# Problem 1310: XOR Queries of a Subarray

## Problem Information

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

## Problem Description

You are given an array

arr

of positive integers. You are also given the array

queries

where

queries[i] = [left

i,

right

i

]

.

For each query

i

compute the

XOR

of elements from

left

i

to

right

i

(that is,

arr[left

i

] XOR arr[left

i

+ 1] XOR ... XOR arr[right

i

]

).

Return an array

answer

where

answer[i]

is the answer to the

i

th

query.

Example 1:

Input:

arr = [1,3,4,8], queries = [[0,1],[1,2],[0,3],[3,3]]

Output:

[2,7,14,8]

Explanation:

The binary representation of the elements in the array are: 1 = 0001 3 = 0011 4 = 0100 8 = 1000 The XOR values for queries are: [0,1] = 1 xor 3 = 2 [1,2] = 3 xor 4 = 7 [0,3] = 1 xor 3 xor 4 xor 8 = 14 [3,3] = 8

Example 2:

Input:

arr = [4,8,2,10], queries = [[2,3],[1,3],[0,0],[0,3]]

Output:

[8,0,4,4]

Constraints:

1 <= arr.length, queries.length <= 3 * 10

4

1 <= arr[i] <= 10

9

queries[i].length == 2

0 <= left

i

<= right

i

< arr.length

## Code Snippets

**C++:**

```
class Solution {
public:
vector<int> xorQueries(vector<int>& arr, vector<vector<int>>& queries) {

}
};
```

**Java:**

```
class Solution {
public int[] xorQueries(int[] arr, int[][] queries) {

}
}
```

**Python3:**

```
class Solution:
def xorQueries(self, arr: List[int], queries: List[List[int]]) -> List[int]:
```

**Python:**

```
class Solution(object):
def xorQueries(self, arr, queries):
"""
:type arr: List[int]
:type queries: List[List[int]]
:rtype: List[int]
"""
```

**JavaScript:**

```
/**
* @param {number[]} arr
* @param {number[][]} queries
* @return {number[]}
*/
var xorQueries = function(arr, queries) {

};
```

**TypeScript:**

```
function xorQueries(arr: number[], queries: number[][]): number[] {

};
```

**C#:**

```
public class Solution {
public int[] XorQueries(int[] arr, int[][] queries) {

}
}
```

**C:**

```
/**
* Note: The returned array must be malloced, assume caller calls free().
*/
```

```
int* xorQueries(int* arr, int arrSize, int** queries, int queriesSize, int*
queriesColSize, int* returnSize) {

}
```

**Go:**

```
func xorQueries(arr []int, queries [][]int) []int {

}
```

**Kotlin:**

```
class Solution {
fun xorQueries(arr: IntArray, queries: Array<IntArray>): IntArray {

}
}
```

**Swift:**

```
class Solution {
func xorQueries(_ arr: [Int], _ queries: [[Int]]) -> [Int] {

}
}
```

**Rust:**

```
impl Solution {
pub fn xor_queries(arr: Vec<i32>, queries: Vec<Vec<i32>>) -> Vec<i32> {

}
}
```

**Ruby:**

```
# @param {Integer[]} arr
# @param {Integer[][]} queries
# @return {Integer[]}
def xor_queries(arr, queries)
```

```
end
```

**PHP:**

```
class Solution {

/**
* @param Integer[] $arr
* @param Integer[][] $queries
* @return Integer[]
*/
function xorQueries($arr, $queries) {

}
}
```

**Dart:**

```
class Solution {
List<int> xorQueries(List<int> arr, List<List<int>> queries) {

}
}
```

**Scala:**

```
object Solution {
def xorQueries(arr: Array[Int], queries: Array[Array[Int]]): Array[Int] = {

}
}
```

**Elixir:**

```
defmodule Solution do
@spec xor_queries(arr :: [integer], queries :: [[integer]]) :: [integer]
def xor_queries(arr, queries) do

end
end
```

**Erlang:**

```
-spec xor_queries(Arr :: [integer()], Queries :: [[integer()]]) ->
[integer()].
xor_queries(Arr, Queries) ->
.
```

