Trie Interview

We can construct two arrays lbest and rbest for the left hand side of the expression and for the right hand side of the expression.
To calculate lbest[i] we will iterate m from i to i such that (arr[m] ^ arr[m+1] ^ arr[m+2] .... arr[i]) is maximised. Then lbest[i] = Max(lbest[i-1], val). Similar can be done for rbest. Now we will calculate val. Let C[i] = (arr[1] ^ arr[2] ^ arr[3] .... arr[i]). For some j <= i, C[j-1] ^ C[i] = (arr[j] ^ arr[j+1] ^ arr[j+2] ... arr[i]). We can now say that lbest[i] = max(lbest[i−1],val) where val = maximum of (C[j−1]C[i]) for j = 1 to i. Now using Tries we can easily calculate ‘val’ in (log arr(max)). We can store the bits of each number in the nodes of trie and iterate through the trie such that we can get the maximum xor possible.

```
import java.io.*;
import java.util.*;

public class interview {

    public static class trieNode {
        trieNode left;
        trieNode right;
    }

    public static void insert(int n, trieNode head) {
        trieNode curr = head;
        for (int i = 31; i >= 0; i--) {
            int bit = (n >> i) & 1;
            if (bit == 0) {
                if (curr.left == null) {
                    curr.left = new trieNode();
                }
                curr = curr.left;
            } else {
                if (curr.right == null) {
                    curr.right = new trieNode();
                }
                curr = curr.right;
            }
        }
    }

    public static int findMaxXorPair(trieNode head, int val) {
        trieNode curr = head;
        int value = val;
        int curr_xor = 0;
        for (int j = 31; j >= 0; j--) {
            int b = (value >> j) & 1;
            if (b == 0) {
                if (curr.right != null) {
```

```
                curr = curr.right;
                curr_xor += (int) Math.pow(2, j);
            } else {
                curr = curr.left;
            }
        } else {
            if (curr.left != null) {
                curr = curr.left;
                curr_xor += (int) Math.pow(2, j);
            } else {
                curr = curr.right;
            }
        }
    }

    return curr_xor;
}

static class Reader {
    final private int BUFFER_SIZE = 1 << 16;
    private DataInputStream din;
    private byte[] buffer;
    private int bufferPointer, bytesRead;

    public Reader() {
        din = new DataInputStream(System.in);
        buffer = new byte[BUFFER_SIZE];
        bufferPointer = bytesRead = 0;
    }

    public Reader(String file_name) throws IOException {
        din = new DataInputStream(new FileInputStream(file_name));
        buffer = new byte[BUFFER_SIZE];
        bufferPointer = bytesRead = 0;
    }

    public String readLine() throws IOException {
        byte[] buf = new byte[64]; // line length
        int cnt = 0, c;
        while ((c = read()) != -1) {
            if (c == '\n')
                break;
            buf[cnt++] = (byte) c;
        }
        return new String(buf, 0, cnt);
    }

    public int nextInt() throws IOException {
        int ret = 0;
        byte c = read();
        while (c <= ' ')
            c = read();
```

```
        boolean neg = (c == '-');
        if (neg)
            c = read();
        do {
            ret = ret * 10 + c - '0';
        } while ((c = read()) >= '0' && c <= '9');

        if (neg)
            return -ret;
        return ret;
    }

    public long nextLong() throws IOException {
        long ret = 0;
        byte c = read();
        while (c <= ' ')
            c = read();
        boolean neg = (c == '-');
        if (neg)
            c = read();
        do {
            ret = ret * 10 + c - '0';
        } while ((c = read()) >= '0' && c <= '9');
        if (neg)
            return -ret;
        return ret;
    }

    public double nextDouble() throws IOException {
        double ret = 0, div = 1;
        byte c = read();
        while (c <= ' ')
            c = read();
        boolean neg = (c == '-');
        if (neg)
            c = read();

        do {
            ret = ret * 10 + c - '0';
        } while ((c = read()) >= '0' && c <= '9');

        if (c == '.') {
            while ((c = read()) >= '0' && c <= '9') {
                ret += (c - '0') / (div *= 10);
            }
        }

        if (neg)
            return -ret;
        return ret;
    }
```

```
        private void fillBuffer() throws IOException {
            bytesRead = din.read(buffer, bufferPointer = 0, BUFFER_SIZE);
            if (bytesRead == -1)
                buffer[0] = -1;
        }

        private byte read() throws IOException {
            if (bufferPointer == bytesRead)
                fillBuffer();
            return buffer[bufferPointer++];
        }

        public void close() throws IOException {
            if (din == null)
                return;
            din.close();
        }
    }

    public static void main(String[] args) throws IOException {

        Reader scn = new Reader();
        PrintWriter pw = new PrintWriter(System.out);
        int n = scn.nextInt();
        int result = 0;
        int[] arr = new int[n];
        int[] lbest = new int[1000000];
        int[] rbest = new int[1000000];
        int left_value = 0, right_value = 0;
        trieNode head = new trieNode();
        insert(0, head);
        for (int i = 0; i < n; i++) {
            arr[i] = scn.nextInt();
            left_value ^= arr[i];
            lbest[i] = Math.max((i == 0) ? 0 : lbest[i - 1],
findMaxXorPair(head, left_value));
            insert(left_value, head);
        }
        head = new trieNode();
        insert(0, head);
        for (int i = n - 1; i >= 0; i--) {
            right_value ^= arr[i];
            rbest[i] = Math.max((i == n - 1) ? 0 : rbest[i + 1],
findMaxXorPair(head, right_value));
            insert(right_value, head);
            int val = rbest[i] + (i == 0 ? Integer.MIN_VALUE : lbest[i - 1]);
            if (result < val)
                result = val;
        }
        pw.println(result);
        pw.flush();
    }
```

```
}
```