Short Python/C++ solution

97 StefanPochmann ★ 47270 Last Edit: October 7, 2018 4:18 AM 11.4K VIEWS

Python Solution (accepted in ~870 ms)

```python
def wordSquares(self, words):
    n = len(words[0])
    fulls = collections.defaultdict(list)
    for word in words:
        for i in range(n):
            fulls[word[:i]].append(word)
    def build(square):
        if len(square) == n:
            squares.append(square)
            return
        for word in fulls[''.join(zip(*square)[len(square)])]:
            build(square + [word])
    squares = []
    for word in words:
        build([word])
    return squares
```

Explanation

I try every word for the first row. For each of them, try every fitting word for the second row. And so on. The first few rows determine the first few columns and thus determine how the next row's word must start. For example:

```
wall      Try words      wall                      wall                      wall
a...   => starting  =>   area      Try words       area                      area
l...      with "a"       le..   => starting  =>    lead      Try words       lead
l...                     la..      with "le"       lad.   => starting   =>   lady
                                                             with "lad"
```

For quick lookup, my `fulls` dictionary maps prefixes to lists of words who have that prefix.

C++ Solution (accepted in ~180 ms)

```cpp
class Solution {
public:
    vector<vector<string>> wordSquares(vector<string>& words) {
        n = words[0].size();
        square.resize(n);
        for (string word : words)
            for (int i=0; i<n; i++)
                fulls[word.substr(0, i)].push_back(word);
        build(0);
        return squares;

    }
    int n;
    unordered_map<string, vector<string>> fulls;
    vector<string> square;
    vector<vector<string>> squares;
    void build(int i) {
        if (i == n) {
            squares.push_back(square);
            return;
        }
        string prefix;
        for (int k=0; k<i; k++)
            prefix += square[k][i];
        for (string word : fulls[prefix]) {
            square[i] = word;
            build(i + 1);
        }
    }
};
```