Assignment 10

14CH3FP18

25.03.2016

Abstract

Finding arbitrage opportunities in currency conversion using the Bellman-Ford Algorithm

Part I
Introduction

Arbitrage is the use of discrepancies in currency exchange rates to transform one unit of a currency into more than one unit of the same currency. Suppose, 1 U.S. dollar bought 0.82 Euro, Euro bought 129.7 Japanese Yen,1 Japanese Yen bought 12 Turkish Lira, and one Turkish Lira bought 0.0008 U.S. dollars. Then, by converting currencies, a trader can start with U.S. dollar and buy 1.02 U.S. dollars, thus turning a $0.02 profit. Suppose that we are given n currencies nxn table of exchange rates, we should find if such a cyclic arbitrage opportunity exists.

Part II
Bellman Ford Algorithm

```
function BellmanFord(list vertices, list edges, vertex source)
    init distance[],predecessor[]
    // This implementation takes in a graph, represented as // lists of vertices and edges, and fills two arrays (distance and predecessor) with shortest-path (less cost/distance/metric) information
    // Step 1: initialize graph for each vertex v in vertices:
    distance[v]= inf
    // At the beginning , all vertices have a weight of infinity
    predecessor[v]= null
    // And a null predecessor
        distance[source] = 0
        // Except for the Source, where the Weight is zero
// Step 2: relax edges repeatedly for i from 1 to size(vertices)-1:
for each edge (u, v) with weight w in edges:
if distance[u] + w < distance[v]:
    distance[v] = distance[u] + w
    predecessor[v] = u
// Step 3: check for negative-weight cycles
for each edge (u, v) with weight w in edges:
    if distance[u] + w < distance[v]:
        error "Graph contains a negative-weight cycle"
    return distance[], predecessor[]
```

Part III

Using this in our Arbitrage Problem

A solution to the problem is achieved when the product of subsequents weights is >1. This can be converted to a BellMan ford problem by converting the edged weights to negative logs and then finding negative cycles in the graph.

Part IV

Complexity Analysis

It takes O(V^2) time to create graph from the table, which has V(V-1) edges. Then it takes O(n^3) time to run BELLMAN-FORD. Thus, the total time is O(n^3) .