Getting Closer to Your Dream Girl through *Six Degrees of Separation* with Graph Theory

Jones Napoleon Autumn - 13518086
Informatics Major
School of Electrical Engineering and Informatics
Bandung Institute of Technology, Jl. Ganesha 10 Bandung 40132, Indonesia
jonesnapoleon@students.itb.ac.id

***Abstract* — Disconnection and unfamiliarity across individuals are inevitable, and the feeling of obscurity knowing the impossibility in acquainting one's dream girl is truly painful. However, little did we know that a simple implementation of Graph theory could help one in realizing connections to his dream girl, with the cost of world's population acquaintance list, indeed. This paper shall cover the implementation of Graph theory in discovering connection path from one person to another under seven connections – Six Degrees of Separation as the curtail, and hopefully, help any little boy in getting closer to his dream girl.**

***Keywords* — Connectedness, Dream Girl, Graph Theory, Six Degrees of Separation.**

I. INTRODUCTION

Six Degrees of Separation has been a popular idea that indicates the connectedness of individuals. The idea that all people are six, or fewer, social connections away from each other, is often called as *6 Handshakes rule*. As a result, a chain of "a friend of a friend" statements can be made to connect any two people in a maximum of six steps.

The idea, again, was re-popularized by a Hollywood actor Kevin Bacon with the invention of the game "Six Degrees of Kevin Bacon". The concept is rather simple: link any celebrity to Kevin Bacon in under seven connections, and the jump of connection is being referred to as 'Bacon Number'. But rather than linking any celebrity to Kevin Bacon, we could actually link people from any side of the world to the other. We could even specifically link a guy from one end to a stranger girl on the other.

Having mentioned that, it would be interesting to connect a guy to his dream girl, instead of a stranger, but still under the rule of maximum six degrees to make it even more interesting

As the purpose of this paper is to help solving real-life problem with Graph theory, author would love to up the ante and also help solve personal issue, in this case the disconnectedness of a guy and his dream girl.

A classical approach in getting acquaintance to other people would be through connection from a mutual friend. This paper is specifically written with those way of acquaintance in mind. With the global population exceeding 7.7 billion people, there are numerous graphs with countless interconnection within individuals. With the theory of Six Degree of Separation as the number of connection curtail, Graph theory is ready to help connecting a guy to his dream girl by displaying all the connection path possibilities.

II. FUNDAMENTAL THEOREM

A. Graph

A graph is a pair *G = (V, E)*, where *V* is a set whose elements are called *vertices* (singular: *vertex*), and *E* is a set of two-sets of vertices, whose elements are called *edges*. The vertices *x* and *y* of an edge {*x*, *y*} are called the *endpoints* of the edge and the vertices *x* and *y* being called *adjacent*, whereas the edge is said to be *incident* on *x* and *y* (represented as (*x*, *y*)) [1].

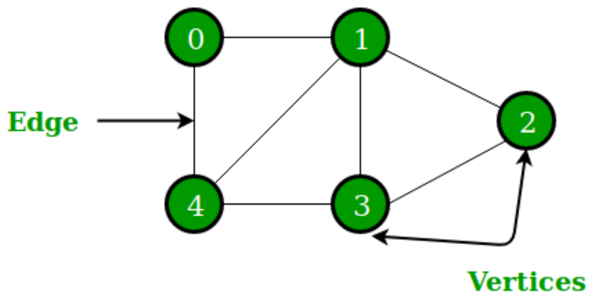


Figure 1. Example of graph. [5]

A vertex may not belong to any edge, called isolated vertex. If every vertices in a graph is an isolated vertex, the graph is called a *null graph*; a complete opposite of this graph is called a *complete graph* where each pair of vertices is joined by an edge, containing all possible edges in a single graph. Besides, one interesting concept of graph is *subgraph*. If there exist a couple of graph pair *G = (V, E)* and *G1 = (V1, E1)*, where *V1* ⊆ *V* and *E1* ⊆ *E*, then the graph *G1* is said to be the subgraph of *G*.

In general, graph can be categorized by several of its properties. Based on the orientation of the edges, graph is divided into directed graph and undirected graph. Directed graph is a graph where all of its edges has direction (displayed with arrow); one of the endpoints acts as the source and the other as the destination. Undirected graph is one where all the edges are bidirectional, thus, should a vertex *u* connects to a vertex *v*, *v* must also connect to *u*. In computer science, directed graph is often utilized to represent conceptual graph, like finite state machines. Figure 1 also represents an unweighted and undirected graph.

Another property of a graph is weight. A weighted graph is one in which every edges has a value assigned, whereas an unweighted graph is one in which every edges has the same value (weight). The application of weighted graph becomes prominent in cases like *Minimum Spanning Tree* – an interesting topic out of this paper scope, while that of the unweighted is usually used in social network where connections only has 'True' or 'False' value.

