Minimum Array End

solved by	Senan
Platform	LeetCode
difficulty	Medium
# Serial	3133
tags	Bit Manipulation
language	C++
solved on	@09/11/2024
link	https://leetcode.com/problems/minimum-array-end/
Completion	☑

Intuition

To satisfy the problem requirements, we need to create an array `nums` of size `n` where each subsequent element is greater than the previous one. Additionally, the bitwise AND of all elements in `nums` must equal `x`. This requires careful selection of each element's bits so that when combined, they still yield `x` as the bitwise AND result. The main challenge lies in constructing the array with a minimum last element while ensuring the bitwise condition holds.

Approach

1. **Initialize with x**: Start with `result` as `x`, which will hold the minimum possible value of `nums[n-1]`.
2. **Bitwise Manipulation**: For each bit position, we check if it contributes to the bitwise AND result. If a bit is unset in `x`, it has no impact on the AND outcome, allowing us to set this bit conditionally for optimization.
3. **Update Strategy**: Use a bitwise mask that shifts to the left by one position in each iteration. If `x` has a `0` at a certain bit position, we may set this bit in `result` based on the parity of `n`, helping achieve a minimal increase while preserving the AND result as `x`.
4. **Halve n**: For each checked bit, halve `n`, focusing only on the remaining bits that need to be set or adjusted in `result` until `n` becomes 1.

The result will be the smallest possible last element that satisfies both the size and bitwise conditions.

Complexity

Time Complexity:

The approach runs in **O(log n)** since we halve `n` in each iteration, making it logarithmic in terms of bit positions.

Space Complexity:

The space complexity is **O(1)** as we only use a few variables.

Code

```cpp
class Solution {
public:
    long long minEnd(int n, int x) {
        long long result = x, mask;

        for (mask = 1; n > 1; mask <<= 1) {
            if ((mask & x) == 0) {
                result |= (n & 1) * mask;
                n >>= 1;
            }
        }
        return result;
    }
};
```