Solutions for Data Structures and Algorithms Spring 2023 — Problem Sets

By Dmitriy Okoneshnikov, B22-DSAI-04

April 23, 2023

Week 13. Homework 3

3.1 All-Pairs Shortest Paths (15 points)

Given a directed weighted graph G, perform the following:

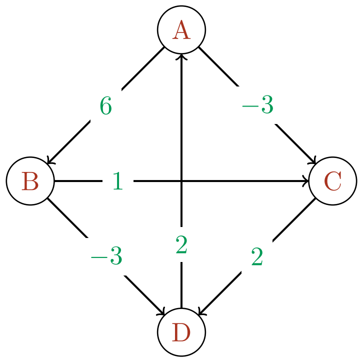

Figure 1: Graph G.

1. Write down representation of G as an adjacency matrix;

   Answer.

	A	**B**	**C**	**D**
A	0	6	-3	∞
B	∞	0	1	-3
C	∞	∞	0	2
D	2	∞	∞	0

2. Find lengths of shortest paths between every pair of vertices in G using Floyd-Warshall algorithm. Provide your solution by constructing a matrix after every iteration of the algorithm (by iteration here we mean, a choice of the intermediate vertex). The solution should have exactly 5 matrices (1 initial + 4 iterations).

   Answer.

   (a) Iteration 0:

	A	**B**	**C**	**D**
A	0	6	-3	∞
B	∞	0	1	-3
C	∞	∞	0	2
D	2	∞	∞	0

(b) Iteration 1:

	A	**B**	**C**	**D**
A	0	6	-3	∞
B	∞	0	1	-3
C	∞	∞	0	2
D	2	8	-1	0

(c) Iteration 2:

	A	**B**	**C**	**D**
A	0	6	-3	3
B	∞	0	1	-3
C	∞	∞	0	2
D	2	8	-1	0

(d) Iteration 3:

	A	**B**	**C**	**D**
A	0	6	-3	-1
B	∞	0	1	-3
C	∞	∞	0	2
D	2	8	-1	0

(e) Iteration 4:

	A	**B**	**C**	**D**
A	0	6	-3	-1
B	-1	0	-4	-3
C	4	10	0	2
D	2	8	-1	0

3.2 Longest Paths (10 points)

Consider a directed graph P in which every edge has a strictly positive weight.

Graph R is created from graph P by reciprocating every weight (e.g. weight 5 becomes $\frac{1}{5}$). For a given vertex s in R you compute all shortest paths from s to every other vertex in R using Dijkstra's algorithm.

Are the following statements `TRUE` or `FALSE`? Justify your answers (providing a proof or a counterexample).

1. Dijkstra's algorithm might not be applicable to find shortest paths in R (depending on original graph P).

   Solution.

   Dijkstra's algorithm always works when the graph has no negative weights. In the problem statement it is mentioned that all weights in P are strictly positive and, therefore, all weights in R are also strictly positive as we get their inverse. So, the statement is false.

   Answer. `FALSE`

2. If Dijkstra's algorithm is applicable, the resulting shortest paths from a vertex s in R correspond to the longest (i.e., with largest total weight) paths from s in P.

   Solution.

   Let us take a look into one shortest path from vertex s to v in graph R. The length of this path is equal to the sum of weights of some edges $w^R_{i,j}$ that are a part of this path: $w^R_{s,a}, w^R_{a,b}, ..., w^R_{z,v}$. Due to the problem statement: $W^Pi, j = \frac{1}{W^R i,j}$, Therefore, this path in graph P will look like this: $\frac{1}{w^R s,a}, \frac{1}{w^R a,b}, ..., \frac{1}{w^R z,v}$. But w^Ri, j is the smallest number, so the inverse will be the largest. Therefore, the statement is true.

   Answer. `TRUE`

3.3 Exchanging for Profit (25 points)

Sometimes, in the exchange market, it is possible to turn a profit by converting items or currencies in the correct order. For examples, support that one Apple share can be sold for 64 Indian rupees, one Indian rupee can buy 1.8 Japanese yen, and one yen can buy 0.009 of Apple shares. Then, by we can trade 1 Apple share for 64 Indian rupees, then trade 64 Indian rupees for 115.2 Japanese yen, which we then trade for 1.0368 Apple shares, turning a profit of 3.68%!

In this problem, you are given n different types of items or currencies $c_1, ..., c_n$ to trade with. You are also given an $n \times n$ matrix E of exchange rates, such that 1 unit of c_i can be exchanged for $E[i, j]$ units of c_j.

Perform the following:

1. Give an efficient algorithm to determine whether there exists a sequence of items $c_{k_1}, c_{k_2}, ..., c_{k_m}$ such that it is possible to turn a profit simply by exchanging items:

$$E[k_1, k_2] \cdot E[k_2, k_3] \cdot ... \cdot E[k_{m-1}, k_m] \cdot E[k_m, k_1] > 1$$

Answer.

$$E[k_1, k_2] \cdot E[k_2, k_3] \cdot ... \cdot E[k_{m-1}, k_m] \cdot E[k_m, k_1] > 1$$

$$\frac{1}{E[k_1, k_2]} \cdot \frac{1}{E[k_2, k_3]} \cdot ... \cdot \frac{1}{E[k_{m-1}, k_m]} \cdot \frac{1}{E[k_m, k_1]} < 1$$

$$e^{\ln \frac{1}{E[k_1,k_2]} + \ln \frac{1}{E[k_2,k_3]} + ... + \ln \frac{1}{E[k_{m-1},k_m]} + \ln \frac{1}{E[k_m,k_1]}} < e^0$$

$$\ln \frac{1}{E[k_1, k_2]} + \ln \frac{1}{E[k_2, k_3]} + ... + \ln \frac{1}{E[k_{m-1}, k_m]} + \ln \frac{1}{E[k_m, k_1]} < 0$$

Therefore, if we change every weight $E[i, j]$ to $\ln \frac{1}{E[i,j]}$ we can simplify the problem to finding a negative cycle. This can be done using Floyd-Warshall algorithm.

```
IsNegativeCycle($E$, $n$)
    for $i$:=1 to $n$:
        for $j$:=1 to $n$:
            if $E[i,j]$ != INF:
                $E[i,j]$ := log(1 / $E[i,j]$)
    for $k$:=1 to $n$:
        for $i$:=1 to $n$:
            for $j$:=1 to $n$:
                $E[i,j]$ := min($E[i,j]$, $E[i,k]$ + $E[k,j]$)
    for $i$:=1 to $n$:
        if $E[i,i]$ < 0:
            return true
    return false
```

2. Analyze the running time of your algorithm. Provide the worst case time complexity of the algorithm with justification.

Solution.

The very first update of the weights is $O(n^2)$. Then the main part of Floyd-Warshall algorithm is $O(n^3)$. The last check for negative cycles is $O(n)$. Therefore, the overall worst case time complexity of the algorithm is $O(n^3)$.

Answer. $O(n^3)$

Hint №1: reduce to a problem of finding a particular kind of cycle in a graph.
Hint №2: $x \cdot y = e^{\ln x + \ln y}$.