Graph can be represented in several ways. Two of the most commonly used representations are Adjacency Matrix and Adjacency List.

Adjacency matrix defines the graph by using matrix where the number of rows and columns equal to the number of vertices and the possible value for every matrix element is only 1 (True) and 0 (False). An edge (u, v) exist in the graph if and only if the element $A_{u,v}$ has the value of 1. Otherwise, the value of element $A_{u,v}$ must be 0. In weighted graph, the value of element $A_{u,v}$ can be replaced as the weight value. A complete graph has an adjacency matrix full of 1 except its main diagonal and an undirected graph has an adjacency matrix where its main diagonal acts as a mirror.

Adjacency list defines the graph by a list with effective indices equals to the number of total vertices, with each index represents the corresponding vertex. Then, only if a vertex v is connected to vertex u will u be listed on v's adjacency list. If the graph is weighted, the adjacency list will be populated with tuple (u, w), where w is the weight.

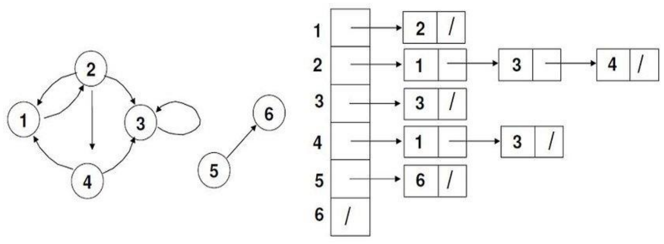


Figure 2. A directed graph represented by adjacency lists. [6]

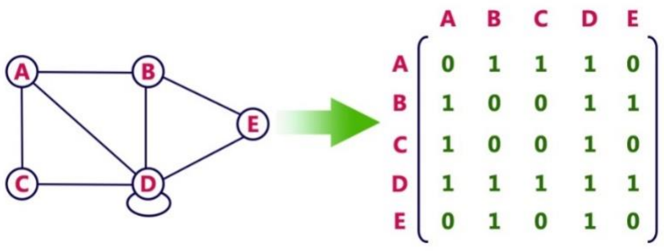


Figure 3. A graph represented by adjacency matrix. [7]

B. Six Degree of Separation theory

Six Degrees of Separation is the idea that all people are six, or fewer, social connections away from each other, often called as *6 Handshakes rule*. As a result, a chain of "a friend of a friend" statements can be made to connect any two people in a maximum of six steps. It was originally coined by Frigyes Karinthy in 1929 and later popularized in an eponymous 1990 play written by John Guare. It is sometimes generalized to the average social distance being logarithmic in the size of the population [2].

The idea has been well supported by other early conceptions as well. Shrinking world; due to technological advances in all fields, including communication and travel, friendship networks has grown larger and has spanned a greater distance. Karinthy himself believed that the modern world was '*shrinking*' due to this ever-increasing connectedness of human beings. He posited that despite great physical distances between the globe's individuals, the growing density of human networks made the actual social distance far smaller [3].

With its popularity, several studies have specifically been conducted to measure the connectedness of global people empirically. Initially developed by Karl Bunyan, a Facebook platform application named "Six Degrees" was established to calculate the degrees of separation between people. After some redevelopment of the application, Facebook's data team released two papers in November 2011 which document that amongst all Facebook users at the time of research (721 million users with 69 billion friendship links), there is an average distance of 4.74. In 2016, the figure fell to 4.57, showing the interconnectedness among social people. Probabilistic algorithms were applied on statistical metadata to verify the accuracy of the measurements, and in return, 99.91% of Facebook users were found interconnected, forming a large connected component [4].

After the release of John Guare's play in 1990, the idea, again, was popularized by a Hollywood actor Kevin Bacon with the invention of the game "Six Degrees of Kevin Bacon". The concept is rather simple: link any celebrity to Kevin Bacon in under seven connections, and the jump of connection is being referred to as 'Bacon Number' [8].

III. APPLICATION OF GRAPH IN SIX DEGREES OF SEPARATION THEORY

Six degrees of separation represents at most six jumps or seven people (including the people to be linked) in connections leap. In graph representation, the people can be represented by vertices, while the connections can be represented by edges. The total of which for the former should not exceed 7, and that for the latter should not exceed 6. Therefore, the first requirement for a single *Six Degrees* connection is a subgraph *G1* (of a global graph *G*) with a maximum of 6 nodes and 7 vertices.

Since social connectedness (relations) could not be measured quantitatively (at least for this paper's purpose) - for instance connection in social media means a truly-connected (*True*) value, otherwise falsely-connected (*False*), the graph used in Six Degrees of Separation theory would be unweighted. In addition, a true connection would be referred to as *True* if and only if both parties know each other well, here it means that the right graph representation would be that of undirected.

