Bit Magic

Maximum AND Value| Explanation

Given an array arr[] of N positive elements. The task is to find the Maximum AND Value generated by any pair of the element from the array.

Note: AND is bitwise '&' operator.

Examples:

> **Input**: a[] = {4, 8, 12, 16}
> **Output**: 8
> The pairs 8 and 12 gives us the '&' value as 12.
>
> **Input**: a[] = {4, 8, 16, 2}
> **Output**: 0

A **naive** approach is to iterate for all the pairs using two for loops and check for the maximum '&' value of any pair.

Note: This approach will not fit in the given time limit since the complexity of the above method jis O(N^2).

```
int findMaxium(int a[], int n)
{
    int maxi = 0;
    for(int i = 0;i<n;i++)
    {
        for(int j = i+1;j<n;j++)
            maxi = max(maxi, a[i] & a[j]);
    }

    return maxi;
}
```

Efficient Approach: An efficient approach will be to look at this problem bitwise. Since we need to find the maximum '&' value. The first thing that strikes our mind is that the answer should have its LSB as far as possible. So, if two elements are considered as a pair, then their LSB should be set to as much left as possible. Let's take an example to understand this. Consider three elements {10, 8, 2}, so to get a maximum '&' value we need to take those elements whose LSB is as far as possible. In the given example, we can clearly see that 10(1010) and 8(1000) have their 4th-bit from the left set and hence will maximize the answer. Taking 2 and 10 will give our 2nd bit to be set which won't maximize our answer.

So since the constraints permit till 10^4, hence the '&' value will also be less than that. 10^4 will range in 2^0 to 2^14, which means we need to start our checking from the 15th bit. Initially we loop from 15 to 0 and check for the count of numbers whose that particular bit is set. Once we get the count more than 2, the answer will have that bit set, and for the next bit from the left to be set we need to check for both the previous all bits and the current i-th bit. The previous bits can be added to the current bit using a '|' operator. In this way, we get all the positions of the bit which are set, which can be easily represented as a number.

Note: We have started checking from bits 31 so that if the constraints were high, it can easily fit in.

```
// Utility function to check number of elements
// having set msb as of pattern
int checkBit(int pattern, int arr[], int n)
{
    int count = 0;
    for (int i = 0; i < n; i++)
        if ((pattern & arr[i]) == pattern)
            count++;
    return count;
}

// Function for finding maximum and value pair
int maxAND (int arr[], int n)
{
    int res = 0, count;

    // iterate over total of 30bits from msb to lsb
    for (int bit = 31; bit >= 0; bit--)
    {
        // find the count of element having set  msb
        count = checkBit(res | (1 << bit),arr,n);

        // if count >= 2 set particular bit in result
        if ( count >= 2 )
            res |= (1 << bit);
    }

   return res;
}
```