Commentary - Methods to find all the edges on any of the shortest paths between two given nodes of a directed acyclic graph

Deepak Tatyaji Ahire[‡], Omkar Sanjay Jadhav[§]

‡ University of California, San Diego, United States of America
§ Google LLC, Banglore, India

Corresponding author: Deepak Tatyaji Ahire (ahiredeepak20@gmail.com)

Abstract

This article puts forth all the existing methods proposed by the various authors of the Stack Exchange community to find all the edges on any shortest path between two given nodes of a directed acyclic graph. For a directed acyclic graph with ***N*** number of nodes, an exponential number of paths are possible between any two given nodes and, thus, it is not feasible to compute every path and find the shortest ones in polynomial time to generate a set of all edges that contribute or make any of the shortest paths. The methods discussed in this article are not limited only to this specific use case, but have a much broader scope in graph theory, dynamic programming and counting problems. Generally, various other questions and answers, raised on the community portal having similar scope to those that the users specifically seek, do not receive sufficient hits and, hence, enough attention and votes for various reasons worth contemplating. Therefore, this article also aims to highlight the various scopes of the methods discussed in this article and acknowledge the efforts of the authors, moderators and contributors of the Stack Exchange community for their expertise and time to write precise answers and share their opinions and advice. Finally, it also appeals to all the other beneficiaries in the community to use their privileges responsibly and upvote the posted answers, if they helped solve their queries, as one upvote is free of cost.

Keywords

Shortest-path, Dijkstra, Single-source-shortest-path, All-edges-on-any-shortest-path, Directed-acyclic-graph, Stack-Exchange-community, All-possible-shortest-paths, Stack-Exchange-privileges, Upvote

Introduction

In this article, we have discussed all the existing methods proposed by the various authors of the Stack Exchange community to find all the edges on any of the shortest paths between two given nodes of a directed acyclic graph. For a directed acyclic graph with ***N*** number of nodes, an exponential number of paths are possible between any two given nodes and, thus, it is not feasible to compute every path and find the shortest ones in polynomial time to generate a set of all edges that contribute or make any of the shortest paths. The methods discussed in this article are not limited only to this specific use case, but have a much broader scope in graph theory, dynamic programming and counting problems.

The ranking of any page on the search engine depends on several factors and it is not significantly affected by the number of upvotes or downvotes received for a particular answer (Buchanan 2013). Many questions are too specific to the askers' need, because of which various answers or methods posted on another page that have similar applications to those that the users specifically seek, do not receive sufficient attention and, hence, not enough page hits and votes.

Currently, there is no reliable mechanism to link any answer posted on the Stack Exchange page to different use cases other than the question on that page which, in turn, reduces the chances of these answers being referenced on various other pages with relevant content and hence resulting in a lower page rank. As content is also one of the critical aspects of search engine optimisation (Rego 2010), this article not only draws attention to and analyses the various methods proposed by the authors of the Stack Exchange community, but also discusses the other different use cases where these methods can be utilised.

Finally, it acknowledges the time and the efforts of the moderators and other contributors to write precise answers and share their advice and opinions. Several new users who do not have accounts on the Stack Exchange community access these helpful posts, but leave without expressing their feedback and votes (SMR 2014). The other novice users with valid accounts who wish to, not have enough privileges to upvote or comment (Park 2020). Therefore, finally, it appeals to all the beneficiaries in the community to use their privileges responsibly and upvote the answers posted by the authors, if the answers helped solve their queries, as one upvote is free of cost.

Abbreviations

Table 1 lists all the abbreviations used in the article.

Existing Approaches

The following are the existing approaches:

Approach 1: Computing all possible paths between two nodes to find all the shortest paths

Computing all possible paths and sorting them to find all the shortest paths is a brute force approach and is not the feasible solution to execute the program in polynomial time, as the maximum number of paths in a directed acyclic graph can be exponential and the runtime will be of the order of $\Omega(N + (P * \log(P)))$ (D.W. 2015a, Nicholas Mancuso 2015).

