Assignment #2

Professor Ahmad Namini
Python and Applications to Business Analytics Fall 2018, Module 1

September 25, 2018

Exercise 1. Below is the code for Dijkstra's Algorithm for computing the smallest cost along some path which is modeled as a weighted graph.

- Input consists of nodes and costs. The hard-coded input is that of the graph from your notes.
- Output is the variable *visited_nodes* which represents the least cost path from a given start node to other nodes.
- Please note that I have inserted commented print statements to follow the flow and data of the algorithm. Printing variable names during execution of code is a great way to understand the flow and data changes during the execution of your code. Another way is to use your IDE's debugger, but unfortunately, each IDE has a different flavor to what and how a debugger works. I would suggest investing time in understanding how your IDE's debugger works.

The foreign exchange (FX) market, where one currency is traded for another, is the largest financial market in the world, with about 5 trillion USD being traded every day. This market determines the exchange rate for local currencies when someone travels abroad and when international banks settle accounts between each other.

Let's model a currency exchange's FX rates as a weighted graph, where each node represents a currency (e.g. CNY, JPY, USD) and each edge represents the conversion of currency A to currency B. The weighted graph's edge cost is the FX conversion rate between the two end currencies.

At the following site, Wells Fargo shows current FX rates between many currencies and the US dollar. Dealers in the FX market will usually add or subtract a percentage of the FX rate depending on whether the client is selling a currency (the Bid rate) or buying a currency (the Ask rate).

An arbitrage condition exists when you can buy in one currency and sell in another currency and make a risk-less profit.

Using Dijkstra's Algorithm, do the following:

- Model the FX market rates as a weighted graph for the following currencies (CNY, JPY, EUR, USD, GBP, CAD)
- Model bid and ask rates as a percentage of FX rates. Use 1, 2, and 5 percents on the bid side, and 1, 2, and 5 percent on the ask side. This gives nine different scenarios.
- Check whether an arbitrage condition exists. (Think about how best to do this.)
- Make your output so that anyone requiring to understand the markets for arbitrage can understand from your output what the results are.

```
# Dijkstra's Algorithm
#
# An algorithm for finding the least cost path between nodes in a graph.
# It was conceived by computer scientist Edsger W. Dijkstra in 1956.

# network topology and costs
nodes = ('a', 'b', 'c', 'd', 'e', 'f')
costs = {'a': {'b': 7, 'c': 9, 'f': 14},
         'b': {'a': 7, 'c': 10, 'd': 15},
         'c': {'a': 9, 'd': 11, 'f': 2},
         'd': {'b': 15, 'c': 11, 'e': 6},
         'e': {'d': 6, 'f': 9},
         'f': {'a': 14, 'c': 2, 'e': 9}}
start_node = 'a'

# initialize for Dijkstra's algorithm
unvisited_nodes = {node: None for node in nodes}
visited_nodes = {}
current_node = start_node
current_cost = 0
unvisited_nodes[current_node] = current_cost

# apply algorithm
while True:
    # print("-----------------------------------")
    # print("unvisited to start: ")
    # print(unvisited_nodes)

    for neighbor, cost in costs[current_node].items():
        # print ("{0} {1} {2}".format(current_node, neighbor, cost))
        if neighbor not in unvisited_nodes:
            continue
        new_cost = current_cost + cost
        if unvisited_nodes[neighbor] is None or new_cost < unvisited_nodes[neighbor]:
            unvisited_nodes[neighbor] = new_cost

    # print("unvisited: ")
    # print(unvisited_nodes)
    visited_nodes[current_node] = current_cost
    # print("visited: ")
    # print(visited_nodes)
    del unvisited_nodes[current_node]
    if not unvisited_nodes:
        break
    candidates = [node for node in unvisited_nodes.items() if node[1]]
    current_node, current_cost = sorted(candidates, key = lambda x: x[1])[0]
    # print("candidates: ")
    # print(candidates)
    # print("current_node, current cost: ")
    # print(current_node, current_cost)

# print nodes with least costs
print(visited_nodes)
```