Interactive graph constructing on graph theory application development

Dairina¹, M Ihsan²,³ and M Ramli²

¹ Mathematics Graduate Study Program, Syiah Kuala University
² Department of Mathematics, Syiah Kuala University
³ Multimedia Research Group, Department of Mathematics, Syiah Kuala University

E-mail: mahyus@unsyiah.ac.id

Abstract. This paper discusses the steps of constructing simple graphs on graph theory application development. Most graphs constructing on application development use an adjacency matrix and an adjacency list before the graph is constructed. However, it is considerably difficult to construct a graph in such way. The steps of constructing a graph in this work were started by directly construct a graph on stage. The graph was constructed by creating the vertices and every edge can be created after creating two vertices as its endpoint. The graphs can be changed by moving the vertex and deleting vertices and edges. Furthermore, the graph can be changed by adding vertices and edges. The adjacency list is then obtained from the given graph and used for further processes. The steps of constructing graph are interactive so that constructing graph can be made easier and more efficient.

1. Introduction
Graph theory is one of the fields in mathematics that accommodate various problems in real life. Many problems can be modeled in the form of a graph, such as shortest-path problem, a scheduling problem, simple mapping graph problem, and traveling salesman problem (TSP). Developing graph theory application is needed to fulfill the need for solving a large complex problem, including accuracy, speed, and interactive use.

A graph \( G = (V, E) \) consists of a non-empty set vertex \( V(G) \), a set edge \( E(G) \) and a relation that map each edge to two vertices [1]. A graph can be represented in many ways, namely sets [2], [3], a graphical form [2, 3], and [4], an adjacency matrix [5], and an adjacency list [5].

In graph theory application development, a graph is formed by inputting an adjacency matrix or an adjacency list first. Afterward, a graph is constructed, and information related to coordinates is added. However, inputting an adjacency matrix or an adjacency list is considered not effective. For adjacency, it is caused by a large number of entries that is \( n^2 \). Meanwhile, for adjacency list, the problem arises when determining the pair of vertices that form edges. Another relatively easy way is drawing the graph first. Afterward, an adjacency matrix and an adjacency list can be obtained from the graph.

Inputting a graph using its adjacency matrix can be found in a few of studies, such as finding a spanning tree of a simple graph [2]. Another example is plotting a graph from its adjacency matrix [6], [7], [8], [9], [10], and each vertex's coordinate can be obtained in the following ways: data file, random coordinate and using mouse pointer. Another way to draw a graph is by determining the adjacency list...
first. For example, a graph is plotted from an adjacency list, and each vertex's coordinate is obtained from data file [11].

Graph application can be used easier if the graph is visually and interactively portrayed. Vertices and edges can be directly inputted by clicking and tapping on stage. Deleting and adding vertices and edges can be done by double-clicking and changing vertex's position can be done by clicking and dragging the vertex. Afterward, an adjacency matrix and an adjacency list can be obtained from the graph and are used for further processes. Therefore, this research will explain the steps needed to draw and construct a simple graph interactively, which can be used in developing a graph theory application.

2. Methodology
In an undirected simple graph, there exists an edge $uv$ which connects vertex $u$ and vertex $v$. In this context, edge $uv$ is identical to edge $vu$ which connects vertex $v$ and vertex $u$.

The graph is drawn in two-dimensional stage with coordinate $(0,0)$ located in the top-left corner, corresponding to a monitor's coordinate. On stage, the horizontal axis ($x$-axis) is always positive. The same thing applies to the vertical axis ($y$-axis), see figure 1.

![Figure 1. Coordinate of stage.](image)

Constructing a graph is started by drawing or inputting vertices. A vertex can be inputted by clicking directly on stage. The vertex is a small circle with the same size and is marked with a label like as numbers, letters, or indexed variables (such as $v_1, v_2, v_3$). Its label will be located in the center of the corresponding vertex, see figure 2.

![Figure 2. Vertex's label in the centre.](image)

Vertex’s label appears in an ascending order such as one, two, three, a, b, c, or $v_1, v_2, v_3$ and so on. The position or the coordinate of each vertex is obtained from the center where the vertex is located. The tolerance of placing a vertex must meet the minimal distance so the edge between two vertices can be seen. The distance between any two vertices can be calculated using Euclidian formula. The tolerance between any two vertices is $d_i > 2R + k$, and the distance between any two vertices can be computed using equation 1.

$$d = \sqrt{(x_c - x_i)^2 + (y_c - y_i)^2}$$

(1)
where $d_i$ is the distance between any two vertices, $R$ is a radius of vertex, $k$ is tolerance distance between any two vertices, $(x_c, y_c)$ is the center point of the added or moved vertex, and $(x_i, y_i)$ is the center point of another vertex on stage.

![Figure 3. Distance between any two vertices.](image)

An edge can be added after creating two vertices in minimal. The edge can be drawn by clicking two vertices that will be connected. The edge will be a straight line from the center point of one vertex to another. The two vertices can be chosen randomly and do not have to be in ascending order, see figure 4.

