

# 2307. Check for Contradictions in Equations

[Leetcode Link](#)

## Problem Description

You are given a 2D array of strings `equations` and an array of real numbers `values`, where `equations[i] = [Ai, Bi]` and `values[i]` means that `Ai / Bi = values[i]`. Your task is to determine if there exists a contradiction in the equations. Return `true` if there is a contradiction, or `false` otherwise.

## Example

Consider the following input:

- `equations = [["a", "b"], ["b", "c"]]`
- `values = [2.0, 3.0]`

For this input, there is no contradiction and the output should be `false`.

## Approach

We can use the concept of a directed weighted graph to represent equations that reflect their divisions. The graph vertices are the unique element strings in the input `equations` array, and the edges are the given fractions or their reciprocals (reverse fractions). As we traverse the graph, we need to track the accumulated products of the weights.

We can use a depth-first search (DFS) algorithm to traverse the graph, keeping track of the visited nodes and their corresponding accumulated products. A contradiction will occur if we visit a previously visited node and the ratio of the current accumulated value to the previously recorded value is significantly different.

The input string values are converted to integer indices to improve performance.

## Algorithm

- Create a hash map to map strings to integer indices.
- Build a directed weighted graph from the input `equations` array.
- Initialize a `seen` array to track the accumulated value of visited nodes.
- Iterate through the graph and perform a DFS from each node.
- If there is a contradiction during DFS traversal, return `true`, else return `false`.

## DFS Traversal Example

Input:

- `equations = [["a", "b"], ["b", "c"]]`
- `values = [2.0, 3.0]`

Step by step process:

- Convert string to int:

```
strToInt = {"a": 0, "b": 1, "c": 2}
```

- Build directed weighted graph:

```
graph = [ [(1, 2.0)], # ("a", "b") with value 2.0 [(0, 0.5), (2, 3.0)], # ("b", "a") with value 0.5, ("b", "c") with value 3.0 [(1, 1/3)] # ("c", "b") with value 1/3 ]
```

- Initialize seen array:

```
seen = [0, 0, 0]
```

- Iterate through graph and perform DFS:

- For node 0, DFS traversal: 0 -> 1 -> 2
  - Update `seen` array: [1, 2, 6]
- All nodes are visited, no contradiction found.

- Return `false` as no contradiction found.

## C++ Solution

```
1  cpp
2  #include <cmath>
3  #include <unordered_map>
4  #include <vector>
5  using namespace std;
6
7  class Solution {
8  public:
9      bool checkContradictions(vector<vector<string>>& equations,
10                             vector<double>& values) {
11          // Convert string to int for better performance
12          unordered_map<string, int> strToInt;
13
14          for (const vector<string>& equation : equations) {
15              const string& u = equation[0];
16              const string& v = equation[1];
17              if (!strToInt.count(u))
18                  strToInt[u] = strToInt.size();
19              if (!strToInt.count(v))
20                  strToInt[v] = strToInt.size();
21          }
22
23          vector<vector<pair<int, double>>> graph(strToInt.size());
24          vector<double> seen(graph.size());
25
26          for (int i = 0; i < equations.size(); ++i) {
27              const int u = strToInt.at(equations[i][0]);
28              const int v = strToInt.at(equations[i][1]);
29              graph[u].emplace_back(v, values[i]);
30              graph[v].emplace_back(u, 1 / values[i]);
31          }
32
33          for (int i = 0; i < graph.size(); ++i)
34              if (!seen[i] && dfs(graph, i, seen, 1.0))
35                  return true;
36
37          return false;
38      }
39  private:
40      bool dfs(const vector<vector<pair<int, double>>>& graph, int u,
41              vector<double>& seen, double val) {
42          if (seen[u])
43              return abs(val / seen[u] - 1) > 1e-5;
44          seen[u] = val;
45          for (const auto& [v, w] : graph[u])
46              if (dfs(graph, v, seen, val / w))
47                  return true;
48          return false;
49      }
50  };
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
```

