Problem Pattern

1. Which representation of the number is more helpful to you? Binary or Decimal ?
2. Can you take advantage of the fact that all numbers are <= 2^30 - 1 i.e. are 30 bit numbers.
3. When is the XOR of two numbers maximum? When the MSB of two numbers in binary form are opposite for the same position. So increasing the number of different MSBs will increase the XOR. [MSB = Most significant Bit]
4. Can you think of a data structure where you can store all the numbers from MSB to LSB and then check for dissimilar and similar bits for all of them simultaneously?

Problem Approach

1. A trie is the best data structure for this question. Implement a trie class and initialize its object.
2. Add the 30 bit binary representation for all the numbers in the trie.
3. Now for each number , look for the number in the trie that will give you maximum XOR.
4. You can do it since all numbers are stored MSB to LSB. For a given bit, check if its inverse exists as a child of this node. If it does, go to the inverse , since opposite bits XOR to 1 increasing the final XOR value. If no such child exists, go to the child with the current bit.(Inverse of 0 is 1 and inverse of 1 is 0).
5. Find the maximum xor by obtaining the XOR of the two numbers.

Problem Pseudocode

```
maximumXor(nums) :

       T = Trie()

       for num in nums :

              T.add(Binary(num)) # add 30 bit binary form of the number

       Ans = 0

for num in nums :

              X = find(T,num)

              Ans = max(Ans,num ^ X)

       Return Ans


find(T,num) :

       Val = str()

       For bit in num :

If inverse(bit) in T.children :

       T = T.children[inverse(bit)]

       Val += inverse(bit)

Else :

       T = T.children[bit]

       Val += bit


      Return Decimal(Val) # convert the number to decimal and return it
```