Find Your Circle

Information Push Study through The Kosaraju Algorithm

Introduction

Background of The Topic:

Information being pushed to cell phones is a very common daily activity. Sometimes the pushing information is very useful and helpful, but sometimes it is just a noise that is like a garbage message. This study will go over the general information pushing knowledge, and further analysis on The Kosaraju Algorithm application at information pushing, at last, give the study conclusion and next step researching approach directions.

Context View from Team:

Yiran(Eran): In society, people connect directly or indirectly with others. No matter who you are, you are in a social circle. But twenty years ago, many people did not record the contact information of others due to technological limitations. There's no WhatsApp, Telegram, Teams, Facebook, or even a smartphone. For example, when I was in primary school, I gave my classmates my home phone number to contact me. But many years later, my home phone number has been changed, which means that my primary school classmates can't contact me. After a long time, we didn't know each other's news, and this friendship gradually faded. But I really hope to find this lost friendship. Therefore, I want to design a recommendation system that can recommend based on people's social circles. It can not only help you find lost classmates but also help you find like-minded people. This allows everyone to be in a different social circle, not only can make new friends but can even promote the integration or interaction of social circles.

Chen: This topic reminds me of the movie *The Social Network* which focuses on the story of how the well-known Facebook was established. It was until then that people realized the power of connection. The practical application of this kind of algorithm is much wider than I thought. In addition to social networks, there are fields like e-commerce, online education, and job-hunting that now all have different kinds of recommendation functions to better serve their users. In addition, it also plays a role in the field of language learning and automatic translation can be of great help. So it comes to my mind is there any other field where this method can be implemented to find and build connections to aid us in solving various problems in different areas. That's why I am so interested in this topic and want to explore it more.

Yuchen: The first thing that came to my mind when I saw this topic was the boost of the advertising field in recent years. Before the creation of the modern recommendation system, the topics of ads were posted randomly. This kind of posting approach was targetless and inefficient. Few view numbers can be converted to the company revenue. However, if we can categorize similar ads and consider them as a whole unit, we can push the ads in this unit to the user who watches one of the ads in this unit. By doing this, we can make recommendations based on the user's preferences and convert

the views to the actual income with larger possibilities. And this kind of ideology can be extended to job hunting websites, online education applications, and social networking apps.

Yong: About this topic, the first thing that came to my mind is that I am inspired by a TV show discussion about reforming the structure of the world. No matter if you feel it or not, there are many new things that are changing the world structure, eg. the digital currency, AI and NLP, plan to immigrate to Mars and people are chosen to prepare for it… All these new things may jump out a new giant company, or a new remarkable concept, or a new world. But within all of them, information technology is the core of the development engine. It brings people tons of data and resources and makes connections quickly and effectively. Every day there is a lot of information or advertisements which are pushed to my cell phone, this is also influencing people's mental activities. But the majority of them are based on my browsing history, which I find interesting. I am thinking if there is a new way or algorithm that can change these kinds of activities to rely on one's self-interest but can introduce more diversity and make the world more inclusive when the world is entering the next and new era.

Problem Going to Solve:

Without our active search, this "Find Your Circle" system can automatically help you to find the people who have some degree of strong connection with you. The strong connection can not only be our hobby circle, our social status circle, our career circle, but your life trace circle, which means we may find the people who lose contact with us.

By using the Kosaraju algorithm, the system flags the strongly connected components with the target user. And then it excludes the people in the target user's component but recommends people in other components that have one degree of connection with the target user. Thus, the target user will be recommended by the system with people he needs to add.

Scope of The Project:

Regarding this topic of strong and weak connections, our first goal is to introduce and help others to understand the basic concepts of the Kosaraju algorithm. In this part, we will cover the history, inventor, and other background information of the algorithms.

Then we will demonstrate examples to implement them to show how the algorithms work step by step. This will give us a better understanding of the function of how it works and complexity analysis. Besides that, we will code this algorithm so we can do a live code demonstration of their performance and actual implementation.

The next part is to run an analysis on Kosaraju to figure out the advantages and disadvantages.

Finally, we will cover the real-world applications of this algorithm to explain the significance of the connection topics and how it impacts our real life.

Analysis

Information Push Technical Introduction:

In mobile and server applications, mobile technology can transmit data to mobile without requiring it to send emails. For comparison, the World Wide Web is based on Pull Technology, so consumer browsers must send web page requests before they can send out the information they need.

Dissemination belongs to the scope of application of push technology, because no matter whether someone receives it or not, the information is still correct. Information Push services usually express preferences in advance. This is the so-called publish or subscribe model. Various information channels that a client may subscribe to. Whenever new content is available in one of these channels, the server will push the information to the user.

Synchronous meetings and instant messaging are typical examples of promoting services. Mail and chat, sometimes files are pushed to users as long as they are received by the information service. Peer-to-peer programs where peers are dispersed, and centralized programs allow pushing files, which means that the sender initiates the data transmission, not the recipient.