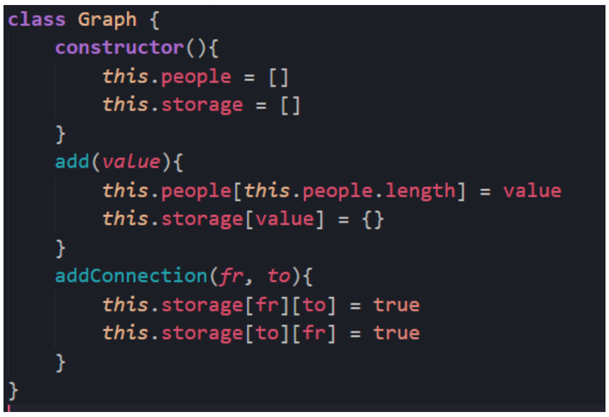
With a vast connection potentially occur in the global graph (human population exceeding billions), the usage of Adjacency Matrix would be inefficient as it requires the square of those number (billions x billions). Thus, Adjacency List is of a better choice for an effective memory usage.

In summary, a better approach of Six Degrees of Separation with graph (at least for this paper's purpose) would harness an unweighted and undirected graph, with the people's relations being represented by Adjacency List, with the subgraph not exceeding six degrees.

IV. IMPLEMENTATION OF GRAPH IN SIX DEGREES OF SEPARATION THEORY

For a better dynamic Graphical User Interface (GUI) of the graph representation, author uses JavaScript language to demonstrate the implementation in Six Degrees of Separation.

First and foremost, create the object of Graph.



```
class Graph {
    constructor(){
        this.people = []
        this.storage = []
    }
    add(value){
        this.people[this.people.length] = value
        this.storage[value] = {}
    }
    addConnection(fr, to){
        this.storage[fr][to] = true
        this.storage[to][fr] = true
    }
}
```

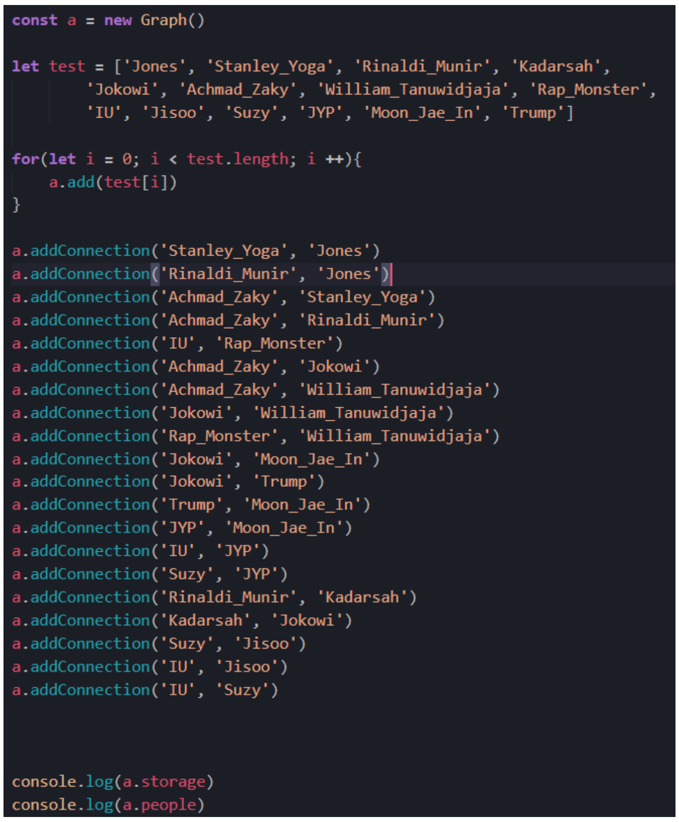
Figure 4. Graph object with its constructor and some prominent methods.

The *storage* variable stores a number of lists of the people's relations as an Adjacency List. The *people* variable stores an array of people involved in the graph system. Upon creating the graph object, these variables are both assigned to an empty list.

The function *add* takes one parameter (*person*) as its input, and add the *person* to the list of people as well as setting its relations list to an empty list.

The function *addConnection* takes a couple arguments (*person1*, *person2*) as its input and set the connectedness value of *person1* and *person2* to *True*.

Then, initialize the graph globally and add several people with its connection, try printing it on the console afterwards.

