JavaScript Groups of Special-Equivalent Strings

Challenge

You are given an array of strings of the same length `words`.

In one move, you can swap any two even indexed characters or any two odd indexed characters of a string `words[i]`.

Two strings `words[i]` and `words[j]` are special-equivalent if after any number of moves, `words[i] == words[j]`.

For example, `words[i] = 'zzxy'` and `words[j] = 'xyzz'` are special-equivalent because we may make the moves `'zzxy' -> 'xzzy' -> 'xyzz'`.

A group of special-equivalent strings from `words` is a non-empty subset of words such that:

- Every pair of strings in the group are special equivalent.
- The group is the largest size possible (ex. there is not a string `words[i]` not in the group such that `words[i]` is special-equivalent to every string in the group).

Return the number of groups of special-equivalent strings from `words`.

1st Example

```
Input: words = ['abcd','cdab','cbad',
                'xyzz','zzxy','zzyx']
Output: 3
Explanation: One group is ['abcd', 'cdab', 'cbad'], since
             they are all pairwise special equivalent, and
             none of the other strings is all pairwise
             special equivalent to these. The other two
             groups are ['xyzz', 'zzxy'] and ['zzyx']. Note
             that in particular, 'zzxy' is not special
             equivalent to 'zzyx'.
```

2nd Example

```
Input: words = ['abc','acb','bac','bca','cab','cba']
Output: 3
```

Constraints

- `1 <= words.length <= 1000`
- `1 <= words[i].length <= 20`
- `words[i]` consist of lowercase English letters.
- All the strings are of the same length.

Solution

```
const numSpecialEquivGroups = (words) => {
    const transform = (word) => {
        const odd  = [],
              even = [];

        for (let index = 0; index < word.length; index++) {
            const str = word[index];

            index & 1 ? odd.push(str) : even.push(str);
        }

        return `${odd.sort()}${even.sort()}`;
    };

    return words.reduce((set, word) => {
        return set.add(transform(word));
    }, new Set()).size;
};
```

Explanation

I've defined a function called `numSpecialEquivGroups` that takes in an array of words as an argument and returns the number of groups of words that have the same special equivalence.

Inside the function, there is a helper function called `transform`. This function takes in a word and splits it into two arrays: one for the odd-indexed characters and one for the even-indexed characters. It then sorts both arrays and concatenates them into a single string.

The main function, `numSpecialEquivGroups`, uses the `reduce` method on the input array of words. It applies the `transform` function to each word and adds the resulting string to a new `Set` object.

Finally, the function returns the size of the `Set` object, which represents the number of unique special equivalence groups in the input array.

In summary, the `numSpecialEquivGroups` function groups words based on their special equivalence and returns the count of unique groups. The special equivalence is determined by splitting the words into odd-indexed and even-indexed characters, sorting them, and concatenating them into a single string.

Author: Trevor Morin