Shortest Path In A Graph

Rosu Adriana-Stefania
rosustefania10@yahoo.com

University Politehnica Bucharest
Faculty of Automatic Control and Computers

Keywords: Shortest Path · All to all · Dijkstra · Floyd-Warshall · Johnson.

1 Introduction

1.1 The Problem Statement

The shortest path problem is the problem of finding a path between two vertices (or nodes) in a graph such that the sum of the weights of its constituent edges is minimized.

There is the single-source shortest path problem - we have to find shortest paths from a source node to all other nodes in the graph.

There's also the single-destination shortest path problem - we have to find shortest paths from all nodes in the directed graph to a single destination node.

And last, but not least, the single-pair shortest path - we have to find shortest paths between the given pair of nodes in the graph - and the all-pairs shortest path problem - we have to find shortest paths between every pair of nodes in the graph.

In this paper, we will solve the APSP (all-pairs shortest path) problem. Let G be a graph with the set of nodes V and the set of edges E. Each edge e has a weight w assigned. The goal of the all-pair-shortest-paths problem is to find the shortest path between all pairs of nodes of the graph.

1.2 Example of practical applications for the chosen problem

Applications for the shortest paths algorithms are numerous, including: finding directions between locations (for example, Google Maps is based on Dijkstra's Algorithm), finding a solution that uses the minimum possible number of moves (for example, solving a Rubik's Cube or a word ladder puzzle), it's also used in Social Network Analysis (usually people with the strongest bonds tend to communicate through the shortest path) and in telecommunications networks.

1.3 Solutions Overview

Some of the most known algorithms that solve the all-pairs shortest path problem are Floyd-Warshall Algorithm, Dijkstra's Algorithm and Johnson's Algorithm. The first one to publish one of these algorithms was Dijkstra, in 1959.

The Floyd-Warshall Algorithm was first published in 1959 by Bernand Roy and then by Robert Floyd and Stephen Warshall in 1962. Donald B. Johnson published his tehnique based on Bellman-Ford and Dijkstra's Algorithms in 1977.

The studied solutions will be the following:

- using Dijkstra's Algorithm:

This algorithm only works on a directed/undirected graph with positive weighted edges.

Dijkstra's algorithm finds the shortest path between a node and all other nodes in a graph, so we have to apply it for each of the n nodes as the source node.

The algorithm follows a greedy approach (always mark the closest node) and a dynamic programming approach (distances are updated using previously calculated values) to find the shortest paths.

For its implementation we will create a shortest path tree set that keeps track of nodes included in shortest path tree, whose minimum distance from source is calculated and finalized (initially, this set is empty) and then we will assign a distance value to all nodes in the input graph (initially all distance values are set as infinite, except for the source node - its distance value will be 0 so it can be picked first).

- using Floyd-Warshall Algorithm:

This algorithm can be applied on both directed and undirected graphs with both positive and negative weighted edges, but with no negative cycles (the sum of the weights in a cycle must be positive).

The algorithm follows the dynamic programming approach to find the shortest paths and compares all possible paths through the graph between each pair of nodes.

- using Johnson's Algorithm:

For this algorithm, our graph has to be directed. It can have negative weighted edges, but no negative cycles.

Johnson's Algorithm uses both Dijkstra and Bellman-Ford as subroutines. Bellman–Ford is used for removing all the negative weights from the input graph(if this step detects a negative cycle, the algorithm is terminated), and then Dijkstra's Algorithm is applied on the transformed graph.

1.4 Evaluation Criteria

For testing these algorithms, their rightness and their efficency, the input will represented by multiple graphs.

The algorithms will be applied on both directed and undirected graphs. In

this case, Dijkstra's and Floyd-Warshall should return the correct output, while Johnson's Algorithm should work only on directed graphs.

Some of the input graphs will have negative weighted edges, but no negative cycles. In this case, Floyd-Warshall Algorithm and Johnson's Algorithm should work, but not Dijkstra's.

In the input set, will also be some graphs with negative weighted edges and negative cycles and none of the algorithms should work.

2 The solutions

2.1 Algorithms' description

Dijkstra's Algorithm:

Dijkstra's algorithm solves the single-source shortest path problem, so to solve all-pairs shortest path problem it must be applied for all the nodes in the graph.

1. Marks all nodes as unvisited.
2. Marks the source node (current node) with a current distance of 0 and the rest with infinity.
3. Calculates the tentative distances through the current node for all the unvisited nodes (distance of current + weight of the corresponding edge), compare and assign the smaller one.
4. Marks the current node as visited.
5. Continues for all the nodes until they are all visited.

Dijkstra's Algorithm

```
1: procedure DIJKSTRA'S
2:     for v ∈ G.v do
3:         for u ∈ G.v do
4:             distance[v] = inf
5:             visited[v] = false
6:             distance[v] = 0
7:             for x ∈ neighbors[u] do
8:                 if dist[v][u] + weight(u, x) < dist[v][x] & visited[x] = false
   then
9:                     dist[v][x] = dist[v][u] + weight(u, x)
10:                    visited[x] = true
       return distance
```

Floyd-Warshall Algorithm:

1. Creates weighted adjacency matrix that has 0 on the main diagonal and infinity for the other elements.

2. Considers every node (one by one) as an intermediate node.

3. Updates all shortest paths which include the picked node as an intermediate node in the shortest path matrix. (When we pick node number k as an intermediate node, we already have considered nodes 0, 1, 2, .. k-1 as intermediate nodes.)

