Circle of Friends Problem Statement

Problem Statement

Given a circular array of N positive integers, count the number of ways to partition the circular array into subarrays where the values in each subarray bitwise AND to a nonzero value. $N \leq 2 \cdot 10^5$, $0 \leq a_i < 2^{60}$.

Solving a Simpler Problem

Problems on circular arrays tend to be fairly similar to the equivalent versions on regular arrays, so we'll start by solving the following simpler problem instead:

Problem Statement

Given an array of N positive integers, count the number of ways to partition the array into subarrays where the values in each subarray bitwise AND to a nonzero value. $N \leq 2 \cdot 10^5$, $0 \leq a_i < 2^{60}$.

A Dynamic Programming Approach

Let's define $g(i, j)$ to be 1 if the bitwise AND of $a_i, \ldots, a_j$ is nonzero, and zero otherwise.

Furthermore, let's define $f(i)$ to be the number of ways to partition the first i integers such that the bitwise AND condition is satisfied, so the answer to the problem is $f(N)$.

We can therefore set up a dynamic programming recurrence where $f(0) = 1$ and for $i > 0$, $f(i) = \sum_j g(j, i) \cdot f(j - 1)$.

Optimizing the Dynamic Programming Approach

The most naive implementation of the above takes $\mathcal{O}(N^3)$ time, as there are $\mathcal{O}(N)$ states, $\mathcal{O}(N)$ transitions per state, and it naively takes $\mathcal{O}(N)$ time to compute $g(i, j)$.

We first note that if $g(j, i)$ is equal to 1 for some $j < i$, then necessarily $g(j + 1, i)$ is also equal to 1. Therefore, if we precompute a sparse table of the bitwise ANDs of all subarrays with lengths equal to some power of two, we can optimize one $\mathcal{O}(N)$ factor down to $\mathcal{O}(\log N)$ to compute the smallest j where $g(j, i)$ is equal to 1.

There are still $\mathcal{O}(N)$ transitions per state, but because the j values are contiguous, if we maintain a prefix sum for f, we can optimize that down to $\mathcal{O}(1)$ time.

This allows us to solve this problem on a regular array in $\mathcal{O}(N \log N)$.

Returning to the Original Problem

We cannot directly apply the above solution back to the original problem - since there are $\mathcal{O}(N)$ starting positions to orient the circular array, a direct application would result in an $\mathcal{O}(N^2 \log N)$ solution.

The above DP tells us how many partitions exist when a_1 is the "first" element in its group. If we know that $a_N \,\&\, a_1 = a_1$, then the above DP actually tells how many partitions exist when a_N and a_1 are in the same group - namely, $f(N-1)$ of them. This applies for any suffix of elements whose bitwise AND shares all the bits of a_1. Once some suffix no longer meets this criterion, then we can set a_1 equal to the bitwise AND of itself and the given suffix, remove that suffix from the end of the list, and repeat the above DP with the new list.

From the above logic, this means we can run the above DP at most 60 times, since with every suffix removal, at least one bit must be turned off. This yields a solution which is $\mathcal{O}(60N \log N)$.