DAA ASSIGNMENT-5

JAIDEV DAS IIT2019197
DEEPTARSHI BISWAS IIT2019195
PRIYANSHU P IIT2019196

PROBLEM STATEMENT

Given an array arr[] of n numbers and a number K, find the number of subsets of arr[] having XOR of elements as K.

ALGORITHM

1) We define a number m such that m = m | a[i] for all i from 1 to n. This number is actually the maximum value any XOR subset will acquire.

2) We create a 2D array dp[n+1][m+1], such that dp[i][j] equals the number of subsets having XOR value j from subsets of arr[0…i-1].

3) We fill the dp array as following: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^arr[i-1]]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]

PSEUDO CODE

```
BEGIN:
     Take n and k input
     a[n+1]
     orr=0;
     FOR i 1 to n :
      INPUT(a[i])
      orr=orr|a[i]

     IF orr<k:
     PRINT 0
```

```
 dp[n+1][orr+1]
    INITIALIZE(dp to 0)

    dp[0][0]=1;

     FOR i 1 to n:
     FOR j 0 to orr:

dp[i][j]=dp[i-1][j]+dp[i-1][orr^a[i]]

    PRINT(dp[n][k])
END:
```

CODE

```
int main()
{
    lli t;
    cout<<"ENTER TOTAL NUMBER OF TEST CASES:"<<endl;
    cin>>t;
    cout<<"ENTER THE TEST CASES:"<<endl;
    while(t--)
    {
        lli n,k;
        cout<<"ENTER LENGTH OF ARRAY AND THE VALUE OF K:"<<endl;
        cin>>n>>k;
        lli K=k;
        lli a[n+1];
        lli orr=0;
        cout<<"ENTER ARRAY ELEMENTS:"<<endl;
        For(i,1,n+1){cin>>a[i];orr=orr|a[i];}
        if(orr<k)
        {
            cout<<"0"<<endl;continue;
        }
        k=orr;
        lli** dp=new lli*[n+1];
        For(i,0,n+1)
        {
            dp[i]=new lli[k+1]();
        }
        dp[0][0]=1;
        For(i,1,n+1)
        {
            For(j,0,k+1)
            {
                dp[i][j]=dp[i-1][j]+dp[i-1][j^a[i]];
            }
        }
        cout<<"Answer = "<<dp[n][K]<<endl<<endl;
    }
}
```

COMPLEXITY ANALYSIS

Here, we are going through a matrix of order n times r, where

n = number of elements in array

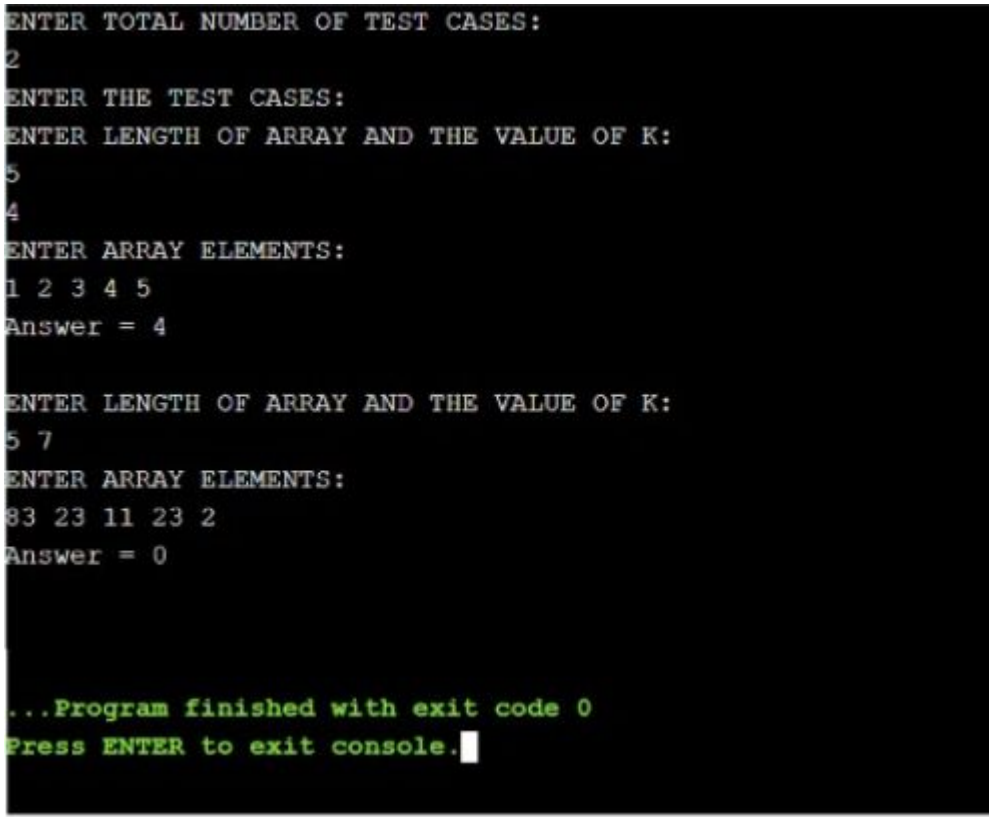
r = OR of all elements in array

Hence time complexity comes out to be **O(nr)**

And for space complexity, since we are making a matrix of order n times r,

Space complexity is **O(nr)**

OUTPUT


ENTER TOTAL NUMBER OF TEST CASES:
2
ENTER THE TEST CASES:
ENTER LENGTH OF ARRAY AND THE VALUE OF K:
5
4
ENTER ARRAY ELEMENTS:
1 2 3 4 5
Answer = 4
ENTER LENGTH OF ARRAY AND THE VALUE OF K:
5 7
ENTER ARRAY ELEMENTS:
83 23 11 23 2
Answer = 0
...Program finished with exit code 0
Press ENTER to exit console.

THANK YOU