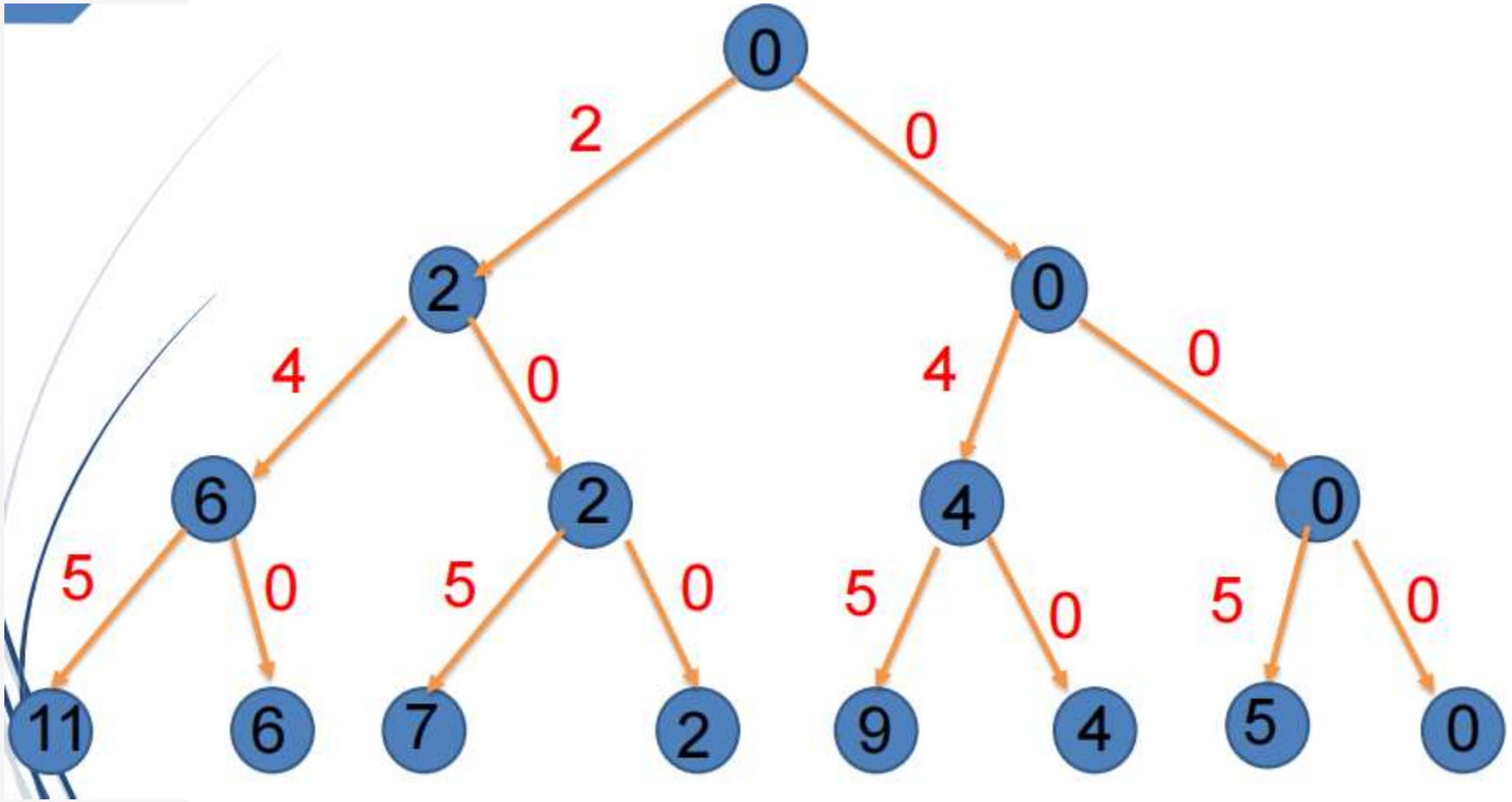
So in totality we have 2^n subsets

$2x2x2x2x2x\ldots\ldots 2 = 2^n$

$x_1\ x_2\ x_3\ \ldots\ldots x_n$

Solution Using NAÏVE APPROACH

A={2,4,5} and m=9



Solution Using BACKTRACKING

Promising Conditions

$s+r>=m$

$s+w(k+1)<=m$

where

$s=\sum w_i x_i$ from i=1 to k-1

$r=\sum w_i$ from i=k to n

Algorithm

Algorithm:

```
sumofsubset(s,k,r)
{
        X[k]=1;
        if (s+W[k]=m) then write(X[1:k]);
        else if (s+W[k]+W[k+1]<=m)
         then sumofsubset(s+W[k], k+1,r- W[k]);

        if ((s+ r-W[k]>=m)and(s+ W[k+1]<=m)) then
        {
                X[k]=0;
                sumofsubset(s, k+1, r- W[k]);
        }
}
```

Time Complexity

Complexity Analysis:

Time Complexity: O(sum*n), where sum is the ‘target sum’ and ‘n’ is the size of array.

Auxiliary Space: O(sum*n), as the size of 2-D array is sum*n.

References

1) https://www.tutorialspoint.com/data_structures_algorithms/divide_and_conquer.htm

2) Data Structures and Algorithms made easy By Narasimha Karumanchi.

3) The Algorithm Design Manual, 2nd Edition by Steven S Skiena

4) Fundamentals of Computer Algorithms - Horowitz and Sahani

THANK YOU