Implementation of Graph and Bidirectional Graph Search in Facebook Database

Putu Gery Wahyu Nugraha 13516100[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]gerywahyunugraha@gmail.com

***Abstract*—Facebook is a for-profit social media company. With a billion monthly active user, Facebook needs to find a way to manage all that user into a single database using the best data structure that they can find and to optimize it so it does not impact the user experience. With many data structure to choose from graph can be considered the best data structure to build these database, and by using bidirectional BFS the impact on performance can be minimal.**

***Keywords*—BFS, Bidirectional, Facebook, Graph.**

I. Introduction

Facebook is a social media that able the user to interact with other user (need revision). Facebook currently has two billion active monthly user. With that amount of active user Facebook will need to save at least two billion amount of data, while also preserving some space for other type of data beside user, like Pages and Location. Considering the amount of user and the immense resource it would take to store the data Facebook need a data structure that able to store each profile independently while also able preserve relation between each profile.

The simplest solution to all of this would be implementing all of the users in an array. Where each element of array store all the information about the user, like the location they like and the name of friend that he or she have relationship with. By using this it would be very easy to show all the users that the current user friends with because all of the name is stored in an array inside the user element. This also applies to the pages that he or she likes as it would only takes one iteration O(n) to iterate through the name of the pages that the current user likes. Searching would be quite problematic but the solution is still simple, to search for a friend profile, program only need to search for the friend's name in the user list of friend and iterates through list of user to get the friend profile. And thus implementing all the data of Facebook in an array seems usable in hindsight.

The drawback of using array to implement Facebook data becomes imminent when we take a consideration of how much user Facebook currently had. With two billion user, we will need to create two billion array element, with each of them might store an average of 130 friend's name. This would not be efficient and it would force Facebook to invest more in server infrastructure, a cost that Facebook would like to avoid. With that Facebook need a much more efficient data structure that can scale with the ever-growing number of users that Facebook has.

Graph is another alternative to this. With graph Facebook can store the user profile in a vertex and by connecting two vertex with an edge Facebook can create a relation between the two vertexes. With this data structure adding a new user would be as simple as adding one vertex to the database. This would make the database itself scale much better than the other alternative.

II. Theoretical Framework

A. Graph

Graph is a data structure consisting of object called nodes. These nodes can be connected to each by an edge [1]. When two of the nodes in a graph is connected by an edge, we said that the two nodes neighbor each other. The edges of a graph might have direction or not at all. When a graph have a direction it will be called directed graph, while the one with no direction is called undirected graph. Each edge of a graph can also have weight attached to it. In technical sense, graph is an ordered pair $G = (V, E)$ of vertices (nodes) and edges.

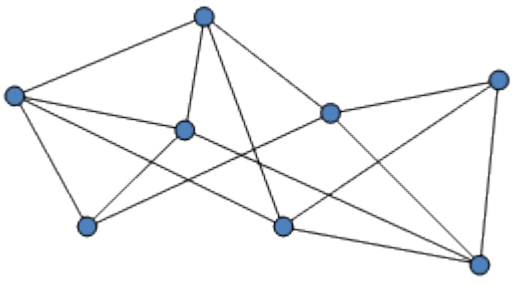


Figure 1 Illustration of a Graph (https://www.uh.edu/engines/graph_theory.gif accessed 3 December 2017)

Some words that associate with graph:

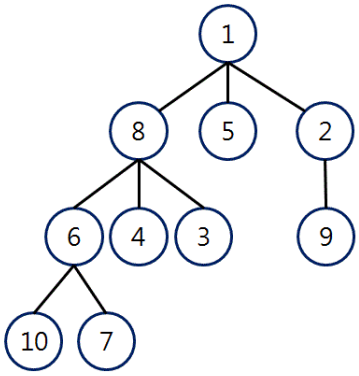
1. Nodes: a point or an object in a graph.
2. Edge: a straight line that connect two nodes together
3. Weight: a number that represent the length of an edge.
4. Neighbor: all nodes that are connected to the first node.
5. Degree: the number of edge that connect to the node.

B. Undirected Graph

Undirected graph is a type of graph that doesn't have any direction when presenting the edges. This type of graph is useful when presented with a problem where each crawler of the graph must be able to crawl from node A to node B and also from node B from node A (although this kind of task is also available in directed graph, it is not as efficient as in undirected graph). As such the application of undirected graph in connecting two user is prevalent as each user must be able to crawl each other.