Email is also a push system, based on the SMTP protocol[1], it is a push protocol. However, the last step, from the mail server to the desktop computer, usually used POP3 or IMAP-like pull protocol. This step of modern email clients seems to pass through a mail server that has repeatedly voted in an instant, often checking for new emails. The IMAP protocol includes the IDLE command, which allows the server to tell the client when new mail arrives. The original BlackBerry was the first popular example of push technology in wireless email.

Another popular Information push technology is the pointcast company network, which gained popularity in the 1990s. It transmits news and stock market data. Netscape and Microsoft's own software integrated to the height of its browser wars, but then gradually disappeared and replaced with One Pull Technology in the 2000s[2].

Web applications for other purposes include promoting market data release, online chat, and messaging systems, auctions, online gambling and games, sports results, monitoring hosts, and sensor network monitoring.

HTTP server pushes also known as HTTP streaming, is a mechanism for sending data from a web server to a web browser. HTTP server push can be achieved through several mechanisms. Generally speaking, after the webserver responds, it terminates the data connection to the client. Push means that the server connection of the website is always open so that if an event is received, the response can be sent to one or more clients immediately. Or put the data in the queue until the client's next request comes, and the response is received by the client. Most web servers provide this functionality via CGI, for example, non-parse headers in Apache scripts. The other mechanism is replaced with a special MIME type called multiple x-mixing, which was developed by Netscape in 1995[3].

Web browsers interpret this as changing whenever the server is like pushing a new version of a file to the client to feel this. It is still supported by Firefox, Opera, and Safari today, but traditionally ignored by Microsoft. It can be applied to HTML files as well as a camera application for streaming images. 1.0 The WHATWG proposal for web applications includes a mechanism to push content to the client. On September 1, 2006, the Opera web browser performed this new experimental technology function

called server-sent events. People are now standardizing as part of HTML5. Another relevant part of HTML5 is the WebSockets API, which allows web servers and clients to communicate through a full-duplex TCP connection[4].

Java pushlet was originally a pushlet to develop Java web applications, although the same technology can be employed in other web frameworks as well as the technology. In this technique, the server needs the advantage of a persistent HTTP connection response, leaving "open" forever, that is, it will never terminate the response, which will effectively fool the "loading" mode to continue the browser's initial page load, usually it's complete. Then, the JavaScript fragments sent by the server regularly update the content of the webpage, so as to realize the driving ability. By using this technology, there is no need for Java applets or other plug-ins on the client to maintain an open connection to the server. The client will automatically notify new events pushed by the server. A serious disadvantage of this method, however, is the lack of control over the server timeout in the browser. A page refresh is always necessary if it occurs at the end of the timeout on the browser.

Long polling is a variation of traditional polling technology and allows an information push emulation from the server to the client. With Long polling, the client requests information in a similar way to the normal voting server. However, if the server does not send everything, instead of any response to the information provided to the client, the server saves the request and waits for some information to be available. Once the information becomes available (or after an appropriate timeout), a complete response is sent to the client. Then, the client usually re-requests information from the server immediately, so the server will almost always have an available waiting request that it can use in response to the event data. In the context of web/AJAX, Long polling is also called Comet programming. Long polling itself is not a push technology, but it can be promoted according to the actual situation.

Flash XMLSocket relays, this technology is used by chat applications such as Cbox, which makes the XMLSocket object use a single-pixel Adobe Flash movie[5]. Under JavaScript control, the client establishes a one-way TCP connection on the server. The relay server does not read this socket but immediately sends a unique identifier to the client. Next, the client sends an HTTP request to the Web server, including its identifier. The web application can then be pushed to the server of the relay, which relays the client's message to their local interface in the flash socket. The advantage of this method is that it appreciates natural read and write asymmetry, which is typical for many web applications, including chat, so it provides high efficiency. Since it does not accept data from the outgoing socket, the relay server does not need to poll all outgoing TCP connections, which can accommodate thousands of concurrent connections to open tens of thousands. In this model, the size limit is the server operating system at the bottom of the TCP protocol stack.

The term Comet has been used to describe Web application push technology applied to Ajax. This is a combination of web technologies, such as HTTP server push and Long polling as a general term.XMPP is generally used to push applications and, in particular, an extension of PubSub. Apple uses its mobile I push to support this technology. Bosh is a long-term HTTP technology used in XMPP, but it can be used on the web. The specification stipulates: This specification defines a transmission protocol that mimics a long-lived, two-way TCP between two entities, such as client and The semantics of the server, connection are effectively used without the need for frequent polling or the use of multiple blocks to synchronously respond to the HTTP request and response pairs.

Information Push Common Algorithm:

Edge Rank Algorithm is one of the most popular algorithms[6]. EdgeRank is the news feed push algorithm developed and used by Facebook, it decides which stories appear in each user's newsfeed. The algorithm hides boring stories, so if your story doesn't score well, no one will see it. Every action someone's friends take is a potential newsfeed story. Facebook calls these actions "Edges." That means whenever a friend posts a status update, comments on another status update, tags a photo, joins a fan page, or RSVP's to an event it generates an "Edge," and a story about that Edge might show up in the user's personal newsfeed.

