**Strongly Connected Components**

(Kosaraju’s and Tarjan’s Algorithm)

1. **Project Objective**

The objective of our project is to find the Strongly Connected Components (SCCs) in a graph using Kosaraju’s and Tarjan’s algorithm. Our project is inspired by a social media network company called Facebook, where it suggests group of common people based on similar interests. The run time complexity of both these algorithms is O (V + E), where V is vertex and E is edge. Although, both algorithms have same time complexity, but we found that there are some constant factors which makes Tarjan’s algorithm optimal. So, our aim is to compare both these algorithms on different kinds and size of the data and find out which one is better.

1. **Application**

* **Social Media Network**

The Facebook suggests people that might be your friends. Through SCC we can find a group of people who shares common interests.

* **Target Audience for Advertisement**

To find the audience based on similar interests and advertise them the products.

* **Vehicle Routing**

It is also used in vehicle routing problems. For e.g., if UPS truck has to deliver packages at certain places in a town, then through SCC we can find that which packages can be delivered fast by delivering the packages to a location which are connected in a SCC.

* **Model Checking**

One of the applications of finding SCC is in model checking or property checking. For e.g., elevator control software can be model-checked to verify the liveness requirement (such as whenever nth floor button is pressed, the cabin must step at the nth floor) as well as safety requirements (such as cabin never moves with its door open).

1. **Work done previously**

* Previously we have worked on the Kosaraju algorithm. We modified the Kosaraju algorithm so that we can run on real dataset.
* We ran the Facebook dataset and email network dataset on the Kosaraju algorithm to analyze the actual runtime.
* We noted down the execution time of both dataset with respect to different numbers of graph edges. Refer table 1 and table 3.
* For better understanding and comparison, we also created the graphical representation of the result data for which you can refer fig. 1 and fig. 4

1. **Experiments**
2. **Datasets used in the Algorithms**

In our project, we have used two different datasets an Email Network Dataset and a Facebook Dataset to compare the Kosaraju and Tarjan’s algorithm. Both of these datasets are directed graphs. First, we tested the algorithm on a smaller graph (i.e., Email Network Dataset) containing 1005 nodes and 25,571 edges and after that we compare it with the bigger graph (i.e., Facebook Dataset) containing 4039 nodes and 88,234 edges.

* + **Email Network Dataset:**

This dataset is an email data generated from a European research institution. It contains information about the incoming and outgoing email between the members of the institution. There is an edge (u, v) in the network, where u represent an email sent by a person to another person v. It contains about 1005 nodes and 25,571 edges. This dataset can be found here: <http://snap.stanford.edu/data/email-Eu-core.html>

* + **Facebook Dataset:**

This dataset contains information about friend’s lists gathered from Facebook, which was collected from survey taken by the participants using the Facebook app. It contains about 4039 nodes and 88,234 edges. This dataset can be found here: <http://snap.stanford.edu/data/ego-Facebook.html>

1. **Purpose of each experiment**

* In our experiment we have measured the runtime algorithms of both these algorithms on the different datasets. Although asymptotically, both of these algorithms have time complexity of O (V+E), where V are vertices and E are edges. But after comparing the raw runtime of these algorithms we find that one of these is superior as compared to another, on the datasets which we have used. We tested the Kosaraju’s algorithm first on 500 edges and its runtime came out to be 1.45 ms and again running the algorithm for the same number of edges its runtime was different i.e., 1.63 ms. So, we computed the runtime 5 times and calculated the mean of it for the accurate result. Following is the figure of the runtime of Kosaraju’s algorithm on the Email Network dataset we have used.
* **Kosaraju on Email Network Dataset**

|  |  |
| --- | --- |
| **Number of Edges** | **Runtime (ms)** |
| 500 | 1.69 |
| 1000 | 3.45 |
| 5000 | 44.18 |
| 20000 | 183.84 |

Table 1: Runtime of Kosaraju’s Algorithm on different edges

Chart, line chart

Description automatically generated

Fig. 1: Graph of Kosaraju’s Algorithm w.r.t. edges

1. **Experimental Results**

At first, we analyzed the Kosaraju’s algorithm on Email Network Dataset having 500, 1000, 5000, 20000 edges and its average runtime came out to be 1.69, 3.45, 44.18 and 183.84 ms. which is mentioned in Table 1 and Figure 1. And then with the Tarjan’s algorithm. After comparing both Kosaraju and Tarjan on Email Network dataset, we have found that the Tarjan works better and runs the program in less amount of time.

