IIIT Allahabad
DAA Assignment-05

Sanket Kokude
IIB2019034

Harshit Kumar
IIB2019035

Viful Nirala
IIB2019036

Abstract—**In this report we designed a Dynamic Programming algorithm to find the number of subsets of given array having a particular XOR value.**

I. INTRODUCTION

Given an array arr[] of size n ,we used dynamic programming approach to find the number of subsets having XOR value as K.
Dynamic Programming (DP) is an algorithmic technique for solving an optimization problem by breaking it down into simpler subproblems and utilizing the fact that the optimal solution to the overall problem depends upon the optimal solution to its subproblems.

II. ALGORITHM DESIGN

Following is Dynamic Programming algorithm :

- First we find a number m, which is actually the maximum value any XOR subset can acquire. We define a number m such that $m = pow(2, (log2(max(arr)) + 1)) - 1$.
- After that check whether the given k is greater or not if it is then return 0.
- After that create a 2D array $dp[n + 1][m + 1]$, such that $dp[i][j]$ equals to the number of subsets having XOR value j from subsets of $arr[0 \ldots i - 1]$.
- We initialize all values of dp[i][j] as 0.
- Set value of dp[0][0] = 1 since XOR of an empty set is 0.
- Iterate over all the values of $arr[i]$ from left to right and for each $arr[i]$, iterate over all the possible values of XOR i.e from 0 to m (both inclusive) and fill the dp array as following:
  for i = 1 to n:
    for j = 0 to m:
      $dp[i][j] = dp[i-1][j] + dp[i-1][j \oplus arr[i-1]]$

- This can be explained as, if there is a subset arr[0…i-2] with XOR value j, then there also exists a subset arr[0…i-1] with XOR value j also if there exists a subset arr[0….i-2] with XOR value $j \oplus arr[i]$ then clearly there exist a subset arr[0…i-1] with XOR value j, as:
  $j \oplus arr[i-1] \oplus arr[i-1] = j$.

- Counting the number of subsets with XOR value k: Since dp[i][j] is the number of subsets having j as XOR value from the subsets of arr[0..i-1], then the number of subsets from set arr[0..n] having XOR value as K will be dp[n][K].

III. ALGORITHM AND ILLUSTRATION

Suppose a given array is $arr = [6, 2, 4, 3, 5]$ and $K = 4$. Then find a number m which is the maximum value any XOR subset can have by
$m = 2^{log_2(max(arr))+1)} - 1$
as $max(arr) = 6$
$m = 2^{log_2 6+1} - 1$
$m = 7$
Then initialise dp (2D array $dp[n+1][m+1]$) with all values as 0 where $n = sizeof(arr)$ i.e $n = 5$.
$dp[0][0] = 1$(Empty set)

So dp matrix will be

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Now we Iterate over all values of arr[i] from i=1 to i=n.Then for each iteration ,we also iterate over all values of XOR.

dp(After 1st Iteration):

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

dp(After 2nd Iteration):

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

dp(After 3rd Iteration):

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 2 & 0 & 2 & 0 & 2 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

dp(After 4th Iteration):

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 2 & 0 & 2 & 0 & 2 & 0 & 2 & 0 \\ 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

dp(After 5th Iteration):

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 2 & 0 & 2 & 0 & 2 & 0 & 2 & 0 \\ 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 \\ 4 & 4 & 4 & 4 & 4 & 4 & 4 & 4 \end{bmatrix}$$

Hence, number of subsets having XOR value k is $dp[n][k] = dp[5][4] = 4$.

IV. ALGORITHM ANALYSIS

Time complexity: We are iterating over whole array one by one finding and storing the count of possible subsets that generate a XOR value j (we say) and store it in the $dp[i][j]$ which will need time to iterate over the array of size n and for each element a loop of time complexity m will be iterated.

$$m = 2^{[log_2(max-element)]+1} - 1$$

This will result in the time complexity of $O(n*m)$.
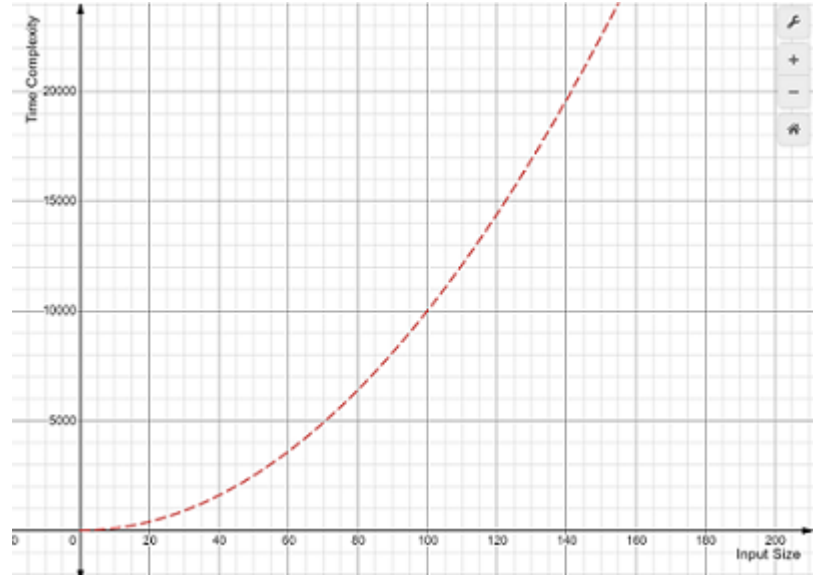
Space complexity: In this approach we will have to store the values for all possible cases that are generated, as the $dp[i][j]$ requires $dp[i-1][j]$ and hence a 2D array of size $n*m$ is required.

This will result in the space complexity of $O(n*m)$.

V. CONCLUSION

We can observe that in Dynamic Programming Approach it may consume more space (i.e.$O(1)$ in case of brute force) but will have better time-complexity than Brute force approach(i.e.$O(2^n)$ in case of brute force).

VI. REFERENCES

1.https://www.geeksforgeeks.org/calculate-xor-1-n/
2. https://www.geeksforgeeks.org/dynamic-programming/
3. Cormen, Leiserson, Rivest, and Stein (2009). Introduction to Algorithms, 3rd edition.

Code for implementation of this paper is given below:

```cpp
#include<bits/stdc++.h>
using namespace std;

int n,k,max_ele=0;

int subsetXOR(int arr[])
{
  // Maximum possible XOR value
  int m = (1 << (int)(log2(max_ele) + 1)) - 1;

  if( k > m )
      return 0;

  int dp[n+1][m+1];
  // Initializing all the values of dp[i][j] as 0
  for (int i=0; i<=n; i++)
    for (int j=0; j<=m; j++)
      dp[i][j] = 0;

  // The xor of empty subset is 0
  dp[0][0] = 1;

  for (int i=1; i<=n; i++){
    for (int j=0; j<=m; j++){
          dp[i][j] = dp[i-1][j] + dp[i-1][j^arr[i-1]];
          // cout<<dp[i][j]<<" ";
      }
      // cout<<"\n";
    }
  return dp[n][k];
}

int main()
{
    cout<<"Enter the size of array : ";
    cin>>n;
    int arr[n];
    cout<<"Enter the array : ";
    for(int i=0;i<n;i++){
        cin>>arr[i];
        if (arr[i] > max_ele)
        max_ele = arr[i];
    }
    cout<<"Enter the XOR value :";
    cin>>k;
  cout << "Count of subsets is " << subsetXOR(arr);
  return 0;
}
```

Listing 1. Code for this paper