Facebook calls this algorithm "EdgeRank" because it ranks the edges. Then they filter each user's newsfeed to only show the top-ranked stories for that particular user. How does EdgeRank work? Rank = Affinity x Weight x Decay. Each Edge is made up of the sum of three key factors that form it. These are Affinity, Weight, and Decay. The higher each of these factors is, the higher the EdgeRank and the more people will see your content. Looking at each factor in detail they are defined as:

Affinity – this is a measure of how ‘close’ the viewing user is to the Edge creator. If a user makes a number of interactions (likes, comments, etc) with the Edges of a particular page then their affinity will be greater. This does seem to resonate in my own news feed, in that I tend to be seeing updates from pages that I have recently interacted with. Since I first created a Facebook account on 10 May 2007, I have apparently liked 197 pages, yet I know for sure I only ever see updates from a handful that I generally interact with and have interacted with recently.

Weight – each type of Edge (eg, photo, status update, question) is given a weighting. Publishing content of a heavier weight may increase the EdgeRank, which we’ll explore further below.

As well as content types, user actions can also carry weight – the general suggestion is that a comment has more weight, as the action is more involved, it requires more effort. Another consideration related to this is ‘accumulation’. When a post gains a number of comments or likes, this can give it more weight, creating a positive cycle.

Decay – the factor based on how long ago the edge was created. Broadly speaking, the older the Edge the less value it has, therefore the less impact it has on EdgeRank. Facebook’s news feeds work in a way that can very quickly push down content, especially during peak times of posting, like in the evening.

EdgeRank’s effect is that EdgeRank works by analyzing all of the Edges that the user is connected to before ranking each of the Edges according to the importance of these to the user.

As a rule of thumb, the item with the highest EdgeRank will be placed at the top of the feed, although this is not always the case as Facebook seems to display some randomly as well, just to make things even harder to predict. This ‘randomness’ is something that needs to be carefully considered when looking at EdgeRank. Sometimes Facebook does throw a ‘curve-ball’, by displaying some content that you don’t expect to see in your news feed.

EdgeRank and its successors have a broad impact on what users actually see out of what they ostensibly follow: for instance, the selection can produce a filter bubble (if users are exposed to updates that confirm their opinions, etc.) or alter people's moods (if users are shown a disproportionate amount of positive or negative updates).

Another commonly used is the PageRank Algorithm. PageRank (PR) is an algorithm used by Google Search to rank web pages in their search engine results. It is named after both the term "web page" and co-founder Larry Page.

PageRank is a way of measuring the importance of website pages. It works by counting the number and quality of links to a page to determine a rough estimate of how important the website is. The underlying assumption is that more important websites are likely to receive more links from other websites.

Currently, PageRank is not the only algorithm used by Google to order search results, but it is the first algorithm that was used by the company, and it is the best known.

The Kosaraju Algorithm:

Kosaraju's Algorithm: Kosaraju's Algorithm is also a Depth First Search based algorithm that is used to find the strongly connected component in a directed graph in linear time complexity. The basic concept of this algorithm is that if we can reach vertex v from vertex u, then we should be able to reach vertex u from vertex v. If this is the case, we can say and conclude that vertices u and v are strongly connected and that they are in a strongly connected subgraph[7].

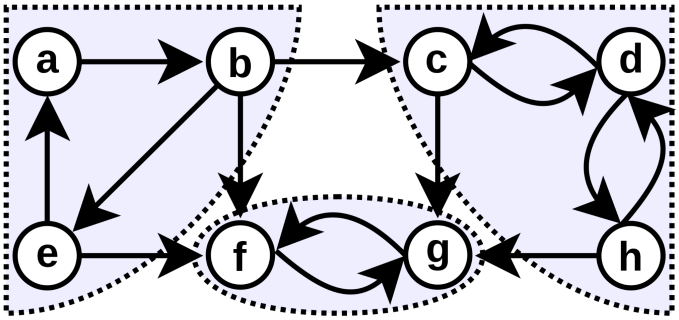

figure1: Strongly Connected Component
(https://www.geeksforgeeks.org/comparision-between-tarjans-and-kosarajus-algorithm/)

- Performs a DFS traversal over the given graph, tracking the completion time of each node. This process can be performed by using a stack.
- When the procedure of running the DFS traversal over the graph finishes, put the source vertex into the stack. In this way, the node with the highest finishing time will be at the top of the stack.
- Reverse the original graph by using an Adjacency List.
- Then another DFS traversal of the inverse graph is performed using the source vertices as the vertices at the top of the stack. When the DFS running on the inverse graph completes, all visited nodes will form a strongly connected component.
- If more nodes are left or remain unvisited, this means that there are multiple strongly connected components on the graph.

- Popping vertices from the top of the stack until a valid unvisited node is found. This will have the highest completion time of all currently unvisited nodes.

Strongly Connected Components:

If we can reach every vertex of a component from every other vertex in that component then it is called a Strongly Connected Component (SCC). A single node is always an SCC[8].