* **Tarjan on Email Network Dataset**

|  |  |
| --- | --- |
| **Number of Edges** | **Runtime (ms)** |
| 500 | 1.03 |
| 1000 | 2.12 |
| 5000 | 27.72 |
| 20000 | 150.47 |

Table 2:Runtime of Tarjan’s Algorithm on different edges

**Chart, line chart

Description automatically generated**

Fig. 2: Graph of Tarjan’s Algorithm w.r.t. edges

**Chart, line chart

Description automatically generated**

Fig. 3: Comparison of Kosaraju and Tarjan on Email Dataset

We have also compared the two algorithms on Facebook dataset. We have tested the algorithm on different number of edges such as 500, 1000, 5000, 20000, 50000.

* **Kosaraju on Facebook Dataset**

|  |  |
| --- | --- |
| **Number of Edges** | **Runtime (ms)** |
| 500 | 1.48 |
| 1000 | 3.41 |
| 5000 | 36.16 |
| 20000 | 191.49 |
| 50000 | 687.18 |

Table 3:Runtime of Kosaraju’s Algorithm on different edges

**Chart, line chart

Description automatically generated**

Fig. 4: Graph of Kosaraju’s Algorithm w.r.t. edges

* **Tarjan on Facebook Dataset**

|  |  |
| --- | --- |
| **Number of Edges** | **Runtime (ms)** |
| 500 | 1.05 |
| 1000 | 2.12 |
| 5000 | 27.88 |
| 20000 | 171.63 |
| 50000 | 606.34 |

Table 4:Runtime of Tarjan’s Algorithm on different edges

**Chart, line chart

Description automatically generated**

Fig. 5: Graph of Tarjan’s Algorithm w.r.t. edges

Chart, line chart

Description automatically generated

Fig. 6: Comparison of Kosaraju and Tarjan on Facebook Dataset

* Here we have compared the Tarjan and Kosaraju on the two different datasets i.e., Email Network Dataset and the Facebook Dataset to see how each algorithm works on the different datasets. We found that on Tarjan’s algorithm works same when using on few numbers of edges but when increasing the size of edges, the Facebook dataset takes more time as compared to the Email data. While the Kosaraju algorithm stays stable on the different datasets.

Chart, line chart

Description automatically generated

Fig. 7: Comparison of Kosaraju and Tarjan on the datasets

1. **Conclusion**

* After comparing both these algorithms on the real time dataset we conclude that, the Tarjan algorithm is about 150% faster than the Kosaraju algorithm.
* So, Tarjan is way faster and more optimal than Kosaraju based on our comparison.
* The Kosaraju’s algorithm does not deviate more as compared to the Kosaraju’s algorithm on the different datasets.

1. **References**

* D. F. Hsu, X. Lan, G. Miller and D. Baird, “A Comparative Study of Algorithm for Computing Strongly Connected Components”, 2017 IEEE 15th Intl. Conf. on Dependable, Autonomic and Secure Computing (DASC), pp.431-437, doi: 10.1109S.
* S. Alshomrani and G. Iqbal, “Analysis of Strongly Connected Components (SCC) using dynamic graph representation”, International Journal of Computer Science Issues, vol. 9, no. 4, pp. 94-100, 2012.
* S. Alshomrani, G. Iqbal, “An Extended Experimental Evaluation of SCC (Gabow’s vs Kosaraju’s) based on Adjacency List”, 2013 Global Journal of Computer Science and Technology Network, Web & Security, vol. 13, no. 11, pp. 0975-4350
* J. Bang-Jensen and G. Z. Gutin, “Digraphs: Theory Algorithms and Applications”, London: Springer-Verlay, pp.795, 2010.
* <https://iq.opengenus.org/kosarajus-algorithm-for-strongly-connected-components/>
* <https://www.thealgorists.com/Algo/GraphTheory/Tarjan/SCC>
* GitHub link for code and ppt reference: <https://github.com/amishgadhia/CSCE-5150---Algo>