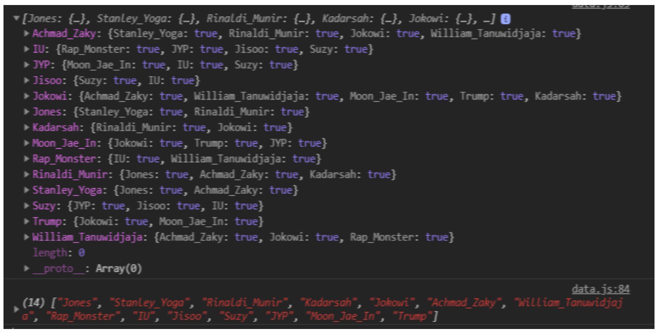


```
const a = new Graph()

let test = ['Jones', 'Stanley_Yoga', 'Rinaldi_Munir', 'Kadarsah',
        'Jokowi', 'Achmad_Zaky', 'William_Tanuwidjaja', 'Rap_Monster',
        'IU', 'Jisoo', 'Suzy', 'JYP', 'Moon_Jae_In', 'Trump']

for(let i = 0; i < test.length; i ++){
    a.add(test[i])
}

a.addConnection('Stanley_Yoga', 'Jones')
a.addConnection('Rinaldi_Munir', 'Jones')
a.addConnection('Achmad_Zaky', 'Stanley_Yoga')
a.addConnection('Achmad_Zaky', 'Rinaldi_Munir')
a.addConnection('IU', 'Rap_Monster')
a.addConnection('Achmad_Zaky', 'Jokowi')
a.addConnection('Achmad_Zaky', 'William_Tanuwidjaja')
a.addConnection('Jokowi', 'William_Tanuwidjaja')
a.addConnection('Rap_Monster', 'William_Tanuwidjaja')
a.addConnection('Jokowi', 'Moon_Jae_In')
a.addConnection('Jokowi', 'Trump')
a.addConnection('Trump', 'Moon_Jae_In')
a.addConnection('JYP', 'Moon_Jae_In')
a.addConnection('IU', 'JYP')
a.addConnection('Suzy', 'JYP')
a.addConnection('Rinaldi_Munir', 'Kadarsah')
a.addConnection('Kadarsah', 'Jokowi')
a.addConnection('Suzy', 'Jisoo')
a.addConnection('IU', 'Jisoo')
a.addConnection('IU', 'Suzy')



console.log(a.storage)
console.log(a.people)
```

Figure 5. Graph object declaration and attributes' assignment.

To simplify computation, author declare only thirteen *people* and set only twenty connections across the graph; despite knowing the case in real-life is not going to be this smooth. Also, note that any names used in this paper is only for academic purpose and do not represent any individual/entities in the real world.

The result of the Adjacency List (*storage* variable) and the list of the people (*people* variable) are shown as follow.



```
▾[Jones: {…}, Stanley_Yoga: {…}, Rinaldi_Munir: {…}, Kadarsah: {…}, Jokowi: {…}, …] i
  ▸Achmad_Zaky: {Stanley_Yoga: true, Rinaldi_Munir: true, Jokowi: true, William_Tanuwidjaja: true}
  ▸IU: {Rap_Monster: true, JYP: true, Jisoo: true, Suzy: true}
  ▸JYP: {Moon_Jae_In: true, IU: true, Suzy: true}
  ▸Jisoo: {Suzy: true, IU: true}
  ▸Jokowi: {Achmad_Zaky: true, William_Tanuwidjaja: true, Moon_Jae_In: true, Trump: true, Kadarsah: true}
  ▸Jones: {Stanley_Yoga: true, Rinaldi_Munir: true}
  ▸Kadarsah: {Rinaldi_Munir: true, Jokowi: true}
  ▸Moon_Jae_In: {Jokowi: true, Trump: true, JYP: true}
  ▸Rap_Monster: {IU: true, William_Tanuwidjaja: true}
  ▸Rinaldi_Munir: {Jones: true, Achmad_Zaky: true, Kadarsah: true}
  ▸Stanley_Yoga: {Jones: true, Achmad_Zaky: true}
  ▸Suzy: {JYP: true, Jisoo: true, IU: true}
  ▸Trump: {Jokowi: true, Moon_Jae_In: true}
  ▸William_Tanuwidjaja: {Achmad_Zaky: true, Jokowi: true, Rap_Monster: true}
   length: 0
  ▸__proto__: Array(0)
                                                                              data.js:84
▸(14) ["Jones", "Stanley_Yoga", "Rinaldi_Munir", "Kadarsah", "Jokowi", "Achmad_Zaky", "William_Tanuwidjaj
a", "Rap_Monster", "IU", "Jisoo", "Suzy", "JYP", "Moon_Jae_In", "Trump"]
```

Figure 6. Adjacency List of the graph (*storage* variable) and its list of people.

For the sake of better User Interface, author implements little code in JavaScript to enable the visualization of the graph, full with each of its people's connections. Below, the result is displayed.