Here is the formal definition: A strongly connected component of a directed graph G is a maximal set of vertices $C \subseteq V$ such that for every pair of vertices u and v, there is a directed path from u to v and a directed path from v to u.

In our project, we will mainly discuss strongly connected components in a **directed graph**. While in an undirected graph, SCC is too easy to identify so we will only do a brief introduction to undirected graphs.

SCC in Undirected Graphs:

An undirected graph that is not connected decomposes into several connected components. Finding the connected components is easily solved using DFS. Each restart finds a new component - done!

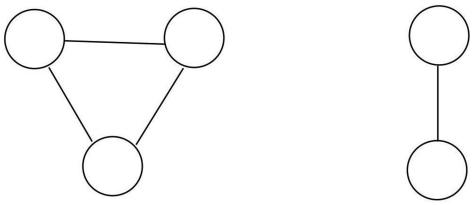

figure2: SCC in Undirected Graphs

SCC in Directed Graphs:

In a directed graph G=(V, E), two nodes u and v are strongly connected if and only if there is a path from u to v and a path from v to u.

The strongly connected relation is an equivalence relation. Its equivalence classes are the strongly connected components. Every node itself is one strongly connected component since the equivalence classes partition the set of nodes.

Let's use the image to demonstrate our ideas. Let G be a directed graph. Then Gscc is a directed acyclic graph(DAG). Indeed, the components in a cycle would have been merged into a single equivalence class. Gscc is a directed acyclic graph, and each node is a strongly connected component of G:

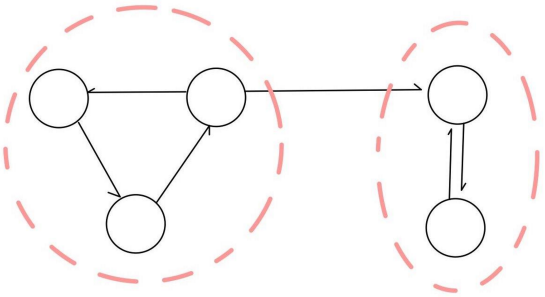

figure3: SCC in Directed Graphs

Source node: In a directed acyclic graph, a node of in-degree 0 is called a source node. **Sink node**: node of out-degree 0 is called a sink node. Each directed acyclic graph has at least one source node and at least one sink node.

Properties of SCC:

Property 1: If the depth-first search of a graph is started at node u, then it will get stuck and restarted precisely when all nodes that are reachable from u are visited. In particular, if we start depth-first search at a node v in G that is in a component C that happens to be a sink in Gscc, then it will get stuck precisely after visiting all the nodes of C. Thus, we have a way of enumerating a strongly connected component given that it is a sink component[9].

Property 2: The node v in G with the highest final [v] timestamp in depth-first search belongs to a start component in Gscc.

Property 3: Let C and D be strongly connected components of a graph. Suppose that there is an edge from a node in C to a node in D. Then the vertex in C that is visited first by depth-first search has a larger final [v] than any vertex in D.

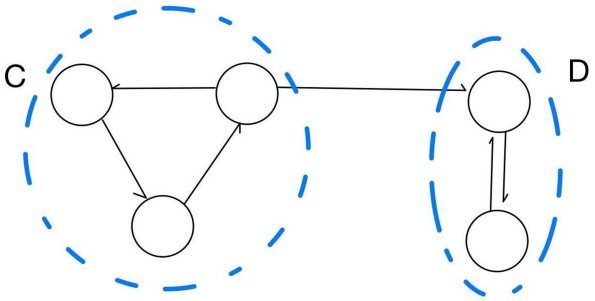

figure4: SCC in Directed Graphs

Why does Kosaraju's Algorithm work:

The transpose graph property is why Kosaraju's algorithm works successfully to search SCCs. SCCs in Graph G and reverse graph G' will always be the same. If we consider each SCC a separate node N, G and G' will be a DAG. As a property of DFS, the largest finishing time in DAG will always be the sync Vertex of Graph (vertex from where there is no outgoing arc) which is N in this case. So if you run DFS from this sync vertex N, it will only iterate through that particular SCC (which forms N).

Below you will see that if we start DFS on the original graph from any node in SCC1 we will be able to reach all the nodes in all the three components, as all are strongly connected components and there is an outgoing edge from the first node to SCCs.

But, if we try doing DFS on the transposed graph from any node in SCC1 we will only be able to reach all nodes in the SCC1 component as there is no outgoing edge from one SCC to another. So, to reach the other SCC we have to make a manual jump. When we do this we can deduce that the last component was a strongly connected component. Now we start our counter and do the above steps again to get the remaining SCCs. Hence, in this iterative way, we will uncover all the SCCs[10].

Fig: SCC of Graph G

Fig: SCC of Graph G transpose

figure5: SCC of Graph G and its reverse graph G'

Kosaraju's Algorithm Demonstration:

Let us run Kosaraju's algorithm on the graph below.