Note: Make sure to include the necessary header files when using the C++ solution.## Python Solution

```
1  python
2  from typing import List, Tuple
3
4  class Solution:
5      def checkContradictions(self, equations: List[List[str]], values: List[float]) -> bool:
6          # Convert string to int for better performance
7          str_to_int = {}
8
9          for equation in equations:
10              u, v = equation
11              if u not in str_to_int:
12                  str_to_int[u] = len(str_to_int)
13              if v not in str_to_int:
14                  str_to_int[v] = len(str_to_int)
15
16          graph = [[] for _ in range(len(str_to_int))]
17          seen = [0] * len(graph)
18
19          for i, (u, v) in enumerate(equations):
20              u_idx = str_to_int[u]
21              v_idx = str_to_int[v]
22              graph[u_idx].append((v_idx, values[i]))
23              graph[v_idx].append((u_idx, 1 / values[i]))
24
25          def dfs(u: int, val: float) -> bool:
26              nonlocal graph, seen
27
28              if seen[u]:
29                  return abs(val / seen[u] - 1) > 1e-5
30
31              seen[u] = val
32              for v, w in graph[u]:
33                  if dfs(v, val / w):
34                      return True
35
36              return False
37
38          for i in range(len(graph)):
39              if not seen[i] and dfs(i, 1.0):
40                  return True
41
42          return False
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
```

## JavaScript Solution

```
1  javascript
2  class Solution {
3      checkContradictions(equations, values) {
4          // Convert string to int for better performance
5          const strToInt = new Map();
6
7          for (let equation of equations) {
8              const [u, v] = equation;
9              if (!strToInt.has(u))
10                  strToInt.set(u, strToInt.size);
11              if (!strToInt.has(v))
12                  strToInt.set(v, strToInt.size);
13          }
14
15          const graph = new Array(strToInt.size).fill(null).map(() => []);
16          const seen = new Array(graph.length).fill(0);
17
18          for (let i = 0; i < equations.length; ++i) {
19              const [u, v] = equations[i];
20              const uIdx = strToInt.get(u);
21              const vIdx = strToInt.get(v);
22              graph[uIdx].push([vIdx, values[i]]);
23              graph[vIdx].push([uIdx, 1 / values[i]]);
24          }
25
26          const dfs = (graph, u, seen, val) => {
27              if (seen[u])
28                  return Math.abs(val / seen[u] - 1) > 1e-5;
29
30              seen[u] = val;
31              for (let [v, w] of graph[u]) {
32                  if (dfs(graph, v, seen, val / w))
33                      return true;
34              }
35              return false;
36          }
37
38          for (let i = 0; i < graph.length; ++i) {
39              if (!seen[i] && dfs(graph, i, seen, 1.0))
40                  return true;
41          }
42
43          return false;
44      }
45  };
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
```

## Java Solution

```
1  java
2  import java.util.ArrayList;
3  import java.util.HashMap;
4  import java.util.HashSet;
5  import java.util.List;
6  import java.util.Map;
7
8  class Solution {
9      public boolean checkContradictions(List<List<String>> equations, double[] values) {
10          // Convert string to int for better performance
11          Map<String, Integer> strToInt = new HashMap<>();
12
13          for (List<String> equation : equations) {
14              String u = equation.get(0);
15              String v = equation.get(1);
16              strToInt.putIfAbsent(u, strToInt.size());
17              strToInt.putIfAbsent(v, strToInt.size());
18          }
19
20          List<List<Pair<Integer, Double>>> graph = new ArrayList<>(strToInt.size());
21          for (int i = 0; i < strToInt.size(); ++i)
22              graph.add(new ArrayList<>());
23          double[] seen = new double[graph.size()];
24
25          for (int i = 0; i < equations.size(); ++i) {
26              List<String> equation = equations.get(i);
27              String u = equation.get(0);
28              String v = equation.get(1);
29              int uIdx = strToInt.get(u);
30              int vIdx = strToInt.get(v);
31              graph.get(uIdx).add(new Pair<>(vIdx, values[i]));
32              graph.get(vIdx).add(new Pair<>(uIdx, 1 / values[i]));
33          }
34
35          for (int i = 0; i < graph.size(); ++i) {
36              if (seen[i] == 0 && dfs(graph, i, seen, 1.0))
37                  return true;
38          }
39
40          return false;
41      }
42
43      private boolean dfs(List<List<Pair<Integer, Double>>> graph, int u, double[] seen, double val) {
44          if (seen[u] != 0)
45              return Math.abs(val / seen[u] - 1) > 1e-5;
46
47          seen[u] = val;
48          for (Pair<Integer, Double> entry : graph.get(u)) {
49              int v = entry.getKey();
50              double w = entry.getValue();
51              if (dfs(graph, v, seen, val / w))
52                  return true;
53          }
54
55          return false;
56      }
57
58      static class Pair<K, V> {
59          private K key;
60          private V value;
61
62          public Pair(K key, V value) {
63              this.key = key;
64              this.value = value;
65          }
66
67          public K getKey() {
68              return key;
69          }
70
71          public V getValue() {
72              return value;
73          }
74      }
75  }
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
```



Level Up Your  
Algo Skills

Get Premium

Got a question? [Ask the Teaching Assistant](#) anything you don't understand.