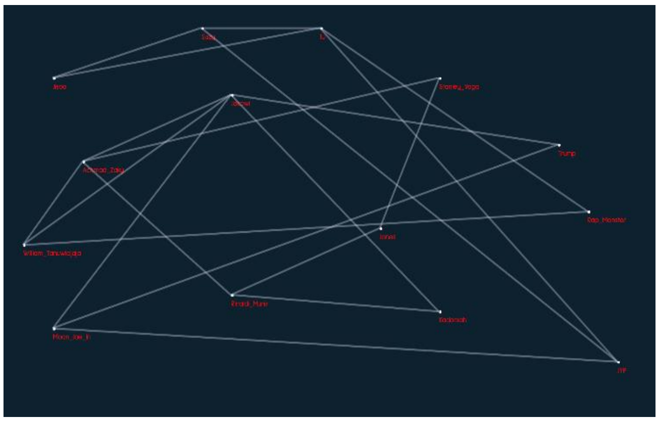


Figure 7. Graphical representation of the interconnection in the graph.

Now, a graph has been visually and completely represented. What's left is the notion of connecting a guy to his dream girl. In this case, say, *IU* is a girl, *Jones* is a guy, and *IU* happened to be *Jones*'s dream girl, assuming *Jones* has had known *IU* from somewhere else, say, TV drama.

But before acquainting *IU*, *Jones* would first need to set up a strategy in order to get her to know about him, and thanks to the power of Graph theory, *Jones* could see a glimpse of hope in truly befriending *IU*.

The graphical-approach strategy would be to learn about *IU*'s social connections, and find *Jones*'s connections way out until reaches *IU*. With convenience and time being put under consideration, *Jones*, of course, would not wait and connect to more than six people in order to only get to know a single person.

Intuitively, we can see several linkages that might allow *Jones* to be a step closer to *IU*. One of which would be *Jones – Rinaldi_Munir – Kadarsah – Jokowi – Moon_Jae_In – JYP – IU* (Six Degree of Separation). Right here, author would love to dynamically display all possibilities of relations to one person to another.

The first step is to prompt user for two *people* (in this case, *Jones* to *IU*) and display all of the interconnection combination from those *people*. Displayed below is the Graphical User Interface of the *player* inputs.

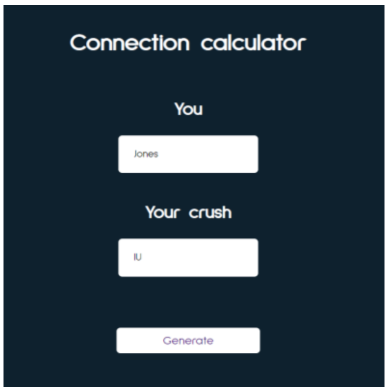
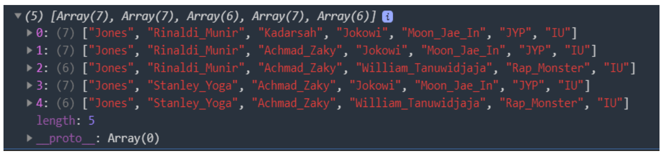


Figure 8. Graphical User Interface of the *player* inputs.

The inputs, then, is sent to the system to generate all under-seven-connection possibilities. Any graph-traversing algorithm, such as *Breadth-first search* or *Depth-first search* would do; these algorithms, however, are not discussed as they are out of this paper's scope. Long story short, we obtain all possibilities under a single list. Displayed below are all the under-seven-connection possibilities from *Jones* to *IU*, and/or vice versa.



```
▼ (5) [Array(7), Array(7), Array(6), Array(7), Array(6)]
  ▶ 0: (7) ["Jones", "Rinaldi_Munir", "Kadarsah", "Jokowi", "Moon_Jae_In", "JYP", "IU"]
  ▶ 1: (7) ["Jones", "Rinaldi_Munir", "Achmad_Zaky", "Jokowi", "Moon_Jae_In", "JYP", "IU"]
  ▶ 2: (6) ["Jones", "Rinaldi_Munir", "Achmad_Zaky", "William_Tanuwidjaja", "Rap_Monster", "IU"]
  ▶ 3: (7) ["Jones", "Stanley_Yoga", "Achmad_Zaky", "Jokowi", "Moon_Jae_In", "JYP", "IU"]
  ▶ 4: (6) ["Jones", "Stanley_Yoga", "Achmad_Zaky", "William_Tanuwidjaja", "Rap_Monster", "IU"]
    length: 5
  ▶ __proto__: Array(0)
```

Figure 9. List of all under-seven-connection possibilities between *Jones* and *IU*.

Figure 9 indicates that there are five possibilities of social connections from *Jones* to *IU*, abiding by the theorem of *Six Degrees of Separation*. Below, all the graphical representation of each connection path are displayed.

