Problem 3387: Maximize Amount After Two Days of Conversions

Problem Information

Difficulty: Medium
Acceptance Rate: 60.48%
Paid Only: No
Tags: Array, String, Depth-First Search, Breadth-First Search, Graph

Problem Description

You are given a string `initialCurrency`, and you start with `1.0` of `initialCurrency`.

You are also given four arrays with currency pairs (strings) and rates (real numbers):

* `pairs1[i] = [startCurrencyi, targetCurrencyi]` denotes that you can convert from `startCurrencyi` to `targetCurrencyi` at a rate of `rates1[i]` on **day 1**. * `pairs2[i] = [startCurrencyi, targetCurrencyi]` denotes that you can convert from `startCurrencyi` to `targetCurrencyi` at a rate of `rates2[i]` on **day 2**. * Also, each `targetCurrency` can be converted back to its corresponding `startCurrency` at a rate of `1 / rate`.

You can perform **any** number of conversions, **including zero** , using `rates1` on day 1, **followed** by any number of additional conversions, **including zero** , using `rates2` on day 2.

Return the **maximum** amount of `initialCurrency` you can have after performing any number of conversions on both days **in order**.

Note: Conversion rates are valid, and there will be no contradictions in the rates for either day. The rates for the days are independent of each other.

Example 1:

Input: initialCurrency = "EUR", pairs1 = [["EUR","USD"],["USD","JPY"]], rates1 = [2.0,3.0], pairs2 = [["JPY","USD"],["USD","CHF"],["CHF","EUR"]], rates2 = [4.0,5.0,6.0]

Output: 720.00000

Explanation:

To get the maximum amount of **EUR** , starting with 1.0 **EUR** :

* On Day 1: * Convert **EUR** to **USD** to get 2.0 **USD**. * Convert **USD** to **JPY** to get 6.0 **JPY**. * On Day 2: * Convert **JPY** to **USD** to get 24.0 **USD**. * Convert **USD** to **CHF** to get 120.0 **CHF**. * Finally, convert **CHF** to **EUR** to get 720.0 **EUR**.

Example 2:

Input: initialCurrency = "NGN", pairs1 = [["NGN","EUR"]], rates1 = [9.0], pairs2 = [["NGN","EUR"]], rates2 = [6.0]

Output: 1.50000

Explanation:

Converting **NGN** to **EUR** on day 1 and **EUR** to **NGN** using the inverse rate on day 2 gives the maximum amount.

Example 3:

Input: initialCurrency = "USD", pairs1 = [["USD","EUR"]], rates1 = [1.0], pairs2 = [["EUR","JPY"]], rates2 = [10.0]

Output: 1.00000

Explanation:

In this example, there is no need to make any conversions on either day.

Constraints:

* `1 <= initialCurrency.length <= 3` * `initialCurrency` consists only of uppercase English letters. * `1 <= n == pairs1.length <= 10` * `1 <= m == pairs2.length <= 10` * `pairs1[i] == [startCurrencyi, targetCurrencyi]` * `pairs2[i] == [startCurrencyi, targetCurrencyi]` * `1 <= startCurrencyi.length, targetCurrencyi.length <= 3` * `startCurrencyi` and `targetCurrencyi`

consist only of uppercase English letters. * `rates1.length == n` * `rates2.length == m` * `1.0 <= rates1[i], rates2[i] <= 10.0` * The input is generated such that there are no contradictions or cycles in the conversion graphs for either day. * The input is generated such that the output is **at most** `5 * 1010`.

Code Snippets

C++:

```
class Solution {
public:
double maxAmount(string initialCurrency, vector<vector<string>>& pairs1,
vector<double>& rates1, vector<vector<string>>& pairs2, vector<double>&
rates2) {

}
};
```

Java:

```
class Solution {
public double maxAmount(String initialCurrency, List<List<String>> pairs1,
double[] rates1, List<List<String>> pairs2, double[] rates2) {

}
}
```

Python3:

```
class Solution:
def maxAmount(self, initialCurrency: str, pairs1: List[List[str]], rates1:
List[float], pairs2: List[List[str]], rates2: List[float]) -> float:
```