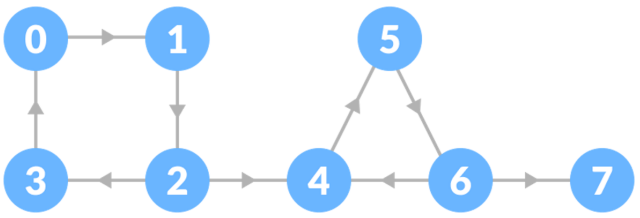

figure6: Initial graph

The strongly connected components of the above graph are:

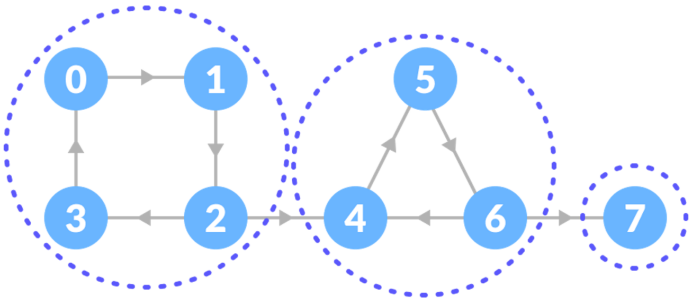

figure7: Strongly connected components

We can observe that in the first strongly connected component, every vertex can reach the other vertex through the directed path.

Now we can run Kosaraju's Algorithm.

This algorithm is implemented based on DFS twice and involves three steps.

Step 1: Perform a depth-first search on the whole graph.

Let us start from vertex-0, visit all of its child vertices, and mark the visited vertices as done. If a vertex leads to an already visited vertex, then push this vertex to the stack.

For example: Starting from vertex-0, go to vertex-1, vertex-2, and then to vertex-3. Vertex-3 leads to already visited vertex-0, so push the source vertex (ie. vertex-3) into the stack.

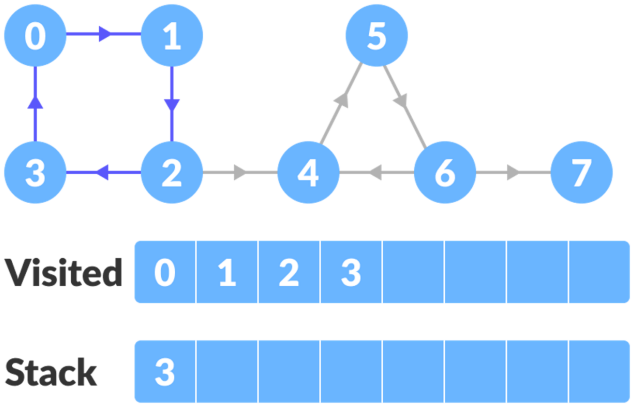

figure8: DFS on the graph-a

Go to the previous vertex (vertex-2) and visit its child vertices i.e. vertex-4, vertex-5, vertex-6, and vertex-7 sequentially. Since there is nowhere to go from vertex-7, push it into the stack.

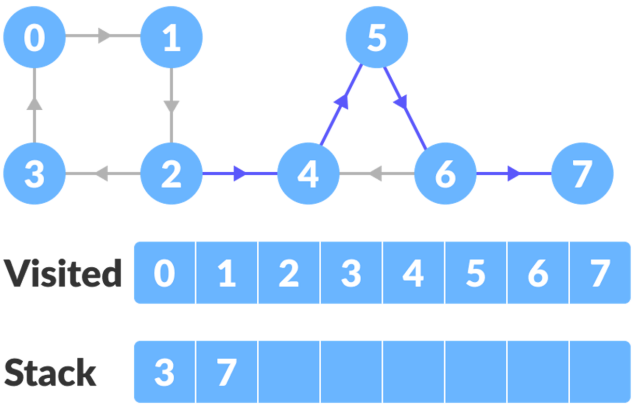

figure9: DFS on the graph-b

Go to the previous vertex (vertex-6) and visit its child vertices. But, all of its child vertices are visited, so push it into the stack.

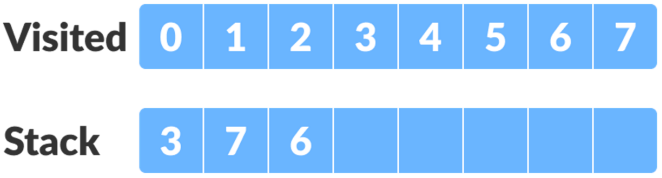

figure10: stacking

Similarly, a final stack is created.

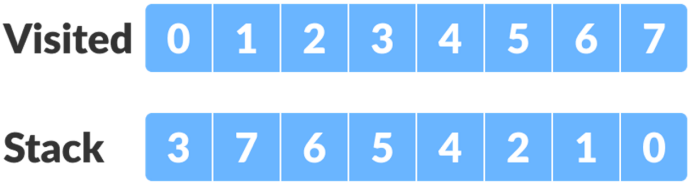

figure11: final stack

Step 2: Reverse the original graph.

Given the graph G=(V,E) consider its reversed graph GR=(V,ER) with ER = { (u,v) | (v,u) in E }, so all edges are reversed. Then GR has the same strongly connected components as G. If we apply depth-first search to GR, then the node v with the largest finishing time belongs to a component that is a sink in Gscc.