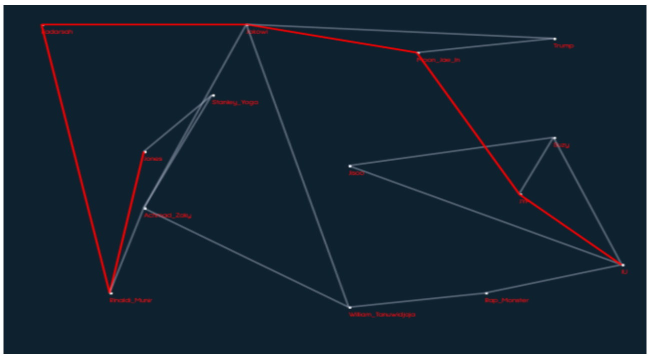


Figure 10. First connection path from *Jones* to *IU*.

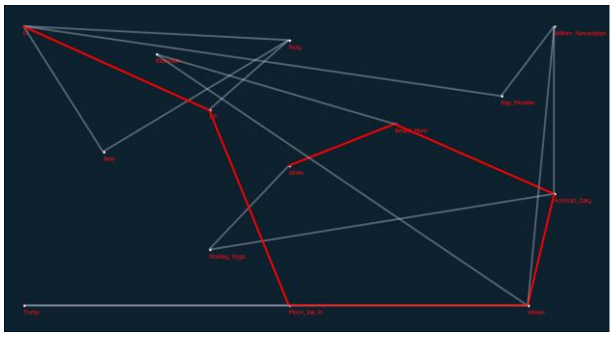


Figure 11. Second connection path from *Jones* to *IU*.

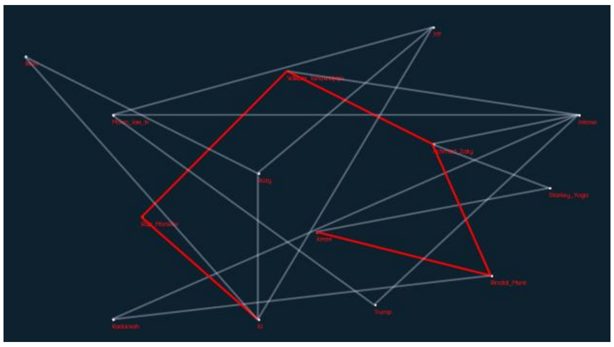


Figure 12. Third connection path from *Jones* to *IU*.

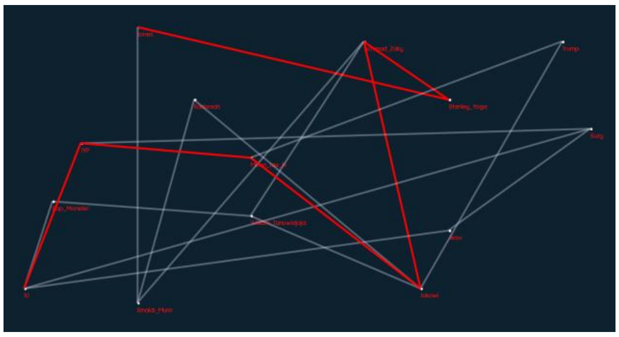


Figure 13. Fourth connection path from *Jones* to *IU*.

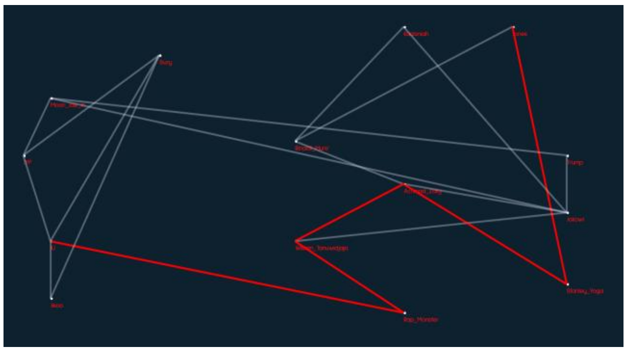


Figure 14. Fifth connection path from *Jones* to *IU*.

Note that this process might look mundane as manually finding connections with hand would be deemed visible. However, in the real world, a relation graph would not be as easy as what's being portrayed in the figures above, but Graph theory manages to ease the computation up a lot.