C. Breadth-first Search (BFS)

Breadth-first search is an algorithm to traverse tree or graph data structure. When applied to a tree it will start from the tree root and traverse to each child that the root had. The algorithm will iterate through all the neighbor nodes first before searching deeper into the next child. The algorithm also works the same on graph, the only difference is the starting nodes for the search.



Order : 1 → 8 → 5 → 2 → 6 → 4 → 3 → 9 → 10 → 7

Figure 2 Order of Search in BFS (https://stackoverflow.com/questions/14208080/breadth-first-search-query-in-mysql accessed December 3, 2017)

The time complexity can be expressed as $O(|V| + |E|)$ since every vertex and every edge will be explored in the worst case. $|V|$ is the number of vertices and $|E|$ is the number of edges in the graph [2].

D. Bidirectional Search

Bidirectional search is a graph search algorithm that finds the shortest path from an initial to the goal vertex. It works the same as other graph search like BFS, the only difference is it doesn't only iterate from the first vertex but also iterates from the goal vertex. The reason bidirectional search is applied to graph is because most of the time compared to the traditional graph search. Compared to BFS that has complexity of $O(k^d)$ Bidirectional BFS has time complexity of $O(k^{\frac{d}{2}} + k^{\frac{d}{2}})$ which is only $O(k^{\frac{d}{2}})$ [3].

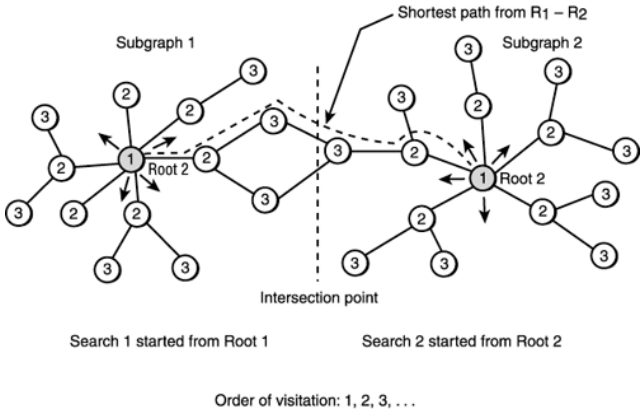


Figure 3 Illustration of Bidirectional BFS Algorithm (http://www.infoarena.ro/blog/meet-in-the-middle accessed December 3, 2017)

E. Facebook

Facebook is an American for-profit corporation and an online social media and social networking service based in Menlo Park, California [4]. The Facebook website was launched on February 4, 2004, by Mark Zuckerberg, along with fellow Harvard College students and roommates, Eduardo Saverin, Andrew McCollum, Dustin Moskovitz, and Chris Hughes. Facebook has grown faster every year, this is shown in their quarterly data. As of the third quarter of 2017, Facebook had 2.07 billion monthly active users. In the third quarter of 2012, the number of active Facebook users had surpassed 1 billion, surpassing all of its competitor, making it to be the first social media to reach 1 billion monthly active user (MAU)

Some terminology associated with Facebook:

1. User: all registered entity in Facebook, including but not limited to person, pages, and group.
2. Monthly Active User: the average number of active user each month in Facebook.
3. Active User: a user that at the very least log in to Facebook in the last 30 days.

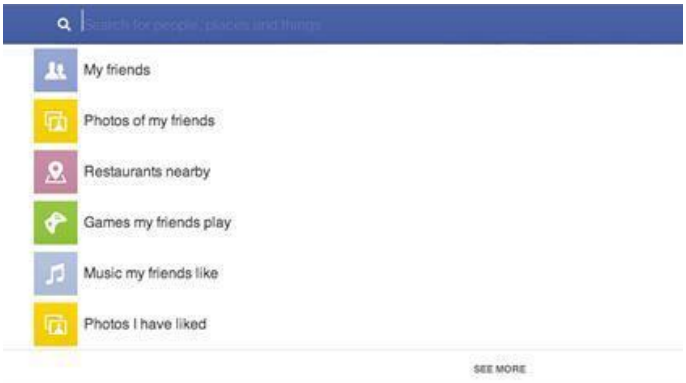


Figure 4 Facebook search use a method of graph search to get the result of the query (https://www.upi.com/Facebook-adds-social-search-function/21161358279306 accessed December 3, 2017)

F. Database

A database is a collection of information that is organized so that it can be easily accessed, managed and updated [5]. Data is organized into rows, columns and tables, and it is indexed to make it easier to find relevant information. Data gets updated, expanded and deleted as new information is added. Databases

process workloads to create and update themselves, querying the data they contain and running applications against it. Computer databases typically contain aggregations of data records or files, such as sales transactions, product catalogs and inventories, and customer profiles.

