Under-Specification #1

In prose, briefly describe the situation.

The original tables handout reads *if there are duplicate entries for the necessary conversion factor, you should raise an exception*. However, when considering the transitive case, a solution may include numerous edges, and such defining the result in this instance isn't as simple as returning an error upon querying the table. There may be multiple repeated terms in the conversion table, and we must consider when they should impact the solution, and when they shouldn't. We could return an error when there is a duplicate conversion anywhere in the table, or only when it is included in at least one of the possible paths, or finally only when it is included in all paths.

Create a small example (which would be a pair of tables, one for artwork and one for currency) that illustrates it.

```
Visualisation of the example. Each X represents a duplicate
edge, while the arrows connote the conversion direction.

#|

      (£)────
       |     \
       ↓      X
       |       \
      (฿)───>───($)
       |         |
       X         ↓
       |         |
      (₭)────<──(¥)
                  \
                   ↓
                    \
                    (₫)

|#
```

The data as it would be defined in Pyret.

An example conversion table.

```
repeated-unconnected =
  table: from-c :: String, to-c :: String, conv-rate :: Number
    row: "Pound", "Baht", 44.30      # GBP to THB
    row: "Pound", "Dollar", 1.29     # GBP to USD
    row: "Pound", "Dollar", 1.29     # x2
    row: "Baht", "Dollar", 0.029     # THB to USD
    row: "Dollar", "Yuan", 7.18      # USD to CNY
    row: "Yuan", "Kip", 3042.59      # CNY to LAK
    row: "Yuan", "Dong", 3376        # CNY to VND
    row: "Baht", "Kip", 635.16       # THB to LAK
    row: "Baht", "Kip", 635.16       # x2
  end
```

An example art table.

```
art-all-currencies =
  table: id :: Number, cost :: Number, currency :: String
    row: 1, 5000, "Pound"
    row: 2, 150000, "Baht"
    row: 3, 2000000, "Dong"
    row: 4, 1000, "Yuan"
    row: 5, 1500, "Dollar"
    row: 6, 200000, "Kip"
    row: 7, 2500000, "Baht"
  end
```

Restate the situation in terms of the example.

In this example, say we were trying to get the value of Artwork #1 in Dong. While there are two repeated conversions in the table, there is still a valid path between the two nodes, as we can see in the visualisation above. We could simply ignore the repeated terms and proceed with this valid path as normal. However, alternatively, we could return an error since there are repeated edges both in the conversion table, and in one of the potential paths.

Describe which outcome you chose, in prose and in terms of the example. If you have a particularly strong justification for why you made that choice, tell us. Don't worry, there are no secret "bad" choices for which you might be penalized; any choice you make that satisfies the problem specification is okay.

I opted for the straightforward approach of excluding all duplicate conversions from the representation that I used for my graph operations. If the removal of this edge meant there were no longer any valid paths through the graph, the function returns an error. But, if another valid path still exists, it just uses that one instead and proceeds as normal. I didn't include a specific error for repeated conversions, instead grouping it in with the general *a valid path could not be found* error. This is due to the significant additional complexity of distinguishing between the two.

Under-Specification #2

In prose, briefly describe the situation.

Considering the transitive case, there can now be practically different valid paths between the two currencies. They could go via totally different currencies, use the inverse of an edge, or instead follow it directly. We must consider how define the correct output in the situation were there is more than one correct path.

Create a small example (which would be a pair of tables, one for artwork and one for currency) that illustrates it.

Visualisation of the example. The arrows connote the conversion direction.

```
#|
      (£)────
       |     \
       ↕      ↓
       |       \
      (฿)───<>───($)
       |         |
       ↕         ↓
       |         |
      (₭)────<───(¥)
                  \
                   ↕
                    \
                    (₫)
|#
```

An example conversion table, as it would be defined in Pyret.

```
with-inverses =
  table: from-c :: String, to-c :: String, conv-rate :: Number
    row: "Pound", "Baht", 44.30      # GBP to THB
    row: "Baht", "Pound", 0.023      # +inv
    row: "Pound", "Dollar", 1.29     # GBP to USD
    row: "Baht", "Dollar", 0.029     # THB to USD
    row: "Dollar", "Baht", 34.37     # +inv
    row: "Dollar", "Yuan", 7.18      # USD to CNY
    row: "Yuan", "Kip", 3042.59     # CNY to LAK
    row: "Yuan", "Dong", 3376        # CNY to VND
    row: "Dong", "Yuan", 1/3376      # +inv
    row: "Baht", "Kip", 635.16       # THB to LAK
    row: "Kip", "Baht", 1 / 635.16   # +inv
  end
```

Plus using the same art-table as above.

Restate the situation in terms of the example.

Say we are again attempting to get the value of Artwork #1 in Dong. With this example test data, there many (around 10) possible distinct paths. Are they all correct or is anyone of them the defined “correct” answer.

Describe which outcome you chose, in prose and in terms of the example. If you have a particularly strong justification for why you made that choice, tell us. Don’t worry, there are no secret “bad” choices for which you might be penalized; any choice you make that satisfies the problem specification is okay.

There are minor benefits to having a shorter path, such as faster computation and error minimization due to the lesser number of calculations that need to be performed. However, the differences are minimal, and if the conversion rates are correct and consistent all paths should produce the same value. Hence I defined all paths through the graph as correct and simply used the first one encountered when using a depth first search.