4. If k is not an intermediate node in path from the source node (matrix' line i) to the destination node (matrix' column j), then the value (matrix[i][j]) remains the same.

If it is and matrix[i][k] + matrix[k][j] is smaller than the assigned value (matrix[i][j]), then update the value as matrix[i][k] + matrix[k][j].

Floyd-Warshall Algorithm

1: **for** $k = 0 \rightarrow n$ **do**
2: **for** $i = 0 \rightarrow n$ **do**
3: **for** $j = 0 \rightarrow n$ **do**
4: **if** $weight(i, j) > weight(i, k) + weight(k, j)$ **then**
5: $weight(i, j) > weight(i, k) + weight(k, j)$
return *weigth*
6:

Johnson's Algorithm:

The algorithm uses weighted adjacency matrix, where matrix[i][j] represents the edge's weight from node i to node j.

1. Adds a new node to the graph that is connected with every other nodes by edges that have the weigth 0.

2. Using Bellman-Ford algorithm (that calculates the shortest path from the added node to all the others and stores it in an array h), verifies if the graph has negative weighted cycles. If it has, then both algorithms (Bellman-Ford and Johnson's) stop. If it doesn't, it reweigths the graph so it doesn't have negative weighted edges (matrix[i][j] = matrix[i][j] + h[i] - h[j]).

3. Removes the added node.

4. Applies Djisktra's algorithm for every node.

5. For the shortest path for every pair of nodes (i, j) adds back h[j] and substract back h[i].

Johnson's Algorithm

```
1: procedure JOHNSON'S
2:     G.V ← G.V + s
3:     for u ∈ G.E do
4:         weight(s, u) = 0
5:         G.E ← G.E + (s,u)
6:     if Bellman-Ford(s) = False then return The input graph has a negative
   weight cycle
7:     else
8:         for v ∈ G.V do
9:             h(v) ← distance(s, v)
10:        for edge(u, v) ∈ G.E do
11:            weight(u, v) ← weight(u, v) + h(u) − h(v)
12:        G.V ← G.V − s
13:        for u ∈ G.E do
14:            G.E ← G.E − (s,u)
15:        for u ∈ G.V do
16:            runDijkstra(G, weight, u)
17:            for v ∈ G.V do
18:                weight(u, v) ← weight(u, v) + h(v) − h(u)
       return weigth
```

2.2 Complexity

Let V be the number of nodes and E the number of edges in a graph.

Dijkstra's Algorithm:

-Best case complexity:

Best case complexity is obtained when implementing using a Fibonacci Heap and an adjacency list. In this case the complexity is O(E + V * log (V)).

Because for all pairs shortest paths problem Dijkstra's algorithm is used V times, the final complexity is O($E * V + V^2 * log(V)$).

-Worst case complexity:

Worst case complexity is obtained when using an array to store minimum distances from a node to the others. In this case the complexity is O(V^2).

Because for all pairs shortest paths problem Dijkstra's algorithm is used V times, the final complexity is O(V^3).

-Original complexity:

The original algorithm implementation uses a min priority queue and has $O(V + E * log(V))$ complexity. For APSP, original complexity is $O(V^2 + V * E * log(V))$.

Floyd-Warshall Algorithm:

Best case complexity is equal with worst case complexity which is $O(V^3)$.
That's because each of the three for loops has an O(V) complexity and updating the minimum distance takes O(1).

Johnson's Algorithm:

Johnson's algorithm complexity is based on Dijkstra's algorithm's complexity and Bellman Ford algorithm's complexity.
Bellman Ford algorithm's complexity is $O(E * V)$ because for every node iterates through all the edges.
The complexity when Dijkstra's algorithm's is implemented using a Fibonacci Heap is $O(E * V + V^2 * log(V))$. (average complexity)
When Dijkstra's algorithm's is implemented using a min priority queue the complexity is $O(E * V * log(V))$.
The Johnson's algorithm works faster for sparse graphs than Floyd-Warshall algorithm.

2.3 Advantages and disadvantages

Dijkstra's Algorithm:

Advantages:
-works on both directed and undirected graphs;
-is used in many applications;

Disadvantages:
-doesn't work on negative weighted graphs;
-a single execution of the algorithm is not sufficient to find the lengths of the shortest paths between all pairs of nodes;
-it does a blind search there by consuming a lot of time waste of necessary resources;

Floyd-Warshall Algorithm:

Advantages:
-works on both directed and undirected graphs;
-works on negative weighted graphs;

-a single execution of the algorithm is sufficient to find the lengths of the shortest paths between all pairs of nodes;
-is easy to write its code;

Disadvantages:
-it's not the fastest algorithm;
-it can find the shortest path only when there are no negative cycles;

Johnson's Algorithm:

Advantages:
-works on both directed and undirected graphs;
-works on negative weighted graphs;
-a single execution of the algorithm is sufficient to find the lengths of the shortest paths between all pairs of nodes;
-it has the best time complexity when using sparse graphs;

Disadvantages:
-it can find the shortest path only when there are no negative cycles;

3 Evaluation

3.1 Test Generation

For the evaluation of the validity and practical performance of the algorithms has been used a C++ program that can generate positive and negative weighted directed graphs.