Some terminology associated with Database:

1. Query: a request of information from the database

III. Implementation of Graph and Bidirectional Graph Search in Facebook Database

A. Graph to Implement Friend and Recommendation

As explained in Introduction, graph is the data structure most suitable to store the data for users. To be more specific each profile—including user, page, and location-- in Facebook's database can be represented as a vertex of a graph. Each vertex contain a detail of each user like their name, bio and stuff. But it does not store the relation that the user have, or which page that he/she likes. The governing of such relation is stored on the edges that connect two vertex. For example, if User A is represented as a vertex of type user and user B is represented as a second vertex of type user. If there is an edge that connect these two vertex Facebook will display user A is in user B friend's list and *vice versa*. This does not apply only to Friends relation but also applies to pages that each user like. If Page A is represented as a vertex of type page and user A is represented as a vertex is of type user. If there is an edge that connect these two vertex Facebook will display user A likes page A.

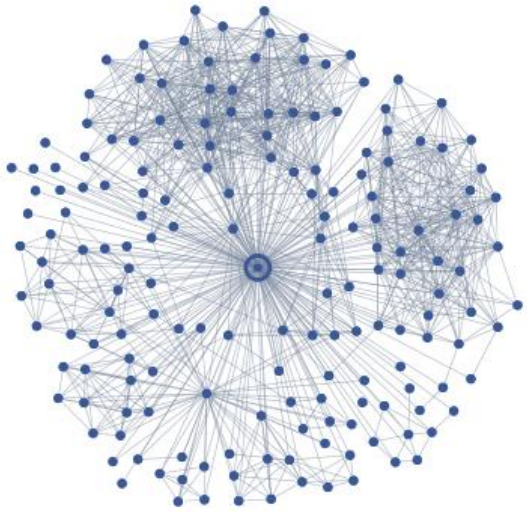


Figure 5 each user connected to other user through an edge (https://research.fb.com/publications/unicorn-a-system-for-searching-the-social-graph accessed December 3, 2017)

With graph data structure, each vertex will only save a pointer to other vertex if a relation is present. This ensure the memory size of each vertex to be relatively small.

Facebook has a feature called Recommendation. This feature help new user to add a new friend that they might know or like a new page that might interest them. Graph data structure in Facebook creates a relation by joining two vertex with edge. Recommendation feature is implemented in similar way but instead of only traversing one edge from the current vertex. It also iterate through the edge of neighboring vertex. By traversing through the edge of neighboring vertex, the crawler will get a list of vertex (user) that is related to the vertex (user) that is related to us. Consider Ray in the image bellow, its vertex is connected to Bryan, Rick, Patrick, Bas, and myself by an edge. If the data is stored in this way, Facebook will treat Ray to be friend with Bryan, Rick, Patrick, Bas and myself. But Facebook can also traverse to the edge of neighboring vertex, thus it also iterates through friend of Bryan, Rick, Patrick, Bas, and myself. This action will return other vertex such as Jamie, Heather, Derek, and Joshua. Because Jamie, Heather, Derek, and Joshua, are not directly connected to Ray—no edges that connect each vertex—it will not display them as friends of Ray. Instead Facebook will display them in a "People You Might Know" section on Ray's Facebook profile.

Algorithm explained above also works to show the list of Pages that user might like. Because each vertex of a page can also connect with other vertex of a page, albeit this time Facebook doesn't treat it as a relation. Consider a page in Facebook called "Marvel" represented as vertex A, and another page called "DC" represented as vertex B. Both vertex can be connected through an edge that represent Genre. As both page consider themselves to be about superhero. Thus when a user A likes "Marvel" Facebook will also shows other vertex that is connected to "Marvel" in this example "DC" and it will show them in the user "Page You Might Like" section.

