MaximumExorofTwoNumbers.java

```
1     package Trie;
2     
3     import java.util.ArrayList;
4     
5     class Node1 {
6         Node1 links[] = new Node1[2];
7     
8         public Node1() {
9         }
10        boolean containsKey(int ind) {
11 2          return (links[ind] != null);
12        }
13        Node1 get(int ind) {
14 1          return links[ind];
15        }
16        void put(int ind, Node1 node) {
17            links[ind] = node;
18        }
19    };
20    class Trie {
21        private static Node1 root;
22    
23        //Initialize your data structure here
24        Trie() {
25            root = new Node1();
26        }
27        //Inserts a word into the trie
28        public static void insert(int num) {
29            Node1 node = root;
30 2          for(int i = 31;i>=0;i--) {
31 2              int bit = (num >> i) & 1;
32 1              if(!node.containsKey(bit)) {
33 1                  node.put(bit, new Node1());
34                }
35                node = node.get(bit);
36            }
37        }
38    
39        public int getMax(int num) {
40            Node1 node = root;
41            int maxNum = 0;
42 2          for(int i = 31;i>=0;i--) {
43 2              int bit = (num >> i) & 1;
44 2              if(node.containsKey(1 - bit)) {
45 2                  maxNum = maxNum | (1<<i);
46 1                  node = node.get( 1 - bit);
47                }
48                else {
49                    node = node.get(bit);
50                }
51            }
52 1          return maxNum;
53        }
54    };
55    
56    
57    
58    public class MaximumExorofTwoNumbers {
59    
60            public int maxXOR(int n, int m, ArrayList<Integer> arr1, ArrayList<Integer> arr2)
61            {
62                Trie trie = new Trie();
63 2          for(int i = 0;i<n;i++) {
64 1              trie.insert(arr1.get(i));
65          }
66          int maxi = 0;
67 2        for(int i = 0;i<m;i++) {
68              maxi = Math.max(maxi, trie.getMax(arr2.get(i)));
69          }
70 1        return maxi;
71          }
72  }
```

Mutations

```
11  1. negated conditional → KILLED
    2. replaced boolean return with true for Trie/Node1::containsKey → KILLED
14  1. replaced return value with null for Trie/Node1::get → KILLED
30  1. changed conditional boundary → KILLED
    2. negated conditional → KILLED
31  1. Replaced Shift Right with Shift Left → KILLED
    2. Replaced bitwise AND with OR → KILLED
32  1. negated conditional → KILLED
33  1. removed call to Trie/Node1::put → KILLED
42  1. negated conditional → KILLED
    2. changed conditional boundary → KILLED
43  1. Replaced bitwise AND with OR → KILLED
    2. Replaced Shift Right with Shift Left → KILLED
44  1. Replaced integer subtraction with addition → KILLED
    2. negated conditional → KILLED
45  1. Replaced Shift Left with Shift Right → KILLED
    2. Replaced bitwise OR with AND → KILLED
46  1. Replaced integer subtraction with addition → KILLED
52  1. replaced int return with 0 for Trie/Trie::getMax → KILLED
63  1. negated conditional → KILLED
    2. changed conditional boundary → KILLED
64  1. removed call to Trie/Trie::insert → KILLED
67  1. negated conditional → KILLED
    2. changed conditional boundary → KILLED
70  1. replaced int return with 0 for Trie/MaximumExorofTwoNumbers::maxXOR → KILLED
```

Active mutators

- CONDITIONALS_BOUNDARY
- EMPTY_RETURNS
- FALSE_RETURNS
- INCREMENTS
- INVERT_NEGS
- MATH
- NEGATE_CONDITIONALS
- NULL_RETURNS
- PRIMITIVE_RETURNS
- TRUE_RETURNS
- VOID_METHOD_CALLS

Tests examined

- Trie.MaximumExorofTwoNumbersTest.testtrie(Trie.MaximumExorofTwoNumbersTest) (0 ms)

Report generated by PIT 1.15.0