JavaScript Subdomain Visit Count

Challenge

A website domain `'discuss.superklok.com'` consists of various subdomains. At the top level, we have `'com'`, at the next level, we have `'superklok.com'` and at the lowest level, `'discuss.superklok.com'`. When we visit a domain like `'discuss.superklok.com'`, we will also visit the parent domains `'superklok.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.superklok.com'` is a count-paired domain that indicates that `discuss.superklok.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.

1st Example

```
Input: cpdomains = ['9001 discuss.superklok.com']
```

Example continues on next page...

```
Output: [
            '9001 superklok.com',
            '9001 discuss.superklok.com',
            '9001 com'
        ]
Explanation:
We only have one website domain: 'discuss.superklok.com'. As
discussed above, the subdomain 'superklok.com' and 'com' will
also be visited. So they will all be visited 9001 times.
```

2nd Example

```
Input: cpdomains = [
                    '900 trevmorin.mail.com',
                    '50 jetwheelreel.com',
                    '1 currentcourant.mail.com',
                    '5 swingcars.net'
                ]
Output: [
            '901 mail.com','50 jetwheelreel.com',
            '900 trevmorin.mail.com','5 swingcars.net',
            '5 org','1 currentcourant.mail.com','951 com'
        ]
Explanation:
We will visit 'trevmorin.mail.com' 900 times,
'jetwheelreel.com' 50 times, 'currentcourant.mail.com' once
and 'swingcars.net' 5 times. For the subdomains, we will
visit 'mail.com' 900 + 1 = 901 times, 'com' 900 + 50 + 1 =
951 times, and 'org' 5 times.
```

Constraints

- `1 <= cpdomain.length <= 100`
- `1 <= cpdomain[i].length <= 100`

- `cpdomain[i]` follows either the `'rep_i d1_i.d2_i.d3_i'` format or the `'rep_i d1_i.d2_i'` format.
- `rep_i` is an integer in the range `[1, 10^4]`.
- `d1_i`, `d2_i`, and `d3_i` consist of lowercase English letters.

Solution

```
const subdomainVisits = (cpdomains) => {
    let visitCounts = {};

    for (let i = 0; i < cpdomains.length; i++) {
        const [visits, domains] = cpdomains[i].split(' ');

        let subdomains = domains.split('.');

        while (subdomains.length) {
            let subdomain = subdomains.join('.');

            visitCounts[subdomain] = visitCounts
                            .hasOwnProperty(subdomain) ?
                            Number(visitCounts[subdomain]) +
                            Number(visits) : visits;

            subdomains.shift();
        }
    }

    return Object
            .keys(visitCounts)
            .map((key) => `${visitCounts[key]} ${key}`);
};
```

Explanation

I've written a function called `subdomainVisits` that takes in an array of strings called `cpdomains`. Each string in the array represents a number of visits to a particular domain. The function returns an array of strings, where each string represents the number of visits and the corresponding subdomain.

Inside the function, an empty object called `visitCounts` is initialized to store the count of visits for each subdomain.

The function then enters a loop that iterates over each element in the `cpdomains` array.

For each element, it splits the string into two parts: the number of visits and the domain. This is done using the `split` method with a space as the separator. The resulting values are stored in variables called `visits` and `domains`.

The `domains` string is further split into an array of subdomains using the `split` method with a dot as the separator. The resulting array is stored in a variable called `subdomains`.

A `while` loop is then initiated, which continues as long as there are elements in the `subdomains` array.

Within the `while` loop, it joins the elements of the `subdomains` array into a string with dots in between using the `join` method. This creates the full subdomain.

It checks if the `visitCounts` object has a property with the current subdomain using the `hasOwnProperty` method. If it does, it increments the count of visits for that subdomain by adding the

current number of visits (converted to a number). If it doesn't, it assigns the current number of visits to that subdomain.

After that, it removes the first element from the `subdomains` array using the `shift` method, allowing the `while` loop to continue with the remaining subdomains.

Once the `while` loop ends, the function returns an array of strings. It uses the `map` method on the keys of the `visitCounts` object. For each key, it creates a string by concatenating the count of visits and the subdomain.

The resulting array of strings is the final result of the function.

In summary, the `subdomainVisits` function counts the number of visits to subdomains based on the input array of strings. It uses an object to store the count of visits for each subdomain and returns an array of strings representing the count and corresponding subdomains.

Author: Trevor Morin