# Problem 421: Maximum XOR of Two Numbers in an Array

## Problem Information

**Difficulty:** Medium
**Acceptance Rate:** 0.00%
**Paid Only:** No

## Problem Description

Given an integer array

nums

, return

the maximum result of

nums[i] XOR nums[j]

, where

0 <= i <= j < n

.

Example 1:

Input:

nums = [3,10,5,25,2,8]

Output:

28

Explanation:

The maximum result is 5 XOR 25 = 28.

Example 2:

Input:

nums = [14,70,53,83,49,91,36,80,92,51,66,70]

Output:

127

Constraints:

1 <= nums.length <= 2 * 10

5

0 <= nums[i] <= 2

31

- 1

## Code Snippets

**C++:**

```
class Solution {
public:
int findMaximumXOR(vector<int>& nums) {

}
};
```

**Java:**

```
class Solution {
public int findMaximumXOR(int[] nums) {

}
}
```

**Python3:**

```
class Solution:
def findMaximumXOR(self, nums: List[int]) -> int:
```

**Python:**

```
class Solution(object):
def findMaximumXOR(self, nums):
"""
:type nums: List[int]
:rtype: int
"""
```

**JavaScript:**

```
/**
* @param {number[]} nums
* @return {number}
*/
var findMaximumXOR = function(nums) {

};
```

**TypeScript:**

```
function findMaximumXOR(nums: number[]): number {

};
```

**C#:**

```
public class Solution {
public int FindMaximumXOR(int[] nums) {

}
}
```

**C:**

```
int findMaximumXOR(int* nums, int numsSize){

}
```

**Go:**

```
func findMaximumXOR(nums []int) int {

}
```

**Kotlin:**

```
class Solution {
fun findMaximumXOR(nums: IntArray): Int {

}
}
```

**Swift:**

```
class Solution {
func findMaximumXOR(_ nums: [Int]) -> Int {

}
}
```

**Rust:**

```
impl Solution {
pub fn find_maximum_xor(nums: Vec<i32>) -> i32 {

}
}
```

**Ruby:**

```
# @param {Integer[]} nums
# @return {Integer}
def find_maximum_xor(nums)

end
```

**PHP:**

```
class Solution {

/**
* @param Integer[] $nums
* @return Integer
*/
function findMaximumXOR($nums) {

}
}
```

**Dart:**

```
class Solution {
int findMaximumXOR(List<int> nums) {

}
}
```

**Scala:**

```
object Solution {
def findMaximumXOR(nums: Array[Int]): Int = {

}
}
```

**Elixir:**

```
defmodule Solution do
@spec find_maximum_xor(nums :: [integer]) :: integer
def find_maximum_xor(nums) do

end
end
```

**Erlang:**

```
-spec find_maximum_xor(Nums :: [integer()]) -> integer().
find_maximum_xor(Nums) ->
.
```

### Racket:

```
(define/contract (find-maximum-xor nums)
(-> (listof exact-integer?) exact-integer?)

)
```

## Solutions

### C++ Solution:

```
/*
* Problem: Maximum XOR of Two Numbers in an Array
* Difficulty: Medium
* Tags: array, hash
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

class Solution {
public:
int findMaximumXOR(vector<int>& nums) {

}
};
```

### Java Solution:

```
/**
* Problem: Maximum XOR of Two Numbers in an Array
* Difficulty: Medium
* Tags: array, hash
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

class Solution {
```

```
public int findMaximumXOR(int[] nums) {

}
}
```

### Python3 Solution:

```
"""
Problem: Maximum XOR of Two Numbers in an Array
Difficulty: Medium
Tags: array, hash

Approach: Use two pointers or sliding window technique
Time Complexity: O(n) or O(n log n)
Space Complexity: O(n) for hash map
"""

class Solution:
def findMaximumXOR(self, nums: List[int]) -> int:
# TODO: Implement optimized solution
pass
```

### Python Solution:

```
class Solution(object):
def findMaximumXOR(self, nums):
"""
:type nums: List[int]
:rtype: int
"""
```

### JavaScript Solution:

```
/**
* Problem: Maximum XOR of Two Numbers in an Array
* Difficulty: Medium
* Tags: array, hash
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
```

```
*/

/**
* @param {number[]} nums
* @return {number}
*/
var findMaximumXOR = function(nums) {

};
```

### TypeScript Solution:

```
/**
* Problem: Maximum XOR of Two Numbers in an Array
* Difficulty: Medium
* Tags: array, hash
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

function findMaximumXOR(nums: number[]): number {

};
```

### C# Solution:

```
/*
* Problem: Maximum XOR of Two Numbers in an Array
* Difficulty: Medium
* Tags: array, hash
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

public class Solution {
public int FindMaximumXOR(int[] nums) {
```

```
}
}
```

### C Solution:

```
/*
* Problem: Maximum XOR of Two Numbers in an Array
* Difficulty: Medium
* Tags: array, hash
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

int findMaximumXOR(int* nums, int numsSize){

}
```

### Go Solution:

```
// Problem: Maximum XOR of Two Numbers in an Array
// Difficulty: Medium
// Tags: array, hash
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) for hash map

func findMaximumXOR(nums []int) int {

}
```

### Kotlin Solution:

```
class Solution {
fun findMaximumXOR(nums: IntArray): Int {

}
}
```

### Swift Solution:

```
class Solution {
func findMaximumXOR(_ nums: [Int]) -> Int {

}
}
```

### Rust Solution:

```
// Problem: Maximum XOR of Two Numbers in an Array
// Difficulty: Medium
// Tags: array, hash
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) for hash map

impl Solution {
pub fn find_maximum_xor(nums: Vec<i32>) -> i32 {

}
}
```

### Ruby Solution:

```
# @param {Integer[]} nums
# @return {Integer}
def find_maximum_xor(nums)

end
```

### PHP Solution:

```
class Solution {

/**
* @param Integer[] $nums
* @return Integer
*/
function findMaximumXOR($nums) {
```

```
}
}
```

### Dart Solution:

```
class Solution {
int findMaximumXOR(List<int> nums) {

}
}
```

### Scala Solution:

```
object Solution {
def findMaximumXOR(nums: Array[Int]): Int = {

}
}
```

### Elixir Solution:

```
defmodule Solution do
@spec find_maximum_xor(nums :: [integer]) :: integer
def find_maximum_xor(nums) do

end
end
```

### Erlang Solution:

```
-spec find_maximum_xor(Nums :: [integer()]) -> integer().
find_maximum_xor(Nums) ->
.
```

### Racket Solution:

```
(define/contract (find-maximum-xor nums)
(-> (listof exact-integer?) exact-integer?)

)
```