811. Subdomain Visit Count

Solved

Medium | Topics | Companies

A website domain `"discuss.leetcode.com"` consists of various subdomains. At the top level, we have `"com"`, at the next level, we have `"leetcode.com"` and at the lowest level, `"discuss.leetcode.com"`. When we visit a domain like `"discuss.leetcode.com"`, we will also visit the parent domains `"leetcode.com"` and `"com"` implicitly.

A **count-paired domain** is a domain that has one of the two formats `"rep d1.d2.d3"` or `"rep d1.d2"` where `rep` is the number of visits to the domain and `d1.d2.d3` is the domain itself.

- For example, `"9001 discuss.leetcode.com"` is a **count-paired domain** that indicates that `discuss.leetcode.com` was visited `9001` times.

Given an array of **count-paired domains** `cpdomains`, return *an array of the **count-paired domains** of each subdomain in the input*. You may return the answer in **any order**.

Example 1:

> **Input:** cpdomains = ["9001 discuss.leetcode.com"]
> **Output:** ["9001 leetcode.com","9001 discuss.leetcode.com","9001 com"]
> **Explanation:** We only have one website domain: "discuss.leetcode.com".
> As discussed above, the subdomain "leetcode.com" and "com" will also be visited. So they will all be visited 9001 times.

```python
class Solution:
    def subdomainVisits(self, cpdomains: List[str]) -> List[str]:
        res = {}
        seen = set()
        for cp in cpdomains:
            n, s = cp.split()
            subs = s.split('.')

            for i in range(len(subs)):
                temp = '.'.join(subs[i:])
                print(temp)
                if temp not in seen:
                    res[temp] = int(n)
                    seen.add(temp)
                else:
                    res[temp] += int(n)

        ans = []
        for r in res:
            t = f"{res[r]} {r}"
            ans.append(t)

        return ans
```

my solution, using the DA thinking

Java | Python — Copy

```python
class Solution(object):
    def subdomainVisits(self, cpdomains):
        ans = collections.Counter()
        for domain in cpdomains:
            count, domain = domain.split()
            count = int(count)
            frags = domain.split('.')
            for i in xrange(len(frags)):
                ans[".".join(frags[i:])] += count

        return ["{} {}".format(ct, dom) for dom, ct in ans.items()]
```

editorial, more efficient