Rather than computing all the paths, one might be tempted to only focus on the shortest paths. However, in the worst case, the number of shortest paths also can be exponential. Stack Exchange users ***Raphael*** and ***unkulunkulu*** proposed an example in which all the possible paths between ***S*** and ***D*** are the shortest and their number is exponential. In both the examples, the number of paths is of the order O(2^***N***) and, therefore, it was clear that ***P*** will dominate this approach (unkulunkulu 2013, Reitzig 2019).

Raphael also provided another example of a graph with an exponential number of shortest paths (Reitzig 2017).

Other Similar Applications

- Computing the total number of paths between two given nodes in linear time of O(***N*** + ***M***) using dynamic programming and the basic counting principle (Nicholas Mancuso 2015, Willcock 2011).
- Railway station routing algorithm using the backtracking method (Catrina 2015).

Approach 2: Using the Floyd Warshal Algorithm

Computer Science Stack Exchange user ***Draconis*** proposed an approach using the Floyd Warshal algorithm. The first step is to compute all pair shortest paths. Then, for every edge **e**(**U**, **V**), where **e** $\in$ ***E*** and **U**, **V** $\in$ ***G***, check if it satisfies the following equation:

Dist[S][U] + |**e**(**U**, **V**)| + ***Dist[V][D]*** = ***Dist[S][D]*** ..... eq(1)

For equation 1, |**e**(**U**, **V**)| represents the edge weight of the directed edge **e**(**U**, **V**).

If the edge **e** $\in$ ***E*** satisfies equation 1, then edge **e** belongs to the shortest path.

This approach has a runtime of the order of O(***N***^3 + ***M***) (Draconis 2018).

Other Similar Applications

- Math Overflow user ***Robert Israel*** discussed a similar approach to find all edges not covered by the shortest path in an all-pairs shortest path over a subset of vertices (Israel 2017).
- To find all the edges that are not part of any shortest path (Singh 2015).

- To check if a vertex is along some possible shortest path from ***S*** to ***D*** (*D.W. 2015b*).

Approach 3: Using Dijkstra's algorithm twice, starting from source to the destination and vice-a-versa

Computer Science Stack Exchange moderator ***D.W.*** proposed an approach which uses a single-source shortest paths algorithm (e.g. Dijkstra's algorithm) twice, ones from ***S*** to all the nodes and then from ***D*** (treating it as the source) to all other nodes after reversing the edges of the graph (D.W. 2015a, Discrete lizard 2019).

Suppose the single source shortest path distances from ***S*** to all other nodes are stored in ***Dist{S}*** and the single-source shortest path distances from ***D*** to all other nodes are stored in ***Dist{D}***, then, for every edge **e**(**U**, **V**), where **e** ∈ ***E*** and **U**, **V** ∈ ***G***, check if it satisfies any one of the following two equations:

Dist{S}[U] + |**e**(**U**, **V**)| + ***Dist{D}[V]*** = ***Dist{S}[D]*** ..... eq(2)

and

Dist{S}[U] + |**e**(**U**, **V**)| + ***Dist{D}[V]*** = ***Dist{D}[S]*** ..... eq(3)

For equation 2 and equation 3, |**e**(**U**, **V**)| represents the edge weight of the directed edge **e**(**U**, **V**).

If the edge **e** ∈ ***E*** satisfies equation 2 or equation 3, then edge **e** belongs to the shortest path.

This approach has the runtime of the order O (2****M*** + ***N*** * log(***N***)) + ***M***) ~ O(***M*** + ***N*** * log(***N***))) when the Fibonacci Heap is used to implement the Dijkstra's algorithm (Sneyers et al. 2006).

Other Similar Applications

- Bidirectional Dijkstra (Vaira and Kurasova 2011).

Approach 4: Using Dijkstra's algorithm from source to destination and BFS from destination to source

Maintaining a set of all the parent nodes (the node from which the arrow is directed is a parent node) for each node, if it is reachable with the minimum possible distance from source ***S*** via those particular parent nodes is the most important technique used in this approach. This technique helps reducing one extra call to Dijkstra's algorithm in the reverse direction from ***D*** (acting as source), as discussed in the previous approach.

