SPOJ

CODEIT02 - PICK UP DROP ESCAPE

*no tags*

**PICK UP DROP ESCAPE**

You are given with an array of n numbers. You must pick K elements from the array such that XOR of all the chosen elements is maximum.

http://en.wikipedia.org/wiki/Bitwise\_operation#XOR

**Input Specification:**

The first line consists of an integer t representing the number of test cases. For each test case the first line consists of two numbers n and K denoting the number of elements in the array and the number of elements to be chosen respectively. Then in the next n lines, the elements in the array are given.

**Output Specification:**

For each test case print a line containing one integer denoting the maximum XOR value of the chosen K elements.

**Input Constraints:**

1<=t<=100

1<=n<=20

1<=k<=n

1<=element value<=10000

**Sample Input:**

2

5 3

1

2

3

4

5

5 3

3

4

5

7

4

**Sample Output:**

7

7

GEEKS FOR GEEKS

# Find maximum xor of k elements in an array

Given an array **arr[]** of **N** integers and an integer **K**. The task is to find the maximum xor subset of size **K** of the given array.

**Examples:**

***Input:****arr[] = {2, 5, 4, 1, 3, 7, 6, 8}, K = 3****Output:****15  
We obtain 15 by selecting 4, 5, 6, 8*

***Input:****arr[] = {3, 4, 7, 7, 9}, K = 3****Output:****14*

GEEKS FOR GEEKS SOLUTION:

* BITMASKING(NAIVE):

#include <bits/stdc++.h>

using namespace std;

// Function to return the maximum xor for a

// subset of size k from the given array

int Max\_Xor(int arr[], int n, int k)

{

    // Initialize result

    int maxXor = INT\_MIN;

    // Traverse all subsets of the array

    for (int i = 0; i < (1 << n); i++) {

        // \_\_builtin\_popcount() returns the number

        // of sets bits in an integer

        if (\_\_builtin\_popcount(i) == k) {

            // Initialize current xor as 0

            int cur\_xor = 0;

            for (int j = 0; j < n; j++) {

                // If jth bit is set in i then include

                // jth element in the current xor

                if (i & (1 << j))

                    cur\_xor = cur\_xor ^ arr[j];

            }

            // Update maximum xor so far

            maxXor = max(maxXor, cur\_xor);

        }

    }

    return maxXor;

}

// Driver code

int main()

{

    int arr[] = { 2, 5, 4, 1, 3, 7, 6, 8 };

    int n = sizeof(arr) / sizeof(int);

    int k = 3;

    cout << Max\_Xor(arr, n, k);

    return 0;

}

* DP+BITMASKING:

#include <bits/stdc++.h>

using namespace std;

#define MAX 10000

#define MAX\_ELEMENT 50

int dp[MAX\_ELEMENT][MAX\_ELEMENT][MAX];

// Function to return the maximum xor for a

// subset of size j from the given array

int Max\_Xor(int arr[], int i, int j, int mask, int n)

{

    if (i >= n) {

        // If the subset is complete then return

        // the xor value of the selected elements

        if (j == 0)

            return mask;

        else

            return 0;

    }

    // Return if already calculated for some

    // mask and j at the i'th index

    if (dp[i][j][mask] != -1)

        return dp[i][j][mask];

    // Initialize answer to 0

    int ans = 0;

    // If we can still include elements in our subset

    // include the i'th element

    if (j > 0)

        ans = Max\_Xor(arr, i + 1, j - 1, mask ^ arr[i], n);

    // Exclude the i'th element

    // ans store the max of both operations

    ans = max(ans, Max\_Xor(arr, i + 1, j, mask, n));

    return dp[i][j][mask] = ans;

}

// Driver code

int main()

{

    int arr[] = { 2, 5, 4, 1, 3, 7, 6, 8 };

    int n = sizeof(arr) / sizeof(int);

    int k = 3;

    memset(dp, -1, sizeof(dp));

    cout << Max\_Xor(arr, 0, k, 0, n);

    return 0;

}

* DP+BITMASKING 2:

#include<bits/stdc++.h>

using namespace std;

int dp[21][21][10000];

int ans(int a[],int n,int k,int res)

{

if(k==0)

return res;

if(n==0)

return 0;

if(dp[n][k][res]!=-1)

return dp[n][k][res];

return dp[n][k][res]=max(ans(a,n-1,k-1,a[n-1]^res),ans(a,n-1,k,res));

}

int main()

{

ios\_base::sync\_with\_stdio(false);

cin.tie(NULL);

cout.tie(NULL);

int t;

cin>>t;

while(t--)

{

memset(dp,-1,sizeof(dp));

int n,k;

cin>>n>>k;

int a[n],i;

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

cin>>a[i];

cout<<ans(a,n,k,0)<<"\n";

}

}