Notes on Perfect Squares & k-nary XOR

1. Definition of Perfect Square

A number is a perfect square if it can be written as k^2 for some integer k.
Examples: 0, 1, 4, 9, 16, 25, …

2. Prime Factorization Property

Every integer can be expressed uniquely as a product of prime factors: n = p1^a1 × p2^a2 × … × pk^ak. n is a perfect square if and only if all exponents a[i] are even.

Examples:

- $36 = 2^2 \times 3^2 \rightarrow$ perfect square.
- $18 = 2^1 \times 3^2 \rightarrow$ not a square. (exponent of 2 is odd)

3. Product Check

If we multiply several numbers together, the product is a perfect square iff the combined exponent count of every prime factor is even. In competitive programming, this is often checked using XOR of exponents mod 2 (parity check).

4. Special Cases

- 0 is a perfect square ($0^2 = 0$).
- Negative numbers are not perfect squares in integers.

5. k-nary XOR (Generalized XOR)

Normally XOR is defined in base 2 (binary). But we can generalize XOR into base k. For k-nary XOR, addition is done modulo k without carry. This is useful when tracking exponents modulo k instead of just modulo 2. For example, to check for perfect cubes, we would track exponents modulo 3.

Examples:

- Binary XOR (k=2): $1 \oplus 1 = 0$, $1 \oplus 0 = 1$.
- Ternary XOR (k=3): $2 \oplus 2 = 1$ (since (2+2) mod 3 = 1) , 2 + 2 + 2 = ((2 + 2) mod 3 + 2) mod 3 = 0 (Freq of 2 is 3 times.)
- General rule: $(a \oplus b) = (a + b) \bmod k$.

6. XOR Hash Trick for Perfect Squares and k-nary Checks

1. **Perfect Square Check**
    - A number is a **perfect square** if every prime factor appears an **even number of times**.
    - Example:
        - 36 = 2^2 * 3^2 → perfect square
        - 18 = 2 * 3^2 → not a perfect square (2 appears odd times)

2. **XOR Hash Trick**
   - Assign a **random hash value** to each prime number.
   - For an array A[1..n], compute **prefix XOR** of the hashes of prime factors modulo 2:
     prefix[i] = hash_xor(A[1..i])
   - To check if the product of a subarray A[l..r] is a perfect square:
     subarray_xor = prefix[r] ^ prefix[l-1]
     if(subarray_xor == 0) => perfect square
   - XOR cancels even occurrences of primes, leaving 0 if all are even.
3. **Handling Zero and Negative Numbers**
   - **Zero**: Any subarray containing 0 is a perfect square.
   - **Negative numbers**: Track the count of negative signs. Product is square if count of negatives is even.
4. **k-nary Generalization**
   - For checking if **each prime occurs a multiple of k times**, define **k-nary XOR**:
     a ⊕_k b = (a + b) % k
   - Compute **prefix k-nary sum** for the array.
   - Subarray check:
     subarray_k_xor = (prefix[r] - prefix[l-1] + k) % k
     if(subarray_k_xor == 0) => all primes occur multiple of k times
   - Example:
     - Binary XOR (k=2) → checks even occurrences.
     - Mod 3 addition (k=3) → checks multiple-of-3 occurrences.
5. **Summary Table**

Check Type	Operation	Condition for Subarray Product
Perfect square	XOR (mod 2)	subarray_xor == 0
Multiple-of-3	Add mod 3	subarray_mod3 == 0
General k	Add mod k	subarray_modk == 0

This technique allows fast checking of multiplicities of prime factors in subarrays without computing large products.