While applying the relaxation process for each edge, if the path's cost to reach a particular node from source ***S*** via any of its parent nodes improves, then that parent node becomes part of the node's new beneficial parents set and the previous set, if any, is discarded. If

the path's cost to reach a particular node from source ***S*** via any of its parent nodes remains the same as the previously computed minimum cost, then that parent node is just added to the node's previous beneficial parents set.

Now, a new graph is constructed using each nodes' beneficial parents set. Unweighted edges from all the nodes, directed towards all of their respective beneficial parent nodes, are added.

Fetch all the edges encountered during the reverse BFS (starting from ***D*** as the source towards ***S*** as the destination) on the new graph. The reverse of these edges represents all the edges on any shortest paths in the original graph.

Stack Overflow user ***Ziyao Wei*** proposed this approach (Ziyao Wei 2013).

This approach has the runtime of the order O((***M*** + ***N*** * log(***N***)) + (***M*** + ***N***)) ~ O(***M*** + ***N*** * log(***N***)).

Other Similar Applications

- Printing paths in Dijkstra's shortest path algorithm (Contributors to Wikimedia projects 2021, Goel et al. 2019).

Appeal

From most of the references mentioned, it is evident that there is a considerable gap between the number of views and the number of upvotes. Less than 10% of the viewers have voted for the answers. Most of the hits on the community pages are because of search engine recommendations dependent on many factors. Several new beneficiaries without an account on the community portal access these helpful posts and leave without expressing their feedback and votes (SMR 2014). The other novice users with valid accounts who wish to, do not have enough privileges to upvote or comment (Park 2020). The Stack Overflow is the most challenging community for new users because it also attracts many antisocial developers. There is no hard and fast rule to upvote or downvote an answer and it is entirely perspective-based. Upvoting and accepting answers is also influenced by the trust the users put into authors, based on the authors' reputation. Furthermore, from **Approach 4** discussed in the previous section, it is clear that the best methods to solve the problem may not always be found on the most relevant pages. These factors make it very difficult to expect a certain number of votes on any particular post.

Therefore, this article also appeals to all the beneficiaries to use their privileges responsibly and the experienced users to encourage others to write and improve answers instead of a downvote. Recently, the Stack community also started encouraging the users to vote for the helpful posts and experimented with the features like reaction buttons as a way for users to thank the authors (Park 2020). Ultimately, this will help the search engines bring the content to the top (Stack Overflow 2021).

Conflicts of interest

The authors declare no conflict of interest.

