infinute ★ 330 April 18, 2019 9:29 PM 186 VIEWS

0

`i,j` are two pointers. `i` points to the char in the pattern we are working on; `j` points to the position where the mapping of `pattern[j]` starts with.
`memo` records the bijection between the `pattern` and `str`.

```
from collections import Counter
class Solution:
    def wordPatternMatch(self, pattern: str, str: str) -> bool:
        def dfs(i, j):
            if i == len(pattern) and j == len(str): return True
            if i >= len(pattern) or j >= len(str): return False
            if pattern[i] in memo:
                if memo[pattern[i]] == str[j: j + len(memo[pattern[i]])] and dfs(i + 1, j + len(memo[pattern[i]])):
                    return True
            else:
                for k in range(j, len(str)):
                    if counter[pattern[i]] <= str[j:].count(str[j: k + 1]) and str[j: k + 1] not in memo.values():
                        memo[pattern[i]] = str[j: k + 1]
                        if dfs(i + 1, k + 1): return True
                        memo.pop(pattern[i])
            return False

        counter = Counter(pattern)
        memo = {}
        return dfs(0, 0)
```