Int. J. Engng Ed. Vol. 23, No. 3, pp. 499–507, 2007
Printed in Great Britain.

0949-149X/91 $3.00+0.00
© 2007 TEMPUS Publications.

MOGRAPH: Mobile Graph Algorithms Library for Engineering Students*

MUSTAFA MURAT INCEOGLU and BIROL CILOGLUGIL
Ege University, Department of Computer Education and Instruction Technology, Bornova 35100, Izmir, Turkey. E-mail: mustafa.inceoglu@ieee.org; birol.ciloglugil@ege.edu.tr

KORHAN KARABULUT
Yasar University, Department of Computer Engineering, Bornova 35100, Izmir, Turkey. E-mail: korhan.karabulut@yasar.edu.tr

In this study, a mobile application called MOGRAPH, which has been developed for the teaching graphs, is presented. By using MOGRAPH, students can draw and edit previously formed graphs, apply Depth First Search (DFS), Breadth First Search (BFS), Dijkstra's Shortest Path, Euler Path/Circuit, Hamilton Path/Circuit and Graph Coloring algorithms on the undirected (weighted or unweighted) graphs created by them and take a quiz to test their knowledge. Results show that at least 79% of the students have liked the educational features of the MOGRAPH package and have thought it would be beneficial for future use.

Keywords: mobile learning; graph algorithms; PDA

INTRODUCTION

WITH THE WIDESPREAD USAGE of mobile devices, individuals become less dependent on time and space for their activities. In the case of education technologies, in class education has become increasingly supported by distance education in time. As the rapid development of information and communication technologies continues, in the near future, mobile integrated devices will support education activities without dependence on time and space. This kind of education activity is called mobile learning (m-learning) [9].

M-learning is realized by mobile phones, handheld computers and other devices especially used by the young [1]. In addition to this, in the literature, m-learning techniques are used in different branches of education. The m-learning study using health students in England [2], a study of English language education on Japan university students [3], a project to train home economics teachers in Finland [4] and an m-learning project for environment education in South Korea [5] are good examples. In most of these projects, short message services (SMS), PDA devices and handheld computers are commonly used. As clearly described [6], there is plenty good text-based information on various topics for mobile devices. Nevertheless, unfortunately, there is no good free software present for natural sciences and mathematics, except a few projects [7–8].

Graph theory is one of the main subjects in discrete mathematics and is applicable for a wide range of areas including computer science/engineering, electrical engineering, chemistry, economics and operations research. For instance, it is used to determine whether a circuit can be implemented on a planar circuit board, to find the best route for connecting network cables in a building, to determine whether two computers are connected by a communications link (using graph models of computer networks), to solve many types of problems like finding the shortest path between two cities in a transportation network (e.g. for the traveling salesmen problem).

Many operations can be applied to graphs such as traversing a graph (to search for a particular vertex), finding the shortest path between two vertices of a graph and determining circuits and paths on a graph. Each of these operations may be carried out with the help of different algorithms. For example, DFS (Depth-First Search) and BFS (Breadth First Search) are two basic algorithms for traversing graphs. They can be used to find the shortest path from one particular node to another and to calculate the spanning tree of a connected graph. The Dijkstra algorithm is one of the most efficient algorithms to find the length of the shortest path between two vertices in a weighted graph. Finding Euler or Hamilton paths/circuits in graphs is another major subject of graph theory that is used to solve problems like the Königsberg Bridge Problem. Graph Coloring, which is related to the coloring of vertices, edges and regions of graphs, has a variety of applications to problems involving scheduling of exams at a university, frequency assignment of television channels, etc. Four-color theorem is one of the basic approaches for graph

* Accepted 15 March 2007.

coloring, which states that the chromatic number of a planar graph is not greater than four.

Nowadays, the application scope of graph theory increases each day in branches of engineering, especially in computer, electrical and industrial engineering. Graph theory is the infrastructure for many academic studies and courses. In the Ege University Computer Engineering Department, a library called MOGRAPH (abbreviation for MObile GRAPH) is developed with the aims of: facilitating the students to better understand the topic of graph theory in discrete mathematics course and do more practice, helping the students to test themselves and helping the lecturers make short quizzes to test the students' understanding. MOGRAPH is developed as an application to run on both mobile devices and desktop computers.