References

- Buchanan A (2013) How does a down vote affect SEO for stack overflow? META Stack Exchange. URL: https://meta.stackexchange.com/a/173930/354413
- Catrina M (2015) Railway station routing algorithm using the backtracking method. University Politehnica of Bucharest Scientific Bulletin 77 (4): 335-346. [In English]. URL: https://www.scientificbulletin.upb.ro/rev_docs_arhiva/fulla34_189839.pdf
- Contributors to Wikimedia projects (2021) Dijkstra's algorithm - Wikipedia. Wikimedia Foundation, Inc.. URL: https://en.wikipedia.org/wiki/Dijkstra%27s_algorithm
- Discrete lizard (2019) Finding all edges on any shortest path between two nodes using Dijkstra. https://cs.stackexchange.com/q/103209/78734
- Draconis (2018) Finding all edges on any shortest path between two nodes. Computer Science Stack Exchange. URL: https://cs.stackexchange.com/q/93722
- D.W. (2015a) Finding all paths between a set of vertices in a DAG. Computer Science Stack Exchange. URL: https://cs.stackexchange.com/q/42043
- D.W. (2015b) How to find all shortest paths between two nodes in a weighted undirected graph? https://cs.stackexchange.com/a/48658
- Goel A, Kumar A, Rajan H (2019) Printing Paths in Dijkstra's Shortest Path Algorithm. GeeksforGeeks. URL: https://www.geeksforgeeks.org/printing-paths-dijkstras-shortest-path-algorithm/#:~:text=Value%20of%20parent%5Bv%5D%20for,path%20using%20below%20recursive%20function.
- Israel R (2017) Find all edges not covered by a shortest path in an all-pairs shortest path over a subset of vertices. MathOverflow. URL: https://mathoverflow.net/q/275681
- Nicholas Mancuso (2015) Algorithm that finds the number of simple paths from s to t in G. Computer Science Stack Exchange. URL: https://cs.stackexchange.com/q/3087
- Park L (2020) Saying thanks: testing a new Reactions feature. Stack Overflow Blog. URL: https://stackoverflow.blog/2020/06/17/saying-thanks-testing-a-new-reactions-feature/
- Rego C (2010) SEO in Stack Overflow. META Stack Exchange. URL: https://meta.stackexchange.com/a/42105/354413
- Reitzig R (2017) Example of graph with exponential many s-t minpaths and min cuts. Computer Science Stack Exchange. URL: https://cs.stackexchange.com/q/33932
- Reitzig R (2019) Optimal algorithm to traverse all paths in the order of shortest path. https://cs.stackexchange.com/q/19778. Accessed on: 2019-1-19.
- Singh R (2015) How do I find all the edges that don't lie on any of the shortest path? Stack Overflow. URL: https://stackoverflow.com/q/32630422
- SMR (2014) Why does it seem so hard to accumulate upvotes on Stack Overflow? Meta Stack Overflow. URL: https://meta.stackoverflow.com/a/254147
- Sneyers J, Schrijvers T, Demoen B (2006) Dijkstra's Algorithm with Fibonacci Heaps: An Executable Description in CHR. In: Fink M, Tompits H, Woltran S (Eds) 20th Workshop

on Logic Programming, Vienna, Austria, February 22--24, 2006, 1843-06-02. Vienna, Austria, February 22--24, 2006. Technische University at Wien, Austria INFSYS Research Report, 10 pp. [In English].

- Stack Overflow (2021) Why is voting important? https://stackoverflow.com/help/why-vote. Accessed on: 2021-5-08.
- unkulunkulu (2013) FInding All Shortest Paths Between Two Vertices. Stack Overflow. URL: https://stackoverflow.com/a/16498391/7422352
- Vaira G, Kurasova O (2011) Parallel Bidirectional Dijkstra's Shortest Path Algorithm. In: Barzdins J, Kirikova M (Eds) Proceedings of the 2011 conference on Databases and Information Systems VI: Selected Papers from the Ninth International Baltic Conference, DB&IS 2010. Ninth International Baltic Conference. IOS Press, Van Diemenstraat 94 1013 CN, Amsterdam, 1Netherlands, 14 pp. [In English]. [ISBN *9781607506874*]. https://doi.org/10.5555/1940590.1940629
- Willcock J (2011) Number of paths between two nodes in a DAG. Stack Overflow. URL: https://stackoverflow.com/a/5164820/7422352
- Ziyao Wei (2013) FInding All Shortest Paths Between Two Vertices. Stack Overflow . URL: https://stackoverflow.com/a/16498102

Table 1.
Abbreviations used in the article

Abbreviation	Definition
G	The given directed acyclic graph.
N	The number of nodes.
S	The given source node.
D	The given destination node.
P	The total number of paths between ***S*** and ***D***
E	The edges in the graph
M	The number of edges in the graph
Dist	A two-dimensional array used to store all pair shortest paths computed using dynamic programming. ***Dist[U][V]*** holds the value of the shortest possible distance between the nodes **U** and **V**, where **U**, **V** $\in$ ***G***.
Dist{U}	A one-dimensional array used to store the single source shortest path distance. The value at the index represented by the $\mathbf{i}^{\mathbf{th}}$ node, i.e. ***Dist{U}[i]***, holds the shortest distance from the source **U** to node **i.**