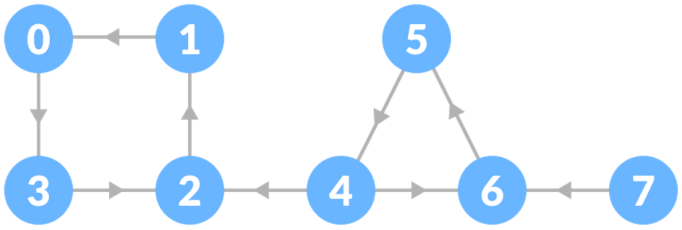

figure12: DFS on reversed graph

Step 3: Perform depth-first search on the reversed graph.

Start from the top vertex of the stack. Traverse through all of its child vertices. Once the already visited vertex is reached, one strongly connected component is formed.

For example: Pop vertex-0 from the stack. Starting from vertex-0, traverse through its child vertices (vertex-0, vertex-1, vertex-2, vertex-3 in sequence) and mark them as visited. The child of vertex-3 is already visited, so these visited vertices form one strongly connected component.

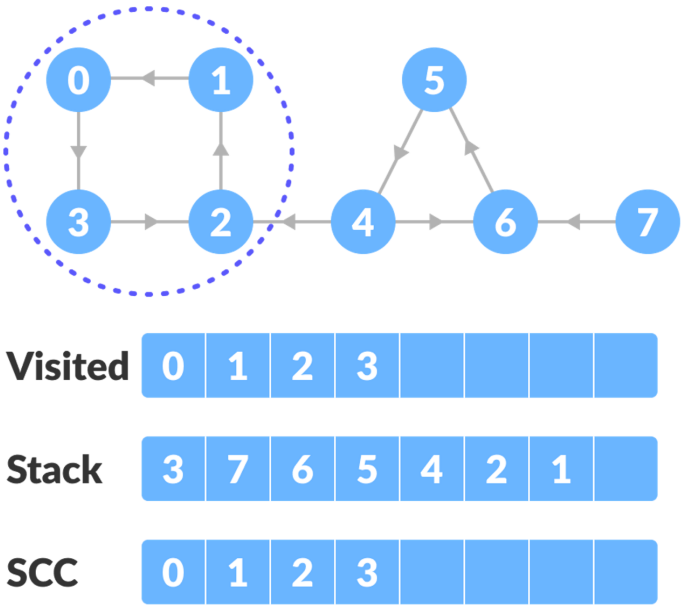

figure13: Start from the top and traverse through all the vertices

Go to the stack and pop the top vertex if it is already visited. Otherwise, choose the top vertex from the stack and traverse through its child vertices as presented above.

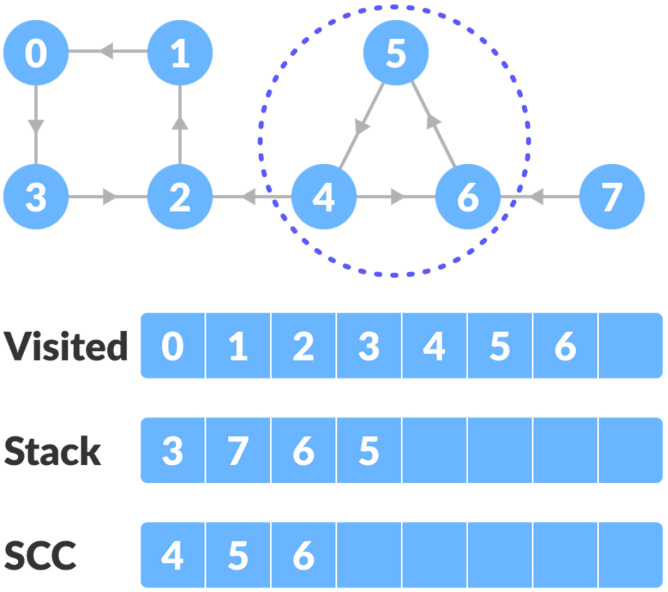

figure14: Pop the top vertex if already visited

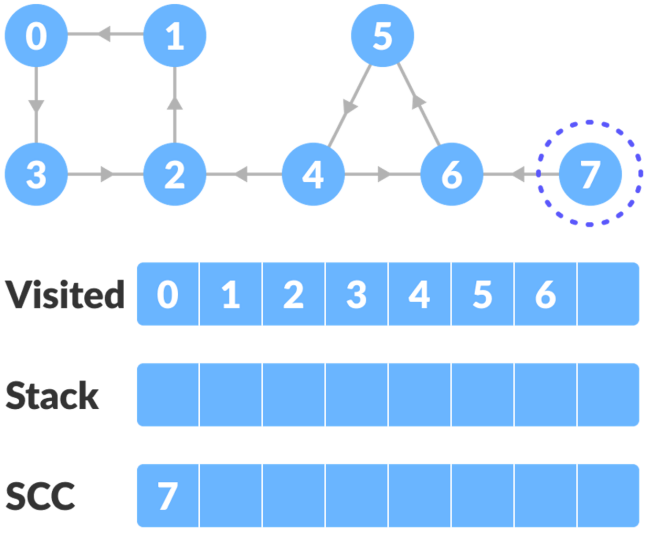

figure15: Last SCCs found

Thus, the strongly connected components are:

