Tries

Ulises [2] (Tirado Zatarain | Mendez Martinez)
(ulises.tirado@cimat.mx,ulisesmdzmtz@gmail.com)

[2]Algorists Group

April, 2016

Outline

A little problem: Maximizing XOR

Statement

Given two integers, a and b, find the maximal value of x xor y, where x and y satisfy the following condition $a \leqslant x \leqslant y \leqslant b$.

Input

The input contains two unsigned integers; a is present in the first line and b in the second line. Where $1 \leqslant a < b < 10^6$.

Output

The maximal value as mentioned in the problem statement.
$\max_{x,y \in [a,b]} x \oplus y$.

Example

Sample input	Sample output
10 15	7

Explanation

The input tells us that $a = 10$ and $b = 15$. All the pairs which comply to above condition are the following:

$10 \oplus 11 = 1$ $10 \oplus 15 = 5$ $11 \oplus 15 = 4$ $13 \oplus 14 = 3$
$10 \oplus 12 = 6$ $11 \oplus 12 = 7$ $12 \oplus 13 = 1$ $13 \oplus 15 = 2$
$10 \oplus 13 = 7$ $11 \oplus 13 = 6$ $12 \oplus 14 = 2$ $14 \oplus 15 = 1$
$10 \oplus 14 = 4$ $11 \oplus 14 = 5$ $12 \oplus 15 = 3$

Here two pairs $(10, 13)$ and $(11, 12)$ have maximum XOR value 7, and this is the answer.

Big question!

How can we solve it?
Some ideas...?

Definitions

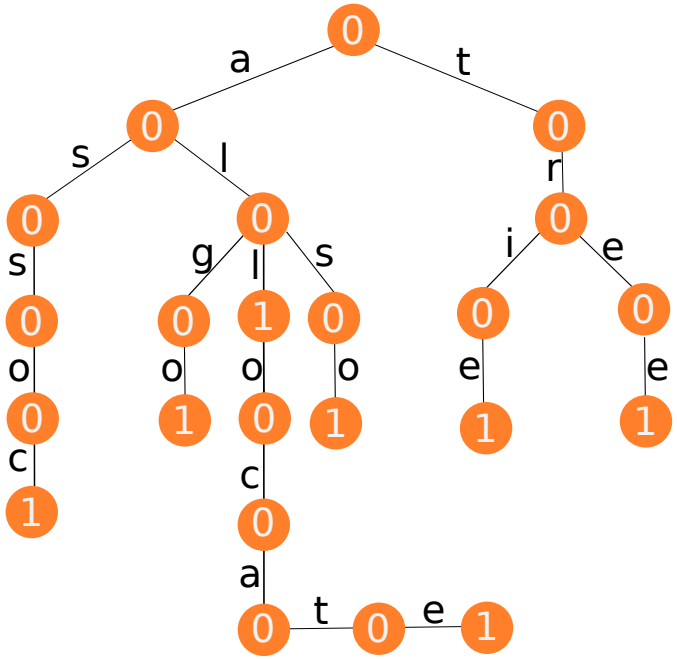
- Let a **word** be a single string and let **dictionary** be a large set of words.
- The **set<string>** and the **hash tables** can only find in a dictionary words that match exactly with the single word that we are finding.
- **Trie** is a tree type data structure that allows to represent a dictionary.
  - We can insert and find strings in $\mathcal{O}(\mathrm{L})$.
  - We can perform incremental search.

Definitions

- The word trie is an infix of the word "re**trie**val" because the trie can find a single word in a dictionary with only a prefix of the word.
- The trie is a tree where each vertex represents a single word or a prefix.
  - The root represents an empty string ε.
  - A vertex that are k edges of distance of the root have an associated prefix of length k.
  - Let v and w be two vertexes of the trie, and assume that v is a direct father of w, then v must have an associated prefix of w.
  - Deterministic acyclic finite state automaton.