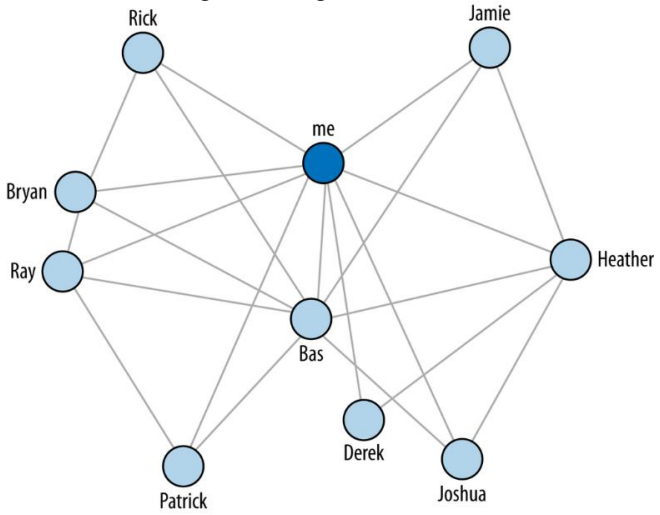


Figure 6 A user can be related to other user through other user (https://www.safaribooksonline.com/library/view/mining-the-social/9781449368180/ch02.html accessed December 3, 2017)

B. Search Algorithm in Graph for Facebook Query

Facebook also has a search function, currently the search bar in Facebook can process natural language and display the result to the user. One such example is if a user query "Friend that lives in Jakarta" Facebook will display a list of user's friend that lives in Jakarta. In the past Facebook only able to search a specific name, searching "Gery Wahyu" will display a list of user with the specific name. Iterating through the graph until we find someone with the given name is simple in graph data structure, as the crawler can just iterate through all the edges of the graph. But Facebook can't just display all the profile in a random order. As that information might be useless to the user that gives the query (the user will be more inclined to give a query of a name of a person that he/she knew). With that Facebook also need to

consider how closely related to person in the query's result to the user who gives the query.

To solve that Facebook use a graph search algorithm called Breadth-first search (BFS). Breadth-first search is a common algorithm to search a graph for a nodes that match the query [6]. BFS solve the problem that Facebook had by searching through all the nodes connected to the current nodes first before doing a deep search (searching through nodes of nodes of nodes). This ensure the first result that Facebook has after applying the query will be the nodes that is closely related to the current node (user applying the search). And the user that give this query will get a result that as closely related as what they had in mind. Although BFS optimize search for the nodes that are closest to the current nodes, sometimes it also needs to iterate deeper through each nodes, this is the drawback for BFS as it is slow to traverse each nodes deeply. But according to Facebook Research Team, any two people in the world that is registered in Facebook are separated by five other people [1]. In technical sense, any nodes in the Facebook database can be connected to other nodes in Facebook database by traversing five nodes. The low amount of nodes needed to traverse makes BFS an efficient algorithm to search the result of a given Facebook query.

C. Optimizing BFS Using Bidirectional Search

Considering the amount of user that Facebook had, Facebook will also need to optimize its breadth-first algorithm to cater for the need of user that has a thousand of users in their friend list. If a user has a thousand other user in their list of friend and the data that the user wants preside in a node two level deep. It would take an amount of 1000^2 iteration to find the goal node. To optimize the speed Facebook needs to find a way to lower the amount of iteration in a search.

The problem stated above can be solved by using bidirectional search algorithm. Using bidirectional search algorithm the crawler will not only crawl from the source nodes but also from the goal nodes. In the best case scenario where the crawler meet in the middle, the first crawler will already iterate through 1000^1 amount of nodes while the second crawler will already crawl through 1000^1 nodes as well. Compared to the regular BFS which have an iteration count of 1 million, bidirectional BFS with iteration count of 2 thousand comes on top in terms of performance.

In the worst case scenario, crawler from the origin node and crawler from the goal node will never met each other, and find the destination node (goal node for crawler from the origin node, and origin node for crawler from the goal node) both BFS and bidirectional BFS in the above case will have an iteration count of 1 million. Making them on par of each other.

The implementation of bidirectional BFS can also be counter-intuitive on Facebook side, as creating two crawler will also increase the amount of processing power needed to process a query from the user. But considering Facebook is a social media company it will try to have the best user experience as possible for the user. A couple of performance hit wouldn't hurt Facebook more if it means that the amount of user will increase. By shortening the processing time, or from the user side the search time of the query. User will tend to stick to Facebook as they consider Facebook to be the faster solution to use social media than the other competitor.

