1 Introduction and Objective

In this tutorial, we will discuss the Single-Source Shortest Paths (SSSP) problem and discuss more about the ‘graph modeling’ soft skill, i.e. ability to model a seemingly random problem into a graph problem (SSSP problem for this tut07).

We will heavily use http://visualgo.net/sssp.html during our discussion.

SSSP problem is quite easily found in many real life applications and it is the source of many interesting Computer Science problems, as you can see in this tut07. Again, we recommend that you put some thoughts on them before discussing the potential solutions during tut07 with your tutor.
2 Tutorial 07 Questions

Standard Stuffs

Q1. In Lecture 08, you are presented with two SSSP algorithms: BFS (only for unweighted graph) and Bellman Ford’s algorithm (general case). First, the tutor will (re-)demonstrate the executions of BFS and Bellman Ford’s on a small directed weighted graph using http://visualgo.net/sssp.html from a certain source vertex \( s \). The tutor will re-explain that BFS will only work if the graph is unweighted, or... all edges have the same weight. The tutor may invite some students to do this live demonstration using different source vertex \( s \) and/or using different graph.

This part is left to the tutor and should not take too much time. There is one thing that the tutor will highlight: That Bellman Ford’s algorithm can be ‘optimized’ a bit from \( O(VE) \) to \( O(kE) \) where \( k \) is the minimum number of rounds plus one that are needed to actually solve the SSSP problem. The maximum \( k \) is still \( V-1 \), but for many cases, it can be much smaller than \( V-1 \). This feature has been integrated in http://visualgo.net/sssp.html just that Steven purposely did not cover it in details during Lecture 07.

Q2. There is one very subtle bug inside http://www.comp.nus.edu.sg/~stevenha/cs2010/demo/BellmanFordDemo.java that is currently left there so that we can discuss the bug here in this class. That subtle bug only appears when the input graph contains negative weight edge. Explain what happen when that BellmanFordDemo.java is run on this graph.

![Figure 1](image)

Figure 1: Source vertex 0 is disconnected from the rest; change weight(1, 2) from -1 to -1B (10^9)

VisuAlgo limits the edge weight to 2 digits for visual aesthetic purpose, but you can try this test case on BellmanFordDemo.java and it will yield funny answers: SSSP(0, 0) = 0 (correct), SSSP(0, 1) = 1000000000 (correct), SSSP(0, 2) = 0 (wrong...). To fix this, we have to modify the relax function a bit so that we do not do relaxation if \( D.get(u) == \text{INF} \) and modify the Cycle Check a bit to take into account this same special case. Steven will update Lecture Note 07 and BellmanFordDemo.java for next year’s version (as well as sample BellmanFord’s code in CP3) to remove this subtle bug. Basically, if an INF is involved in a computation, the result should be an INF too.

Graph Modeling Exercises


Key ideas only, detailed discussions by tutor:
• Vertices = ‘.’ cells in the grid (we can replace the one cell of ‘Y’ and a few cells of ‘F’ to ‘.’ too after recording their positions). There are up to $R \times C$ of them.

• Edges = If there is another ‘.’ cell in the N/S/E/W direction of the current ‘.’ cell, then there is an edge between these 2 cells. These edges are unweighted. There are up to 4 edges per cells.

• This is a (Multi-Sources) Shortest Paths on unweighted graph. There are actually more than one source in this graph, the position of ‘Y’ and the positions of all those ‘F’ s. This is the key highlight that the tutor will explain more in class that we are not just restricted to Single-Source default variant at all times.

• We can run BFS by enqueuing all these sources (‘Y’ and those ‘F’s) first.

• You are safe if you can reach the border before any fire, or you die otherwise. Both you and fire spread at a rate of 1 cell at a time, which is what BFS computes.

Analysis: After running BFS algorithm from multiple sources, all ‘.’ cells in the map are eventually replaced by a ‘Y’ or an ‘F’. This means each of the $R \times C$ cells has been processed at most once in the simulation. Since for each cell we at most consider all 4 directions, the total processing of cells is $4 \times R \times C$. In overall, this algorithms is $O(R \times C)$. Btw, the answer for the last bonus question is 9 minutes and will need careful checks to get the answer right using just manual calculations.


Key ideas only, detailed discussions by tutor:

• Each currency is as a vertex

• There is a directed edge from vertex $u$ to vertex $v$ if you can exchange the currency represented by vertex $u$ to the currency represented by vertex $v$. The edge weight is the exchange rate.

• In this graph, we want to find at least one ‘profitable’ cycle, that is if we multiply all edge weight in such a cycle, the result $> 1$. If you understand this part, you can modify the standard Bellman Ford’s algorithm to check for such ‘profitable cycle’ this way. But if you are not sure, read on.

• Now for any given profitable cycle $u_1 \sim u_1$,
  weight($u_1, u_2$) $\times$ weight($u_2, u_3$) $\times$ ... $\times$ weight($u_i, u_1$) $> 1$ iff
  $\log(\text{weight}(u_1, u_2)) + \log(\text{weight}(u_2, u_3)) + ... + \log(\text{weight}(u_i, u_1)) > 0$.

• Now if we negate the logarithms, we have:
  $-\log(\text{weight}(u_1, u_2)) - \log(\text{weight}(u_2, u_3)) - ... - \log(\text{weight}(u_i, u_1)) < 0$.

• So if we transform the exchange rate to the negative of its logarithm and use it to represent the edge weights, a ‘profitable’ cycle will be a negative cycle. From here, the answer becomes ‘trivial’.
Using the standard $O(VE)$, or in this case $O(nm)$ Bellman Ford’s algorithm, we can easily detect whether such a cycle exists by running 1 more iteration after the $(V-1)$-th iteration and checking if any vertex has its shortest path distance relaxed. If there is, there is a profitable cycle. Done.

Note that in real life, it is not really possible to gain profit using this technique due to the difference of buying and selling rates of each currency. Doing this will just make the money changer happier (and you simply lose your money).

Problem Set 4, Continued

That’s the end of tut07. You can ask your tutor about PS4 that will due on Saturday, 17 October 2015, 07.59am if you still encounter any difficulties.