Python backtracking

kaien ★ 41 June 16, 2019 11:34 AM 2.6K VIEWS

39

```
class Solution(object):
    def expand(self, S):
        """
        :type S: str
        :rtype: List[str]
        """
        self.res = []
        def helper(s, word):
            if not s:
                self.res.append(word)
            else:
                if s[0] == "{":
                    i = s.find("}")
                    for letter in s[1:i].split(','):
                        helper(s[i+1:], word+letter)
                else:
                    helper(s[1:], word + s[0])
        helper(S, "")
        self.res.sort()
        return self.res
```

Comments: 11 Best | Most Votes | Newest to Oldest | Oldest to Newest

Type comment here... (Markdown is supported)

Post

goodstudy123 ★ 24 September 21, 2019 5:30 AM

Nice and clean code. What is the time complexity for this ?

7 Show 3 replies Reply

Zri ★ 12 October 5, 2019 10:44 PM

These are the kinds of solutions that make me doubt myself.... Beautiful Code!

3 Reply

satwikkansal ★ 5 November 25, 2019 12:17 AM

If you use indices instead of slicing you can get 20% speedup.

Following solution beats 99% of Python solutions.

```
class Solution:
    def expand(self, S: str) -> List[str]:
        self.res = []
        self._expand(S, 0, len(S), "")
        return sorted(self.res)

    def _expand(self, string:str, start:int, size:int, prefix:str):
```

Read More

2 Reply

eigenvec ★ 72 December 22, 2019 10:24 AM

(1) same idea, but using indices
(2) we can sort the letters within dfs, instead of sorting at the end

```
    def expand(self, S: str) -> List[str]:
        res = []
        def helper(S, word, idx):
            if idx == len(S):
                res.append(word)
            else:
                if S[idx] == "{":
                    i = idx + 1
```

Read More

1 Reply