With all these possibilities unveiled, *Jones* becomes aware of every single path that might lead him to his dream girl. He, then, could execute a strategy himself with any desired bypass connections he would obtain as well. Taking Figure 12 as the sequel, *Jones*' desired outcome might be as follow:

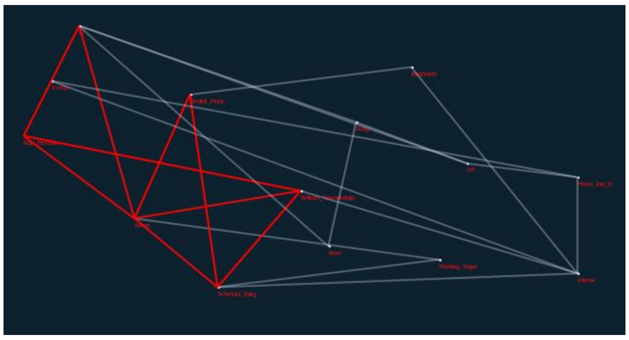
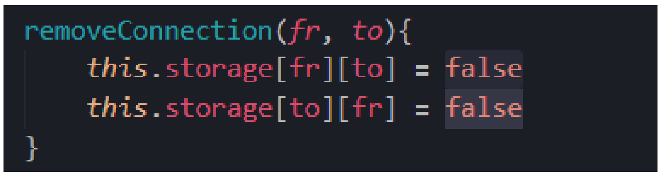


Figure 15. Jones' desired connection-list outcome in the process of acquainting *IU*.

Along the way to befriend *IU*, *Jones* also gets to befriend other *people* in the connection path. Another strategy that *Jones* should implement is to be especially more affable to those *people* in the connection path, otherwise *Jones* might be 'unfriended' and that connection path to *IU* might be completely disrupted, restarting a new process all over again with a fewer list of connection-path possibility.

The process of 'unfriend' can be easily implemented in the Graph object as well.



```
removeConnection(fr, to){
    this.storage[fr][to] = false
    this.storage[to][fr] = false
}
```

Figure 16. Implementation of people's relation removal.

By appending the above code snippet as one of the Graph object's method, *people* has now capability to 'unfriend' others should they are not up to the *people*'s liking.

Again, using Figure 15 as the sequel for 'unfriend', and for instance, *Jones* gets 'unfriended' even from the start by *Rinaldi_Munir*, say, because of his haughtiness. The graph would, then, looked as follow

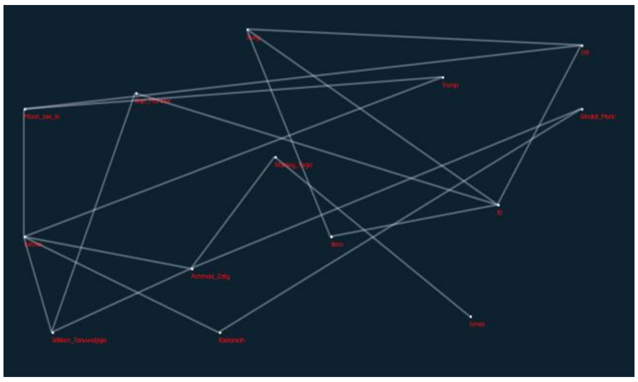
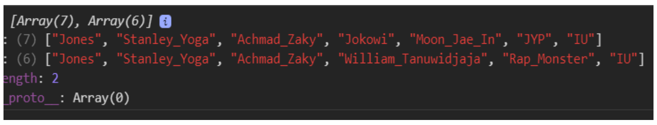


Figure 17. Graph of connectedness from *Jones* to *IU*, being a continuation from Figure 15.

Notice that *Jones* has now lost a good relation with *Rinaldi_Munir*, ending up with only a single direct connection - *Stanley_Yoga*. *Should Jones* be 'unfriended' by *Stanley_Yoga* as well, then, there is no more way for *Jones* to even befriend anyone, moreover his dream girl (*IU*). Note again that to be acquainted with someone in real life, connections (more than zero degree of separation) is not required. Every acts of 'befriend' or 'unfriend' in this paper is only for the sake of demonstration and academic purpose – to represent a better graph.

Resuming the last action where *Jones* has one direct connection left (Figure 17), the number of connection path between *Jones* and *IU* would consequently fall to two.



```
[Array(7), Array(6)] i
: (7) ["Jones", "Stanley_Yoga", "Achmad_Zaky", "Jokowi", "Moon_Jae_In", "JYP", "IU"]
: (6) ["Jones", "Stanley_Yoga", "Achmad_Zaky", "William_Tanuwidjaja", "Rap_Monster", "IU"]
ength: 2
_proto__: Array(0)
```

Figure 20. Jones' possible connection paths to *IU*.

It is apparent from Figure 20 that *Stanley_Yoga* and *Achmad_Zaky* play a vital role in connecting *Jones* and *IU*. Either one of them 'unfriends' *Jones*, and *Jones* has no more possibility in acquainting *IU* (at least for this paper's purpose).

