JavaScript Sort Characters by Frequency

Challenge

Given a string `s`, sort it in decreasing order based on the frequency of the characters. The frequency of a character is the number of times it appears in the string.

Return the sorted string. If there are multiple answers, return any of them.

1st Example

```
Input: s = 'tree'
Output: 'eert'
Explanation: 'e' appears twice while 'r' and 't' both appear
             once. So 'e' must appear before both 'r' and
             't'. Therefore 'eetr' is also a valid answer.
```

2nd Example

```
Input: s = 'cccaaa'
Output: 'aaaccc'
Explanation: Both 'c' and 'a' appear three times, so both
             'cccaaa' and 'aaaccc' are valid answers. Note
             that 'cacaca' is incorrect, as the same
             characters must be together.
```

3rd Example

```
Input: s = 'Aabb'
Output: 'bbAa'
Explanation: 'bbaA' is also a valid answer, but 'Aabb' is
             incorrect. Note that 'A' and 'a' are treated
             as two different characters.
```

Constraints

- `1 <= s.length <= 5 * 10⁵`
- `s` consists of uppercase and lowercase English letters and digits.

Solution

```
const frequencySort = (s) => {
    let hashMap = new Map(),
        result  = [];

    for (let i = 0; i < s.length; i++) {
        if (!hashMap.has(s[i])) {
            hashMap.set(s[i], 1);
        } else {
            hashMap.set(s[i], hashMap.get(s[i]) + 1);
        }
    }

    let sortedMap  = new Map(
        [...hashMap.entries()].sort((a, b) => b[1] - a[1])
    );
```

Solution continues on next page...

```
    for (let [key, value] of sortedMap) {
        let i = 0;

        while (i < value) {
            result.push(key);

            i++;
        }
    }

    return result.join('');
};
```

Explanation

I've created a function called `frequencySort` that takes a string `s` as input. Its purpose is to sort the characters in the string based on their frequency.

Inside the function, an empty `hashMap` is created using the `Map` object to store the frequency of each character in the string. There is also an empty `result` array that will store the sorted characters.

It then iterates over each character in the string using a for loop. Inside the loop, it checks if the character is already present in the `hashMap`. If it is not present, it adds the character as a key in the `hashMap` with a value of `1`. If it is already present, it increments the value of that character in the `hashMap` by `1`.

Next, a new `sortedMap` is created by converting the entries of the `hashMap` to an array, sorting it in descending order based on the

frequency of characters, and converting it back to a `Map`.

Another loop is used to iterate over the entries of the `sortedMap` using a for-of loop. Inside this loop, the key and value of each entry are retrieved. Then, a counter `i` is initialized to `0`.

A while loop is used that runs `value` number of times. Inside the while loop, the key (character) is pushed into the `result` array, and the counter `i` is incremented.

Finally, the function returns the `result` array joined as a string, which represents the sorted characters based on their frequency.

In summary, the `frequencySort` function sorts the characters in the input string based on their frequency. It uses a `Map` object to store the frequency of each character, sorts the entries of the `Map` in descending order, and builds a sorted array of characters. The function then returns the sorted characters as a string.

Author: Trevor Morin