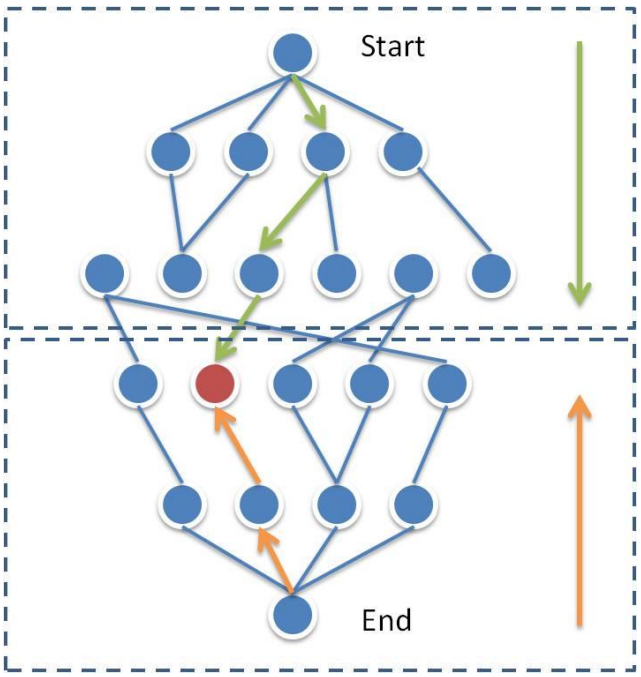


Figure 7 Best case scenario happens when two crawler meet (http://yucoding.blogspot.co.id/2013/08/leetcode-question-127-word-ladder.html accessed December 3, 2017)

D. Analysis of Graph and Bidirectional BFS Usage on Performance

To analyze the impact of graph data structure and bidirectional BFS on the performance of Facebook, we will need to have a basis of the amount of user that is going to be examined. Let say Facebook has 2 million active user for complexity sake, and each user has an average of *n* friends on their friend list.

For the first case we will try to model the usage of simple array as data structure for Facebook's users. In this model, each array element consist of name and a list of name of the user friends. For the second case we will model the usage of graph as data structure for Facebook's users. In this model, each node contain the data for name and = edges that contain address for the other node. Considering in 64 bit operating system the size of the pointer is 64 bit or 8 byte, and Facebook has a limit to the number of character in a name, let's us say 20. It means that each name variable will have a size of 160 bit or 20 byte. Each element of user would take $20 + 20 * numOfFriends$ bytes in array data structure while in graph each user would take $20 + 8 * numOfFriends$ bytes.

Avg. Friends	Graph Size (GB)	Array Size (GB)
50	2.8	4
100	5.6	8
150	8.4	12
250	11.2	16
300	14	20

Figure 8 Size of data in response to change in number of friends

As pointed by table above every time the number of average friends increase the size of graph and the size of array will increase linearly. Although 6 gigabytes might not seem significant it will become much more important if Facebook has a larger database of data.

For the second test we will compare the speed of graph search using a regular BFS compared to bidirectional BFS. When testing this we will treat each one from 2 million users of Facebook have an average of *n* users in their friend list (this is the current condition on Facebook). For regular BFS there are no best case or worst case scenario. But for bidirectional BFS we will treat their complexity as the average of best case scenario and the worst case scenario. Because the best case scenario for bidirectional BFS is gained when the two crawler meet in the middle, in this case the complexity of the algorithm will be

$$O(2k^{\frac{d}{2}}) \tag{1}$$

Worst case scenario happens when the two crawler doesn't meet at all, in this case the complexity of the algorithm will be

$$O(k^d) \tag{2}$$

By taking the average of the value in equation (1) and (2) we will get the average complexity of the algorithm will be

$$O(\frac{k^d + 2k^{\frac{d}{2}}}{2})$$

The complexity of the algorithm for BFS will be the same as the number from equation (2). To calculate the time it takes to complete a task we will use a computer that able to do 1 billion calculation every second. And for this test, we will give query to each algorithm that will traverse the node of the database at least 4 level deep.