Example

- The following trie stores the words: “algo”, “assoc”, “all”, ”allocate”, “also”, “tree” and “trie”.



- Note that every vertex of the tree does not store entire prefixes or entire words.

How to represent tries?

- The most simple way to represent a trie is with an struct like following:

```
struct trie <typename Type = int> begin
    Type data;
    struct trie *edge[ALPHABET_SIZE];
end
```

- For the english alphabet, we can store the $'a'$-edge in $trie::child[0]$, $'b'$-edge in $trie::child[1]$, $'c'$-edge in $trie::child[2]$ and so on until $'z'$-edge in $trie::child[25]$.

How to add a word to dictionary?

- We can add a word w as following:

```
function add-word(struct trie *t , char *w)begin
    if is-empty(w) then:
        t → data ⟵ t → data + 1;
    else:
        if is-null(t → edge[*w]) then:
            t → edge[*w] ⟵new struct trie;
        end
        add-word(t → edge[*w], w + 1);
    end
end
```

How to find a word in dictionary?

- To find a word w, we can perform following algorithm:

```
function find-word(struct trie *t , char *w)begin
    if is-null(t) then:
        return 0;
    end
    if is-empty(w) then:
        return t → data;
    end
    return find-word(t → edge[*w], w+1);
end
```

Maximizing XOR: Big question again!

How can we solve the initial problem with this?
Really?
Some ideas...?

Maximizing XOR: The idea

- Integers are words too.
- Integers are words with fix length.
- Look for the oposite bit branch.
- Save the maximum partial results.

Maximizing XOR: The idea

- Integers are words too.
- Integers are words with fix length.
- Look for the oposite bit branch.
- Save the maximum partial results.

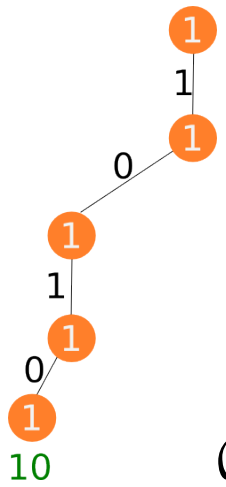
Maximizing XOR: The idea

- Integers are words too.
- Integers are words with fix length.
- Look for the oposite bit branch.
- Save the maximum partial results.

Maximizing XOR: The idea

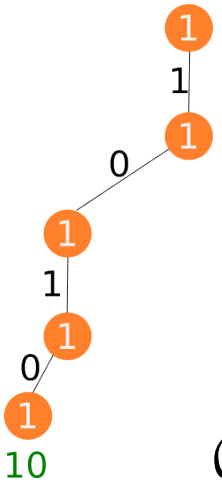
- Integers are words too.
- Integers are words with fix length.
- Look for the oposite bit branch.
- Save the maximum partial results.

