Tries
The Problem

- Given a set dictionary of words
- How many strings exist with a given prefix?

Example:

d = ["cat", "car", "dog", "dope", "doom", "bingo"]

hasPrefix(d, "ca") = 2

hasPrefix(d, "do") = 3
Arrays

- Naive solution
- Store entire set in array
- Have to look at every string for prefix

```d = ["cat", "car", "dog", "dope", "doom", "bingo"]
hasPrefix(d, "ca") = 2
hasPrefix(d, "do") = 3
Runtime?```
HashMap

- Map<String, Integer>
- O(1) time complexity for all operations
- Horrific space complexity:
  - O(N |S|) keys where O(|S|) is the length of the string
## Supported Operations + Runtimes

<table>
<thead>
<tr>
<th>Operation</th>
<th>ArrayList</th>
<th>SortedList</th>
<th>HashMap</th>
<th>Balanced BST</th>
<th>Trie</th>
</tr>
</thead>
<tbody>
<tr>
<td>Insert(S)</td>
<td>$O(1)$</td>
<td>$O(n)$</td>
<td>$O(1)$</td>
<td>$O(\log N)$</td>
<td>$O(</td>
</tr>
<tr>
<td>Query(S)</td>
<td>$O(n)$</td>
<td>$O(\log n)$</td>
<td>$O(1)$</td>
<td>$O(\log N)$</td>
<td>$O(</td>
</tr>
<tr>
<td>Remove(S)</td>
<td>$O(n)$</td>
<td>$O(n)$</td>
<td>$O(1)$</td>
<td>$O(\log N)$</td>
<td>$O(</td>
</tr>
<tr>
<td>CountPrefix(P)</td>
<td>$O(n)$</td>
<td>$O(\log n)$</td>
<td>$O(n)$</td>
<td>$O(\log N)$</td>
<td>$O(</td>
</tr>
<tr>
<td>Difficulty to write</td>
<td>easy</td>
<td>easy</td>
<td>easy</td>
<td>hard</td>
<td>medium</td>
</tr>
</tbody>
</table>
Tries

- Also known as “prefix trees”
- A kind of tree which stores every prefix of a word

Modifications

- In addition to just storing if a word exists or not, we can store the number of words with a given prefix.

Recursive code

https://gist.github.com/arknave/5c2f14b651425ec24c4604f2c671760e

Errors:

Uses $O(|S|^2)$ time to insert because copies the string at each function call!
Iterative Trie

https://gist.github.com/arknave/f68f9f0a4e8bd37b8c5f7965c5b66d31

MAX_LETTERS = MAX_LENGTH * NUM_WORDS

Allocate all the nodes up front
Problems

- Given a set of b-bit binary numbers $S$, and a b-bit input number $N$, choose a number from the set that maximizes $S_i \text{xor} \ N$.
  
  - Tries don't have to be on letters!