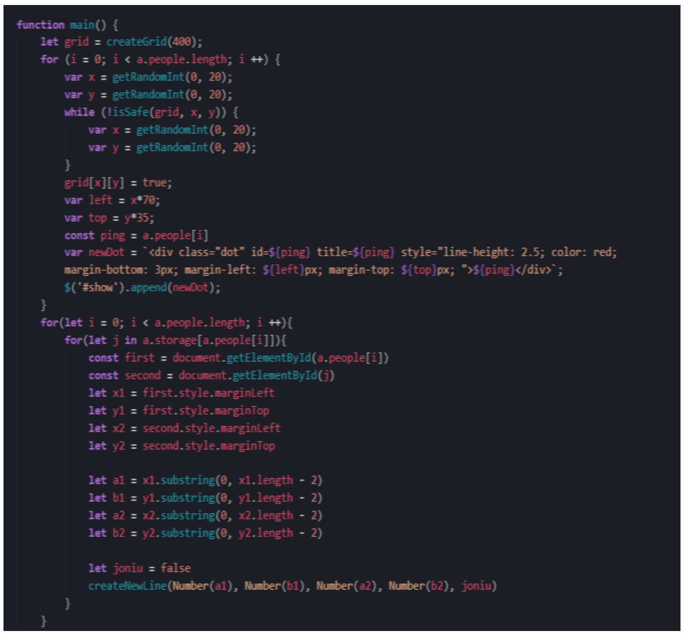
With only a couple of possibility left for *Jones* to explore, author wishes him all the best as Graph theory has helped him in identifying connection paths to his dream girl.

V. CONCLUSION

A guy having a dream girl is normal. It is motivating, but do not be disappointed in the fact that there is no way to befriend her. Opportunity comes and goes, God knows the best, but Graph theory does the rest.

VI. APPENDIX

1. Main program that runs and results in all the figures containing graph.



```
function main() {
    let grid = createGrid(400);
    for (i = 0; i < a.people.length; i ++) {
        var x = getRandomInt(0, 20);
        var y = getRandomInt(0, 20);
        while (!isSafe(grid, x, y)) {
            var x = getRandomInt(0, 20);
            var y = getRandomInt(0, 20);
        }
        grid[x][y] = true;
        var left = x*70;
        var top = y*35;
        const ping = a.people[i]
        var newDot = `<div class="dot" id=${ping} title=${ping} style="line-height: 2.5; color: red;
        margin-bottom: 3px; margin-left: ${left}px; margin-top: ${top}px; ">${ping}</div>`;
        $('#show').append(newDot);
    }
    for(let i = 0; i < a.people.length; i ++){
        for(let j in a.storage[a.people[i]]){
            const first = document.getElementById(a.people[i])
            const second = document.getElementById(j)
            let x1 = first.style.marginLeft
            let y1 = first.style.marginTop
            let x2 = second.style.marginLeft
            let y2 = second.style.marginTop

            let a1 = x1.substring(0, x1.length - 2)
            let b1 = y1.substring(0, y1.length - 2)
            let a2 = x2.substring(0, x2.length - 2)
            let b2 = y2.substring(0, y2.length - 2)

            let joniu = false
            createNewLine(Number(a1), Number(b1), Number(a2), Number(b2), joniu)
        }
    }
```

VII. ACKNOWLEDGMENT

First and foremost, author would love to thank his family for supporting him in everyday choices that he takes. Author would also not forget to thank Bandung Institute of Technology as well as all of its lecturers, including those from the Informatics background for the opportunity to write this original paper. Lastly, author would thank Kevin Bacon for popularizing the notion of Six Degrees of Separation, and IU for existing in this world.

REFERENCES

[1] Biggs, Norman (1993) Algebraic Graph Theory (2nd ed.). Cambridge University Press. ISBN 978-0-521-45897-9

[2] Guare, John (1990) Six Degrees of Separation. Vintage. ISBN 9780679734819

[3] Karinthy, Frigyes (1929) Everything is Different

[4] Ugander, J., Karrer, B., Backstrom, L., & Marlow, C. (2011). The anatomy of the Facebook social graph.

[5] Geeks for geeks, Undirected graph, accessed 4th December 2019, <https://www.geeksforgeeks.org/graph-data-structure-and-algorithms/>

[6] Carla Osthoff, Adjacency List, accessed 4th December 2019, <https://www.researchgate.net/figure/A-directed-graph-represented-by-adjacency-lists_fig3_274143903>

[7] Oreilly, Adjacency Matrix, accessed 4th December 2019, <https://www.oreilly.com/library/view/php-7-data/9781786463890/6bddc759-aa6e-4f1a-a2b3-a5460b5de121.xhtml>

[8] "Actor's Hollywood career spawned 'Six Degrees of Kevin Bacon'". Telegraph. 6 June 2011. Retrieved 7 May 2012.

DECLARATION

I hereby declare that this paper is of original work, neither an adaptation, nor a translation of any existing paper, and not an act of plagiarism.

Bandung, December 5th 2017

[illegible]

Jones Napoleon

Jones Napoleon Autumn
13518086