< Back | Python with bit masks

StefanPochmann ★ 47270 October 5, 2016 3:45 AM 4.8K VIEWS

34

Gets accepted in ~130 ms.

```
def minAbbreviation(self, target, dictionary):
    m = len(target)
    diffs = {sum(2**i for i, c in enumerate(word) if target[i] != c)
             for word in dictionary if len(word) == m}
    if not diffs:
        return str(m)
    bits = max((i for i in range(2**m) if all(d & i for d in diffs)),
               key=lambda bits: sum((bits >> i) & 3 == 0 for i in range(m-1)))
    s = ''.join(target[i] if bits & 2**i else '#' for i in range(m))
    return re.sub('#+', lambda m: str(len(m.group())), s)
```

If the target is `apple` and the dictionary contains `apply`, then the abbreviation must include the `e` as the letter `e`, not in a number. It's the only letter distinguishing these two words. Similarly, if the dictionary contains `tuple`, then the abbreviation must include the `a` or the first `p` as a letter.

For each dictionary word (of correct size), I create a diff-number whose bits tell me which of the word's letters differ from the target. Then I go through the 2^m possible abbreviations, represented as number from 0 to 2^m-1, the bits representing which letters of target are in the abbreviation. An abbreviation is ok if it doesn't match any dictionary word. To check whether an abbreviation doesn't match a dictionary word, I simply check whether the abbreviation number and the dictionary word's diff-number have a common 1-bit. Which means that the abbreviation contains a letter where the dictionary word differs from the target.

Then from the ok abbreviations I find one that maximizes how much length it saves me. Two consecutive 0-bits in the abbreviation number mean that the two corresponding letters will be encoded as the number 2. It saves length 1. Three consecutive 0-bits save length 2, and so on. To compute the saved length, I just count how many pairs of adjacent bits are zero.

Now that I have the number representing an optimal abbreviation, I just need to turn it into the actual abbreviation. First I turn it into a string where each 1-bit is turned into the corresponding letter of the target and each 0-bit is turned into `#`. Then I replace streaks of `#` into numbers.

Comments: 8 Best Most Votes Newest to Oldest Oldest to Newest

Type comment here... (Markdown is supported)

Post

cwl ★ 261 Last Edit: September 21, 2018 5:18 AM