Maximizing XOR: The idea

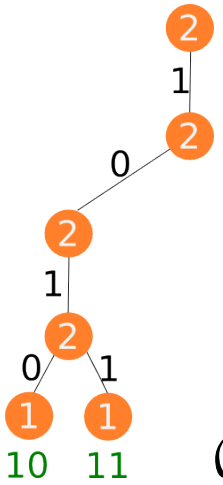


0000_2

Maximizing XOR: The idea

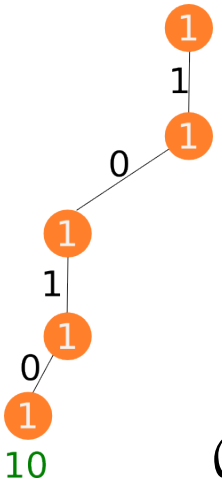


0000_2

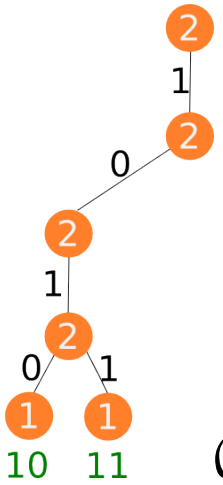


0001_2

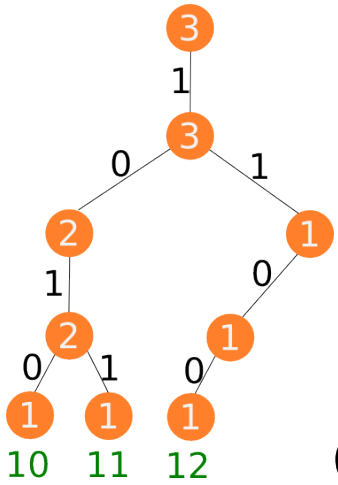
Maximizing XOR: The idea



0000_2

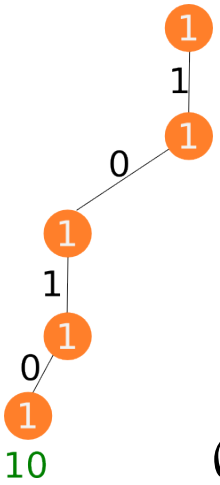


0001_2

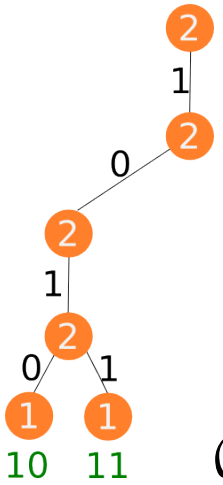


0111_2

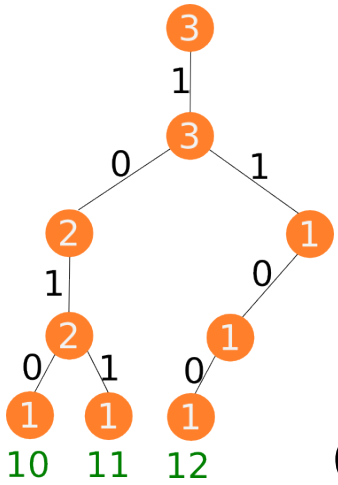
Maximizing XOR: The idea



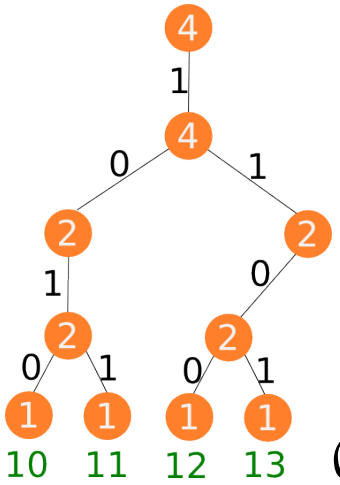
0000_2



0001_2

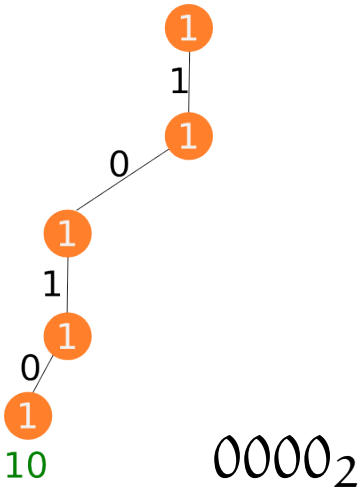


0111_2

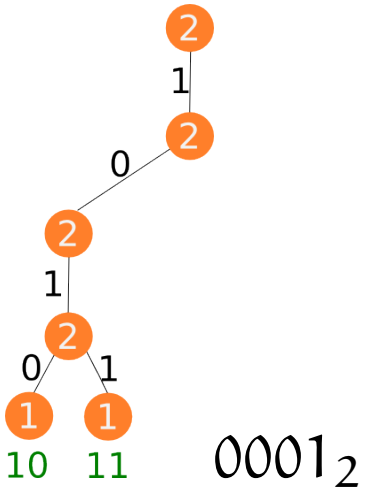


0111_2

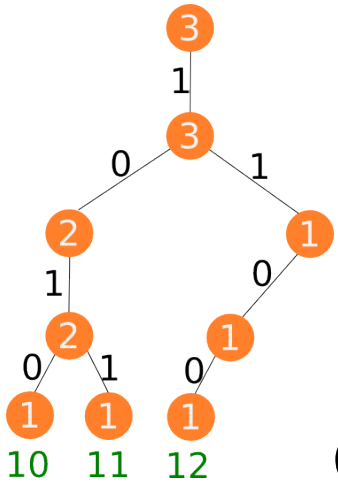
Maximizing XOR: The idea



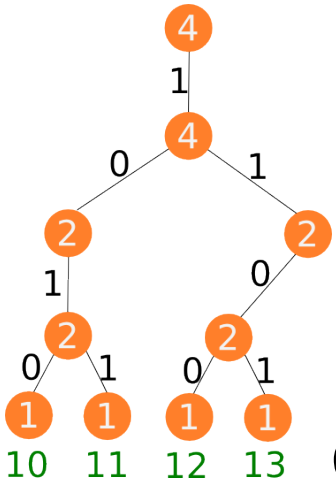
0000_2



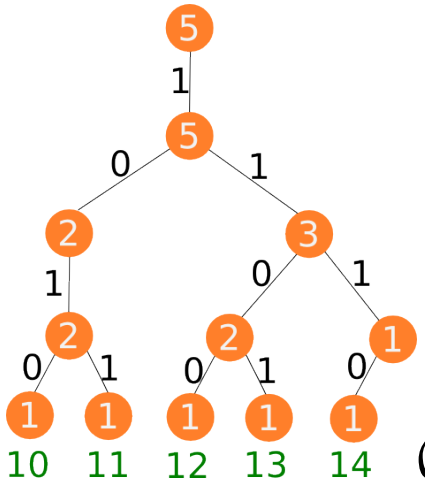
