Problem 2700: Differences Between Two Objects

Problem Information

Difficulty: Medium
Acceptance Rate: 74.82%
Paid Only: Yes

Problem Description

Write a function that accepts two deeply nested objects or arrays `obj1` and `obj2` and returns a new object representing their differences.

The function should compare the properties of the two objects and identify any changes. The returned object should only contains keys where the value is different from `obj1` to `obj2`.

For each changed key, the value should be represented as an array `[obj1 value, obj2 value]`. Keys that exist in one object but not in the other should not be included in the returned object. The end result should be a deeply nested object where each leaf value is a difference array.

When comparing two arrays, the indices of the arrays are considered to be their keys.

You may assume that both objects are the output of `JSON.parse`.

Example 1:

Input: obj1 = {} obj2 = { "a": 1, "b": 2 } **Output:** {} **Explanation:** There were no modifications made to obj1. New keys "a" and "b" appear in obj2, but keys that are added or removed should be ignored.

Example 2:

Input: obj1 = { "a": 1, "v": 3, "x": [], "z": { "a": null } } obj2 = { "a": 2, "v": 4, "x": [], "z": { "a": 2 } } **Output:** { "a": [1, 2], "v": [3, 4], "z": { "a": [null, 2] } } **Explanation:** The keys "a", "v", and "z" all had changes applied. "a" was changed from 1 to 2. "v" was changed from 3 to 4. "z" had a change applied to a child object. "z.a" was changed from null to 2.

Example 3:

Input: obj1 = { "a": 5, "v": 6, "z": [1, 2, 4, [2, 5, 7]] } obj2 = { "a": 5, "v": 7, "z": [1, 2, 3, [1]] } **Output:** { "v": [6, 7], "z": { "2": [4, 3], "3": { "0": [2, 1] } } } **Explanation:** In obj1 and obj2, the keys "v" and "z" have different assigned values. "a" is ignored because the value is unchanged. In the key "z", there is a nested array. Arrays are treated like objects where the indices are keys. There were two alterations to the the array: z[2] and z[3][0]. z[0] and z[1] were unchanged and thus not included. z[3][1] and z[3][2] were removed and thus not included.

Example 4:

Input: obj1 = { "a": {"b": 1}, } obj2 = { "a": [5], } **Output:** { "a": [{"b": 1}, [5]] } **Explanation:** The key "a" exists in both objects. Since the two associated values have different types, they are placed in the difference array.

Example 5:

Input: obj1 = { "a": [1, 2, {}], "b": false } obj2 = { "b": false, "a": [1, 2, {}] } **Output:** {} **Explanation:** Apart from a different ordering of keys, the two objects are identical so an empty object is returned.

Constraints:

* `obj1` and `obj2` are valid JSON objects or arrays * `2 <= JSON.stringify(obj1).length <= 104` * `2 <= JSON.stringify(obj2).length <= 104`

Code Snippets

JavaScript:

```
/**
* @param {Object|Array} obj1
* @param {Object|Array} obj2
* @return {Object|Array}
*/
function objDiff(obj1, obj2) {

};
```

TypeScript:

```
type JSONValue = null | boolean | number | string | JSONValue[] | { [key:
string]: JSONValue };
type Obj = Record<string, JSONValue> | Array<JSONValue>

function objDiff(obj1: Obj, obj2: Obj): Obj {

};
```