![Figure 4. Drawing edge with two random vertices.](image)

For each pair of vertices that is selected, there can only be one edge at maximum (vertex with multiple edges is not allowed). A vertex connected by an edge can be connected with another edge. Thus, that vertex has a degree at least 2. Adding a vertex can be done by creating a small circle with a label on stage. The size of the vertex is the same with other vertices on stage. The label of the vertex is adjusted by the label of the previous vertex.

Deleting a vertex can be done by deleting the small circle, therefore the label and the edges associated with that vertex will also be removed. Changing the position of a vertex can be accomplished by dragging the circle directly on stage. Deleting an edge can be done by choosing two vertices and removing the line that connects the vertices on stage. Adding an edge can be achieved by connecting the center point of the first vertex and the second vertex.

3. Results and discussion

Given a simple graph as shown in figure 4, the graph can be drawn by doing the steps presented in this research.

3.1. Drawing and modifying a graph

Drawing the graph can be done using following ways:

- The first vertex up to the last vertex can be added by clicking the position needed on stage, see figure 5

![Figure 5. Drawing a vertex.](image)
• The edge can be drawn by clicking two vertices that are required to connect, see figure 6.

![Figure 6. Drawing an edge.](image)

• Adding a vertex and edge can be done using step 1 and 2.
• A vertex cannot be added if the tolerance between the vertices is less than \( d \) see figure 3.
• The position of a vertex can be changed by clicking and dragging the vertex, see figure 7.

![Figure 7. Changing vertex’s position.](image)

• Deleting a vertex can be done by double-clicking the vertex needed to remove, see figure 8.

![Figure 8. Deleting an edge.](image)

• To draw a complete graph, a feature (clicking a button) can be added after creating all the vertices.

3.2. Storing a Graph

The data of graph shown in figure 4 is needed for further processes. The data of the graph can be presented in the form of adjacency matrix or adjacency list. In this research, the information of the graph which has been created are the number of vertices and edges, the set of edge, and the coordinate for each vertex will be stored in the form of variables, namely adjacency lists and arrays. The process needed to save the information as shown in figure 4 is as follows.

• Each vertex drawn on stage will be counted, and the number will be stored in a variable \( nV \), see figure 10.

![Figure 9. Saving the number of vertices.](image)
• Each edge drawn on stage will be counted, and the number will be stored in a variable \( nE \), see figure 11.

![Graph with edge count](image)

**Figure 10.** Saving the number of edges.

• Aside from the number of vertices and edges, for each vertex drawn on stage, the position of the vertex will be stored in the form of an array, see figure 12.

![Vertex positions](image)

**Figure 11.** Saving the position of each vertex.

• Aside from the number of edges, for each edge drawn on stage, two vertices for which the edge connects will be stored in the form of an adjacency list, see figure 13.

![Adjacency list](image)

**Figure 12.** Saving pair of vertices that are connected by an edge.

The information obtained from the graph are stored and used for an algorithm or another related process. The information of the graph are also needed to save the graph in storage so the graph can be displayed if needed. Saving the graph can be done by storing information in a file, such as the number of vertices, the coordinates of each vertex, the number of edges, and the endpoint of each edge.

4. Conclusion

The steps of drawing a graph presented in this research are started by drawing the graph directly on stage. Afterward, an adjacency matrix and an adjacency list are obtained from the graph and used for further processes. Thus, the steps presented in this research are more interactive. Therefore, drawing a graph in graph theory application can be made easier.

**Acknowledgments**

The authors express gratitude to Directorate of Research and Community Services for grant of Applied Products Research.
References
[1] West D B 2001 Introduction to Graph Theory (Pearson Education, Inc.)
[2] Goodaire G E and Parmenter M M 2002 Discrete Mathematics with Graph Theory (New Jersey: Prentice-Hall)
[3] Rosen K H 2012 Discrete Mathematics and Its Applications 7th (ed) (New York: The McGraw-Hill Companies, Inc.)
[4] Wilson R J 1996 Introduction to Graph Theory 4th ed. (Addison-Wesley)
[5] Cormen T H, Leiserson C E, Rivest R L and Stein C 2009 Introduction to Algorithms 3rd (London: The MIT Press)
[6] Puspitorini S 2008 Media Informatika 6 39-55.
[7] Nugraha D W 2011 Jurnal Ilmiah Foristek 1
[8] Side S, Sutra and Sulaiman 2013 Jurnal Sainsmat 2 131-139
[9] Lukas S, Anwar T, and Yuliani W 2005 Seminar Nasional Aplikasi Teknologi Informasi
[10] Mardlootillah H I, Suyitno A, and Arini F Y 2014 UNNES Journal of Mathematics 1
[11] Ardiansyah, Efendi F S, Syaifullah, Pinto M, Pujianto, and Tempake H S 2010 J. Infor. 4