0001_2



0111_2

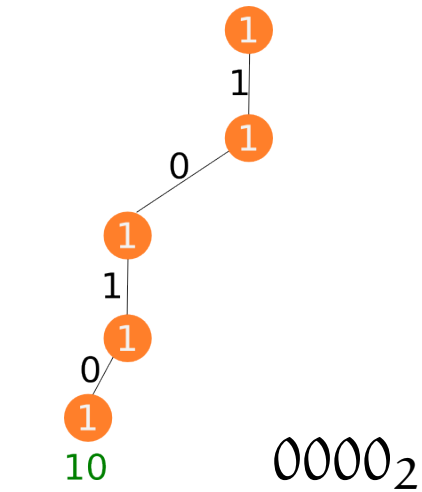


0111_2

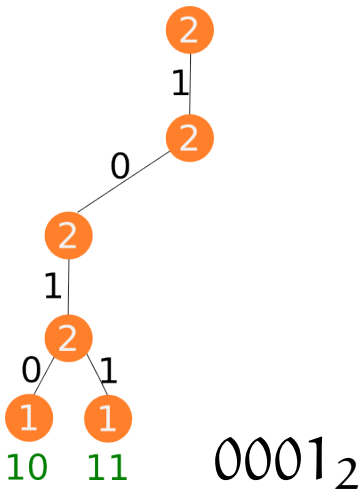


0101_2

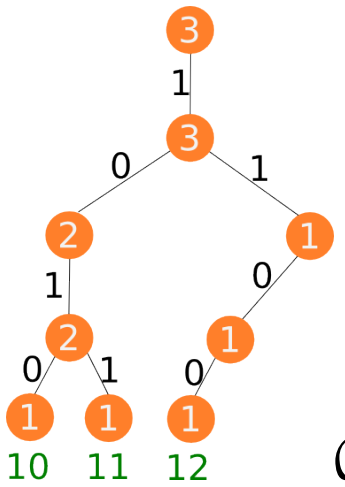
Maximizing XOR: The idea



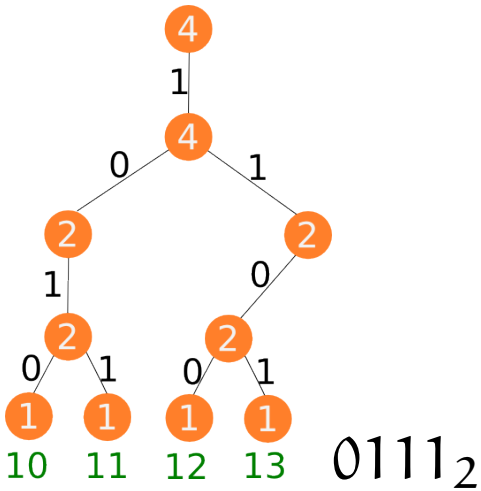
0000_2



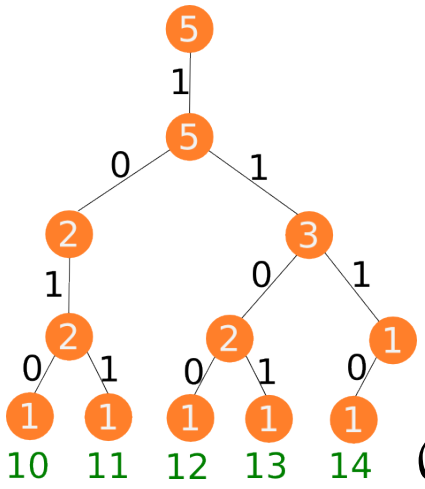
0001_2



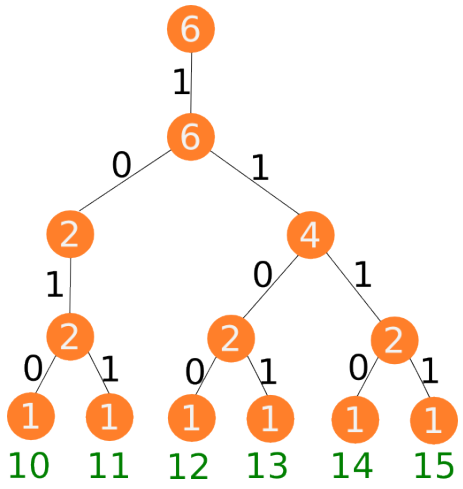
0111_2



0111_2



0101_2



0101_2

Maximizing XOR: The code

```
struct trie{
    struct trie* edge[2];
};
ull query(trie* t,ull x,int k,ull r = 0){
    if(t && k >= 0){
        ull b = (x & (1 << k)) >> k;
        if(t->edge[!b]){
            r |= 1 << k | query(t->edge[!b],x,k-1,r
        }else{
            r |= query(t->edge[b],x,k-1,r);
        }
    }
    return r;
}
```

Maximizing XOR: More big questions!

- Complexity Analysis?

Maximizing XOR: More big questions!

- Complexity Analysis?
  - $\mathcal{O}(n \log n)$

Maximizing XOR: More big questions!

- Complexity Analysis?
  - $\mathcal{O}(n \log n)$
- What about other bitwise operators like AND, OR, ...?

Maximizing XOR: More big questions!

- Complexity Analysis?
  - $\mathcal{O}(n \log n)$
- What about other bitwise operators like AND, OR, ...?
  - It's almost same! What do we need to change?

More aplications & variants

- Incremental search
- Orthographic corrector
- Huffman coding
- Data compression
- Using an **std::map<char,struct trie*>** instead fixed array;
- Probabilistic tries
- Imagine & let's TRIE! :)

References I

- Codeforces
- TopCoder
- Wikipedia
- Quora
- HackerRank