By referring to the calculation shown in Figure 8 we can see that the time it takes to process the query increase exponentially with increase of average number of friends. When comparing the process time in bidirectional BFS and regular BFS we can see that bidirectional BFS complete the process twice as fast as the regular BFS. But if we see the data for the best case scenario for bidirectional BFS, the different in time for process to complete compared with regular BFS is exponentially smaller.

IV. Conclusion

With the ever-growing number of people that join Facebook. The company needs to find a way to solve the problem in memory management and also keep user experience smooth. With that Facebook can't implement a simple data structure like array and it needs to resort to a much complex graph data structure that have shown to be much more efficient to handle database with high amount of users. To preserve the user experience as smooth as possible, Facebook need an algorithm that can handle query faster than a simple graph search. By using BFS Facebook able to query the result that match the user desire, but it still fall short in performance management. Because of that Facebook need to resort to bidirectional BFS that has shown to be exponentially faster than regular BFS in the best case scenario.

VI. Acknowledgment

I would like to express my deep gratitude to Facebook developer team, as their contribution to make their source code open help me tremendously in completing this project. I would also give my gratitude for all my friend that help me to choose what topic should I bring for this paper. I wish to thank my parents for their support and encouragement throughout my study. And last but not least I thank God that have given me guidance and tranquility in the process of writing this paper.

I realize there's still a lot of mistake and simplification that I do when drawing conclusion in this paper and I sincerely apologize for all of that. I hope that these paper can be used as a guidance to help student understand how graph and database works, and I fully support all the works that can branch out from my writing.

References

[1] *Trudeau, Richard J. (1993). Introduction to Graph Theory (Corrected, enlarged republication. ed.). New York: Dover Pub. p. 19. ISBN 978-0-486-67870-2. Retrieved December 3, 2017. A graph is an object consisting of two sets called its vertex set and its edge set.*

[2] *Cormen, Thomas H.*; *Leiserson, Charles E.*; *Rivest, Ronald L.*; *Stein, Clifford* (2001) [1990]. "22.2 Breadth-first search". *Introduction to Algorithms* (2nd ed.). MIT Press and McGraw-Hill. pp. 531–539. *ISBN 0-262-03293-7.*

[3] Anonymous (2017). "Design data structures for a very large social network like Facebook or Linkedln". *http://www.geeksforgeeks.org/design-data-structures-for-a-very-large-social-network-like-facebook-or-linkedln*. Geeks for Geeks. Retrieved December 3, 2017.

[4] Carlson, Nicholas (March 5, 2010). "At Last -- The Full Story Of How Facebook Was Founded". *http://www.businessinsider.com/how-facebook-was-founded-2010-3*. Business Insider. Axel Springer SE. Retrieved December 3, 2017.

[5] Lane, Adrian (2006). "database (DB)". *http://searchsqlserver.techtarget.com/definition/database*. SearchSqlServer. Retrieved December 3, 2017.

[6] Mishra, Arpit (January 6, 2016). "Breadth First Search example (BFS) - How GPS navigation works". http://blog.hackerearth.com/breadth-first-

Avg. Friend	Number of Computation (BFS) (bill)	Number of Computation (Bidirectional BFS) (bill)	Number of Computation (Best case Bidirectional BFS) (bill)	Time for Process (BFS) (s)	Time for Process (Bidirectional BFS) (s)	Time for Process (Best case Bidirectional BFS) (s)
50	0.0625	0.031275	0.0000025	0.0625	0.031275	0.0000025
100	0.1	0.05	0.0000100	0.1	0.05	0.0000100
150	0.50625	0.253125	0.0000225	0.50625	0.253125	0.0000225
200	1.6	0.8	0.0000400	1.6	0.8	0.0000400
250	3.90625	1.953125	0.0000625	3.90625	1.953125	0.0000625
300	8.1	4.05	0.0000900	8.1	4.05	0.0000900
350	15	7.5	0.0001225	15	7.5	0.0001225
400	25.6	12.8	0.0001600	25.6	12.8	0.0001600
450	41.0062	20.5031	0.0002025	41.0062	20.5031	0.0002025
500	62.50	31.75	0.0002500	62.5	31.75	0.0002500

Figure 9 Increase of process time in response to average number of friends of each user

search-algorithm-example-working-of-gps-navigation. HackerEarth|Blog. Retrieved December 3, 2017.

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 3 Desember 2017

Putu Gery Wahyu Nugraha 13516100