**Racket:**

```
(define/contract (xor-queries arr queries)
(-> (listof exact-integer?) (listof (listof exact-integer?)) (listof
exact-integer?))
)
```

## Solutions

### C++ Solution:

```
/*
* Problem: XOR Queries of a Subarray
* Difficulty: Medium
* Tags: array
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

class Solution {
public:
vector<int> xorQueries(vector<int>& arr, vector<vector<int>>& queries) {

}
};
```

### Java Solution:

```
/**
* Problem: XOR Queries of a Subarray
* Difficulty: Medium
* Tags: array
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

class Solution {
public int[] xorQueries(int[] arr, int[][] queries) {

}
}
```

### Python3 Solution:

```
"""
Problem: XOR Queries of a Subarray
Difficulty: Medium
Tags: array

Approach: Use two pointers or sliding window technique
Time Complexity: O(n) or O(n log n)
Space Complexity: O(1) to O(n) depending on approach
"""

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

### Python Solution:

```
class Solution(object):
def xorQueries(self, arr, queries):
"""
:type arr: List[int]
:type queries: List[List[int]]
:rtype: List[int]
"""
```

### JavaScript Solution:

```
/**
* Problem: XOR Queries of a Subarray
* Difficulty: Medium
* Tags: array
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

/**
* @param {number[]} arr
* @param {number[][]} queries
* @return {number[]}
*/
var xorQueries = function(arr, queries) {

};
```

### TypeScript Solution:

```
/**
* Problem: XOR Queries of a Subarray
* Difficulty: Medium
* Tags: array
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

function xorQueries(arr: number[], queries: number[][]): number[] {

};
```

### C# Solution:

```
/*
* Problem: XOR Queries of a Subarray
* Difficulty: Medium
* Tags: array
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

public class Solution {
public int[] XorQueries(int[] arr, int[][] queries) {

}
}
```

### C Solution:

```
/*
* Problem: XOR Queries of a Subarray
* Difficulty: Medium
* Tags: array
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

/**
* Note: The returned array must be malloced, assume caller calls free().
*/
int* xorQueries(int* arr, int arrSize, int** queries, int queriesSize, int*
queriesColSize, int* returnSize) {

}
```

### Go Solution:

```
// Problem: XOR Queries of a Subarray
// Difficulty: Medium
// Tags: array
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

func xorQueries(arr []int, queries [][]int) []int {
}
```

**Kotlin Solution:**

```
class Solution {
fun xorQueries(arr: IntArray, queries: Array<IntArray>): IntArray {

}
}
```

**Swift Solution:**

```
class Solution {
func xorQueries(_ arr: [Int], _ queries: [[Int]]) -> [Int] {

}
}
```

**Rust Solution:**

```
// Problem: XOR Queries of a Subarray
// Difficulty: Medium
// Tags: array
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

impl Solution {
pub fn xor_queries(arr: Vec<i32>, queries: Vec<Vec<i32>>) -> Vec<i32> {

}
}
```

**Ruby Solution:**

```
# @param {Integer[]} arr
# @param {Integer[][]} queries
# @return {Integer[]}
def xor_queries(arr, queries)
```

```



end
```

### PHP Solution:

```
class Solution {

/**
* @param Integer[] $arr
* @param Integer[][] $queries
* @return Integer[]
*/
function xorQueries($arr, $queries) {

}
}
```

### Dart Solution:

```
class Solution {
List<int> xorQueries(List<int> arr, List<List<int>> queries) {

}
}
```

### Scala Solution:

```
object Solution {
def xorQueries(arr: Array[Int], queries: Array[Array[Int]]): Array[Int] = {

}
}
```

### Elixir Solution:

```
defmodule Solution do
@spec xor_queries(arr :: [integer], queries :: [[integer]]) :: [integer]
def xor_queries(arr, queries) do

end
end
```

## Erlang Solution:

```
-spec xor_queries(Arr :: [integer()], Queries :: [[integer()]]) ->
[integer()].
xor_queries(Arr, Queries) ->
.
```

## Racket Solution:

```
(define/contract (xor-queries arr queries)
(-> (listof exact-integer?) (listof (listof exact-integer?)) (listof
exact-integer?))
)
```