# 1829. Maximum XOR for Each Query

You are given a **sorted** array `nums` of `n` non-negative integers and an integer `maximumBit`. You want to perform the following query `n` **times**:

1. Find a non-negative integer `k < 2maximumBit` such that `nums[0] XOR nums[1] XOR ... XOR nums[nums.length-1] XOR k` is **maximized**. `k` is the answer to the `ith` query.
2. Remove the **last** element from the current array `nums`.

Return *an array* `answer`, *where* `answer[i]` *is the answer to the* `ith` *query*.

**Example 1:**

```
Input: nums = [0,1,1,3], maximumBit = 2
Output: [0,3,2,3]
Explanation: The queries are answered as follows:
1st query: nums = [0,1,1,3], k = 0 since 0 XOR 1 XOR 1 XOR 3 XOR 0 = 3.
2nd query: nums = [0,1,1], k = 3 since 0 XOR 1 XOR 1 XOR 3 = 3.
3rd query: nums = [0,1], k = 2 since 0 XOR 1 XOR 2 = 3.
4th query: nums = [0], k = 3 since 0 XOR 3 = 3.
```

**Example 2:**

```
Input: nums = [2,3,4,7], maximumBit = 3
Output: [5,2,6,5]
Explanation: The queries are answered as follows:
1st query: nums = [2,3,4,7], k = 5 since 2 XOR 3 XOR 4 XOR 7 XOR 5 = 7.
2nd query: nums = [2,3,4], k = 2 since 2 XOR 3 XOR 4 XOR 2 = 7.
3rd query: nums = [2,3], k = 6 since 2 XOR 3 XOR 6 = 7.
4th query: nums = [2], k = 5 since 2 XOR 5 = 7.
```

**Example 3:**

```
Input: nums = [0,1,2,2,5,7], maximumBit = 3
Output: [4,3,6,4,6,7]
```

**Constraints:**

- `nums.length == n`
- `1 <= n <= 10^5`
- `1 <= maximumBit <= 20`
- `0 <= nums[i] < 2^maximumBit`

# 1829. Maximum XOR for Each Query

```
/*
   Prefix xor+Math
   Time complexity: O(2n)=O(n)
   Space complexity: O(n)
*/
class Solution {
   public:
      std::vector<int> getMaximumXor(std::vector<int>& nums, int maximumBit){
         int n=nums.size();
         std::vector<int> prefix_xor(n);
         prefix_xor[0]=nums[0];
         for(int i=1;i<n;++i) prefix_xor[i]=prefix_xor[i-1]^nums[i];
         std::vector<int> ans;
         int k=(1<<maximumBit)-1;
         for(int i=n-1;i>=0;--i) ans.push_back(prefix_xor[i]^k);
         return ans;
      }
};
```

# 1829. Maximum XOR for Each Query

```
/*
   Prefix xor space optimization+Math
   Time complexity: O(2n)=O(n)
   Space complexity: O(1)
*/
class Solution{
   public:
      std::vector<int> getMaximumXor(std::vector<int>& nums, int maximumBit){
         int n=nums.size();
         int xor_prefix=0;
         for(int i=0;i<n;++i) xor_prefix^=nums[i];
         std::vector<int> ans(n);
         int k=(1<<maximumBit)-1;
         for(int i=0;i<n;++i){
            ans[i]=xor_prefix^k;
            xor_prefix^=nums[n-1-i];
         }
         return ans;
      }
};
```