GENERAL INFORMATION ABOUT MOGRAPH

MOGRAPH consists of three basic logical components: an interactive graphical user interface section to create, edit, save and load graphs, an algorithm section to apply the algorithms mentioned above onto the graphs and a quiz section. These components are distributed into menus, considering the general Microsoft Windows menu structure and the main menus are 'File,' 'Edit', 'Operations' and 'Help'.

Graph operations are applied by using 'File' and 'Edit' menus. 'File' menu contains 'New', 'Open', 'Save' and 'Exit' options. The 'New' option is used to start drawing a new graph. However, if there is already a graph in the working area, the user is asked if he/she wants to save it. Previously saved graphs can be loaded by selecting the 'Open' option of the 'File' menu. After creating a graph, the user may save it onto the mobile device for future use by clicking the 'Save' option. The 'Exit' option is used to quit from the MOGRAPH application.

'Add Vertex', 'Add Edge', 'Delete Vertex', 'Delete Edge', 'Rename Vertex', 'Assign/Update Edge Weight' and 'Move Vertex' are the available options on the 'Edit' menu that are used to create and edit graphs. Figure 1 shows an example graph.

After selecting the 'Add Vertex' option from the 'Edit' menu, the user can add a new vertex by clicking (or tapping) where he/she wants to place it on the screen of the mobile device. Vertices are named automatically by using the English alphabet. Vertices can be relocated by using the 'Move Vertex' option. The user may tap on a vertex and drag it to a new position. In order to add edges between vertices, the user selects the 'Add Edge' option and clicks on the start and end vertices of the new edge; to delete edges, the user selects 'Delete Edge' option and clicks on the edge he/she wants to delete. 'Delete Vertex' option deletes vertices one by one. When this option is selected, the user is asked to click on a vertex. Then, the selected vertex and all of the edges associated with it are deleted. 'Rename Vertex' is used to rename vertices created with automatic naming. The user also can assign weights to edges of the graph and update them by using the 'Assign/Update Edge Weight' option to form a weighted graph.

The 'Operations' menu contains operations that can be applied on graphs and a quiz option. The 'Adjacency Matrix' option can be used to display the adjacency matrix of the current graph. The 'Algorithms' and 'Quiz' options are two of the basic features of MOGRAPH explained below.

The second major component of the MOGRAPH application is the 'Algorithms' section that can be accessed from the 'Operations' menu. The 'Algorithms' option contains six basic algorithms that can be applied on the currently displayed graph. These algorithms are 'Depth First Search (DFS)', 'Breadth First Search (BFS)', 'Dijkstra's Shortest Path', 'Euler Path/Circuit', 'Hamilton Path/Circuit' and 'Graph Coloring.' The operations of these algorithms can be observed in a step-by-step manner.

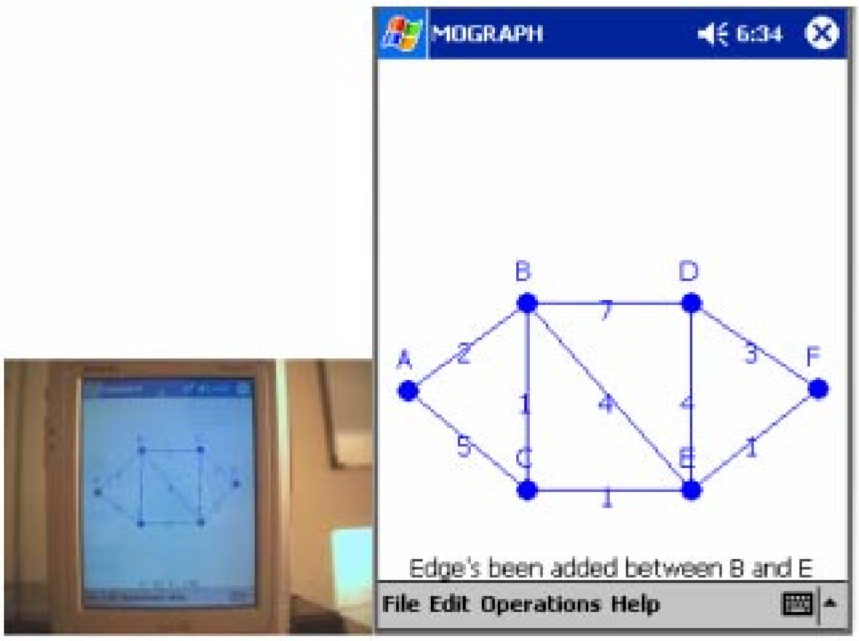
DFS and BFS are two of the main graph traversal algorithms. In these two algorithms, the user is asked to select a starting vertex. Then, the user can observe the traverse of the graph step by step. In each step, the corresponding edge to the next vertex is painted in a different color, which indicates that it is being traversed. The order of traverse is also printed at the bottom of the screen. In Fig. 2, graph traversals using the DFS and BFS algorithms are shown.