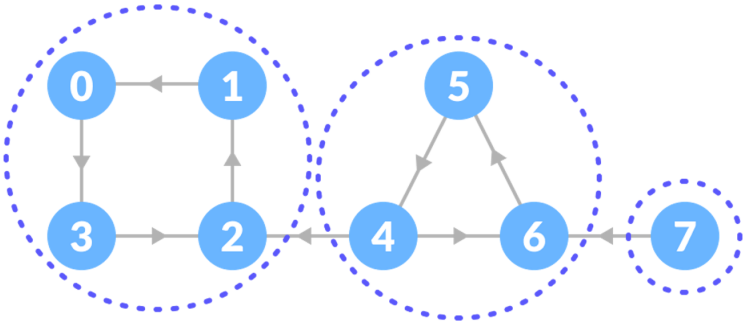

figure16: All strongly connected components

Recommendation System:

In our code we make extensions to Kosaraju Algorithm, we use Floyd-Warshall Algorithm and we design the degree Recommendation code also, which can directly print out the recommendation result for a well user-friendly experience.

Use the below graph as an input example:

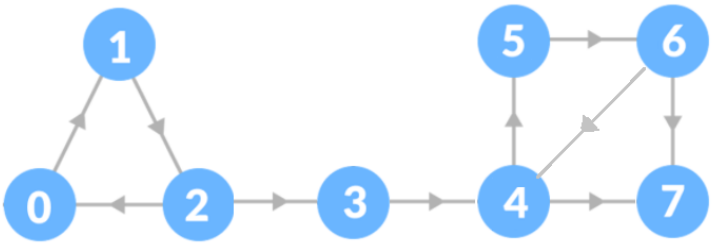

figure17: demo code input graph example

Our code structure is including 4 parts:

Based on the input-directed graph, Kosaraju Algorithm gives the result of all the SCCs as the above-detailed process.

And we use the Floyd-Warshall Algorithm to get all the degrees of each pair of elements.

Then the RecommendService code gives the degree recommendation.

The Main code is the input and output interface. And you can see the print result after running the main code as below screenshot.

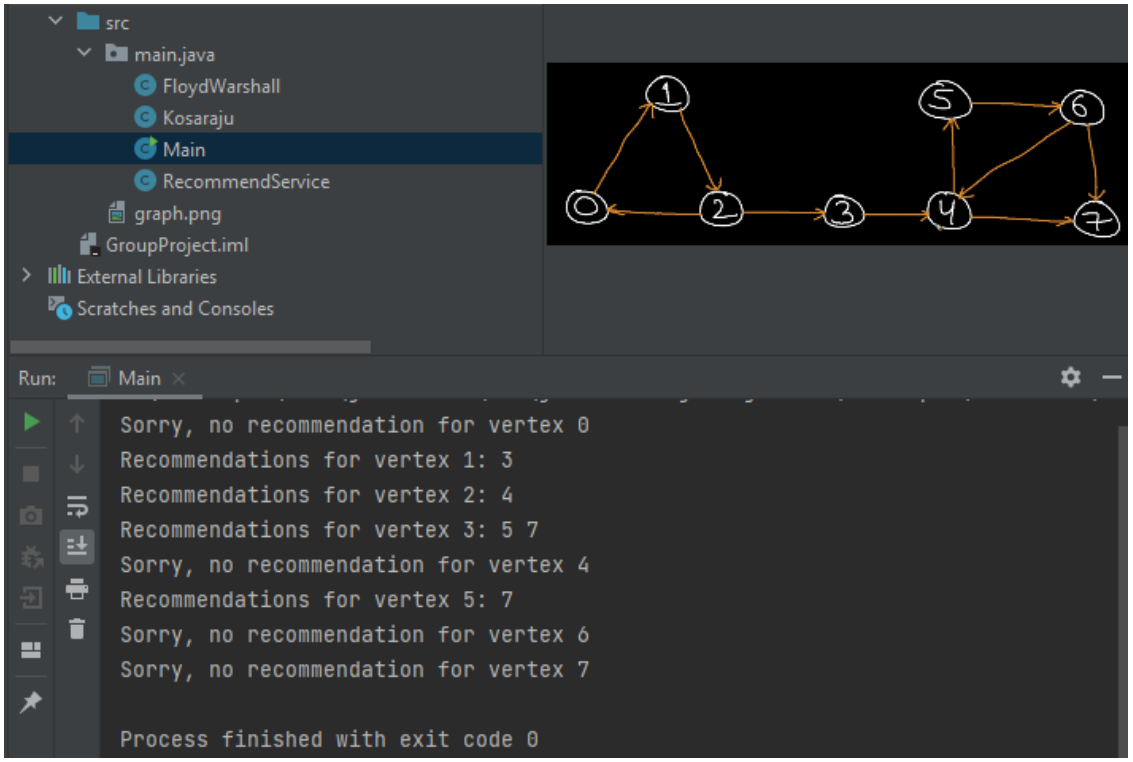

figure18: Java code structure and printing result

The Algorithm Applications Examples:

Vehicle routing applications

A road network can be modeled as a directed graph, with vertices being intersections, and arcs being directed road segments or individual lanes. If the graph isn't Strongly Connected, then vehicles can get trapped in a certain part of the graph, which means they can get in, but not get out. In many of these vehicle routing applications, you want to generate routes for a specific area, for instance, a routing problem within a city. But prior to generating routes, the app would have to extract street data from a map database, which usually covers the entire world. Consequently, it will end up taking a snapshot of the area of interest. The resulting subgraph is not necessarily strongly connected (for instance, you can have a road that enters and leaves the area, but is not connected to any other road inside the area). The next step would be to preprocess the subgraph by enumerating all strongly connected components and throwing away all but the largest component to get the result that users desire.

Global Corporate Control Analysis

The structure of the control network of transnational corporations affects global market competition and financial stability. Transnational corporations form a giant bow-tie structure and a large portion of control flows to a small tightly-knit core of financial institutions. This core can be seen as an economic ‘‘super-entity’’. So it's the small national samples that are studied but there is no great method to assess control globally. Strongly Connected Components can be used to find the set of firms in which every member owns directly and/or indirectly owns shares in every other member.

Hence, by implementing Kosaraju's Algorithm, researchers can get some insights from studying these super entities of the global economy.

Social networks

Kosaraju's Algorithm can be used to find groups of people who are more closely related in a huge set of data, which could also be used to see chunks of a population with common attributes and correlations. Generally speaking, a group of people in the social networks are strongly connected, they must share something in common whether they are in the same company or have the same hobbies. The SCC algorithms can be used to find such groups, and recommend the commonly liked things to the people in the group who have not yet liked these things.

Conclusion

Strongly Connected Components, short for SCC, is a very important concept in graph theory because it can be used for finding the potential connection of currently non-connecting vertices. Its applications are used in various areas, such as social networks, vehicle routing, and financial analysis.

Kosaraju's algorithm follows three steps.

Step 1: Create an empty stack and do a DFS traversal of the input graph.

Step 2: Reverse directions of all edges to get the graph transpose.

Step 3: Start from the top vertex of the stack, do DFS on the reversed graph.

Because Kosaraju's algorithm is heavily dependent on the two DFS, the time complexity is O(V+E), which is also the time complexity of DFS.

In addition to demonstrating how Kosaraju's Algorithm works, we are also programming a recommendation system and combined with Floyd-Warshall Algorithm to simulate how a simple recommendation service works.

Our system “Find Your Circle” can be used to find the node which has some degree of strong connection with other nodes, if we use the nodes to represent a person, they can be considered as a social network recommendation mechanism.

In fact, since we represent the relationships with graphs, the strong connection may reference many common attributes, such as a hobby, career, and so on. If we expand our graph to a more grand level, like all the people in some certain group, we believe our system will help them to find others with potential same attributes and common languages.

This is what interests us most and motivates our group members to dive deep into the power of graphs, and the magic of the world of algorithms.

Next Step:

In our project, we covered the concepts of Strongly Connected Components and how to find them by Kosaraju's algorithm. Whereas, this is not the only way to find SCCs in a directed graph.

Tarjan's Algorithm is another efficient graph algorithm that is used to find the Strongly Connected Component in a directed graph by using only one DFS traversal. Its time complexity is also linear time O(V+E), and its performance is better than Kosaraju's in some aspects.

After investigating the general mechanism of recommendation algorithms, in order to better understand the more practical algorithm used in the tech industry, next step we will go to examine Tarjan's algorithm and understand its more efficient sides.

Reference

[1] Yuan Junhua, Yuan Lin. Research on personalized customization service model based on push technology. CNKI, 2005.

[2] Sun Qingguo, Zhu Wei, Liu Huajun, Zhang Peng. Overview of server push technology in web applications. Computer System Application, 2008.

[3] Wang Zhongmin, Tu Xuyan. The development and application of push technology. Microcomputer Information, 1999.

[4] Jixiang, Gu Xinjian, Dai Feng, Le Chengyi, etc. Product design knowledge push technology based on ontology and rough set. CNKI; WanFang, 2013.

[5] Chen Hang, Zhao Fang. Implementation of WebIM system based on server push technology and XMPP. CNKI; WanFang, 2010.

[6] https://www.wordtracker.com/academy/social/facebook/edgerank-facebook-ranking.

[7] https://www.geeksforgeeks.org/comparision-between-tarjans-and-kosarajus-algorithm/.

[8] https://www.programiz.com/dsa/strongly-connected-components.

[9] https://people.engr.tamu.edu/andreas-klappenecker/csce411-s19.

[10] Micha Sharir. A strong-connectivity algorithm and its applications to data flow analysis. Computers and Mathematics with Applications 7(1):67–72, 1981.