Were generated graphs that have minimum four nodes and four edges and maximum 50 nodes and 50 edges.

The set of tests contains:
- positive weighted directed graphs;
- negative weighted directed graphs with no negative cycles;
- negative weighted directed graphs with negative cycles.

3.2 Test Environment

The code has been compiled using g++ compiler and run on a system with Ubuntu 20.04.1 LTS 64-bit, Intel® Core™ i7-7700HQ processor, 8 GB of RAM and SDD drive.

3.3 Results and explanations

Table 1. Dijkstra's result

Test number	Number of nodes	Number of edges	Dijkstra's algorithm's result
1	4	4	The correct shortest paths matrix.
2	4	7	The graph has negative weighted edges.
3	4	4	The graph has negative weighted edges.
4	5	6	The graph has negative weighted edges.
5	8	8	The correct shortest paths matrix.
6	4	6	The graph has negative weighted edges.
7	8	9	The graph has negative weighted edges.
8	6	12	The graph has negative weighted edges.
9	14	27	The graph has negative weighted edges.
10	32	41	The correct shortest paths matrix.

Table 2. Floyd-Warshall result

Test number	Number of nodes	Number of edges	Floyd-Warshall algorithm's result
1	4	4	The correct shortest paths matrix.
2	4	7	The correct shortest paths matrix.
3	4	4	The graph has negative cycles.
4	5	6	The correct shortest paths matrix.
5	8	8	The correct shortest paths matrix.
6	4	6	The correct shortest paths matrix.
7	8	9	The correct shortest paths matrix.
8	6	12	The correct shortest paths matrix.
9	14	27	The graph has negative cycles.
10	32	41	The correct shortest paths matrix.

Table 3. Johnson's result

Test number	Number of nodes	Number of edges	Johnson's algorithm's result
1	4	4	The correct shortest paths matrix.
2	4	7	The correct shortest paths matrix.
3	4	4	The graph has negative cycles.
4	5	6	The correct shortest paths matrix.
5	8	8	The correct shortest paths matrix.
6	4	6	The correct shortest paths matrix.
7	8	9	The correct shortest paths matrix.
8	6	12	The correct shortest paths matrix.
9	14	27	The graph has negative cycles.
10	32	41	The correct shortest paths matrix.

Table 4. Tests' summary

Algorithm	Number of tests	Failed tests	Succes rate	Execution time
Dijkstra's Algorithm	10	7	30%	0m0,003s
Floyd-Warshall Algorithm	10	2	80%	0m0,005s
Johnson's Algorithm	10	2	80%	0m0,004s

A test is failed when the algorithm can calculate the shortest path between every two nodes because the graph has negative weighted edges or it has negative cycles.

The results are as expected. Dijkstra's algorithm doesn't work on negative weighted graphs, while Floyd Warshall algorithm and Johnson's algorithm don't work on negative cycled graphs.

4 Conclusions

Is it observed that Dijkstra's algorithm is the fastest one, so it's the best solution for positive weighted graphs, both directed and undirected. Applied on negative weighted graphs, this algorithm can gives us wrong numbers.

For negative weighted graphs, Johnson's algorithm seems to be a better solution than Floyd-Warshall algorithm considering the time complexity and the obtained runtime, so it's the most efficient.

References

1. https://www.geeksforgeeks.org/johnsons-algorithm/ - last accessed 12/17/2020
2. https://www.programiz.com/dsa/floyd-warshall-algorithm/ - last accessed 11/09/2020

3. https://www.geeksforgeeks.org/shortest-path-in-a-directed-graph-by-dijkstras-algorithm/ - last accessed 12/17/2020
4. https://www.geeksforgeeks.org/floyd-warshall-algorithm-dp-16/ - last accessed 12/17/2020
5. Mariusz Glabowski, Bartosz Musznicki, Przemyslaw Nowak and Piotr Zwierzykowski. *Efficiency Evaluation of Shortest Path Algorithms.* [*AICT 2013 : The Ninth Advanced International Conference on Telecommunications*].
6. https://inst.eecs.berkeley.edu/cs61bl/r//cur/graphs/dijkstra-algorithm-runtime.html?topic=lab24.topicstep=4course= - last accessed 12/17/2020
7. https://www.geeksforgeeks.org/comparison-dijkstras-floyd-warshall-algorithms/ - last accessed 12/17/2020
8. https://brilliant.org/wiki/johnsons-algorithm/ - last accessed 12/17/2020
9. https://iq.opengenus.org/dijkstras-algorithm-finding-shortest-path-between-all-nodes/ - last accessed 12/17/2020