For 'Dijkstra's Shortest Path' algorithm, the user is asked to select the start and the end vertices to calculate the shortest path in between. Then the shortest path information and the distance value are shown on the screen. In addition, the shortest path is displayed in a step-by-step manner, like the first two algorithms. Figure 3 shows a shortest path example.

'Euler Path/Circuit' and Hamilton Circuit/Path' algorithms are used to detect whether the current graph contains Euler and Hamilton Paths/Circuits respectively. If Euler and Hamilton Circuit/Path algorithms cannot detect such a circuit/path, an appropriate message is displayed to the user. Otherwise, the route to be followed by the circuit/path detected is printed and it is also displayed step by step. Figure 4 shows a Hamilton Circuit example.

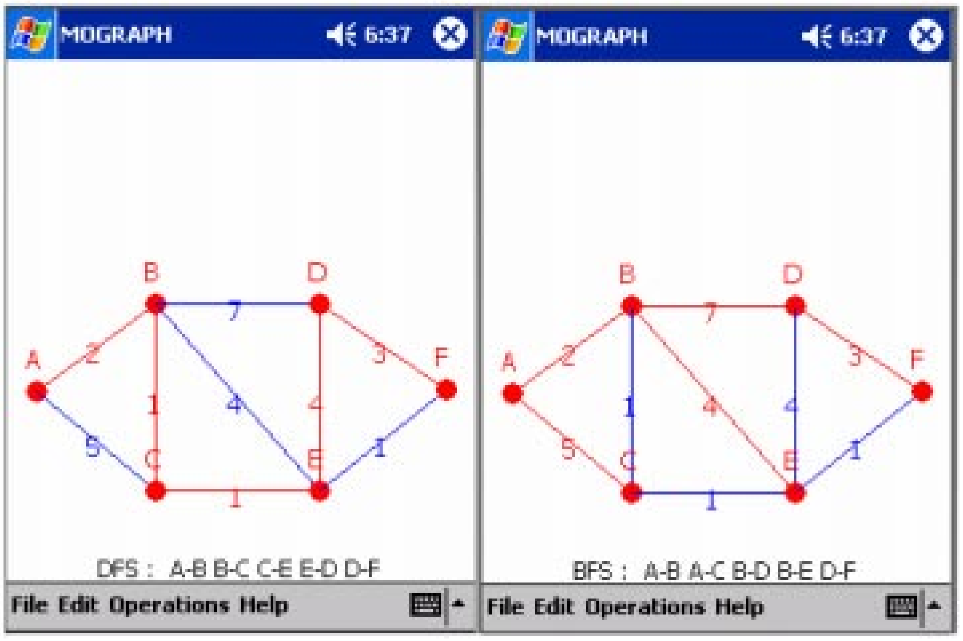
'Graph Coloring' is implemented by using the four-color theorem in MOGRAPH. If a graph is suitable to be displayed according to four-color theorem, its vertices are displayed with the fewest possible number of colors (the chromatic number of the graph). If the graph cannot be colored using the four-color theorem, a message regarding to the four-color theorem limitation is displayed. Figure 5 shows a graph with vertices colored by the four-color theorem.

The 'Quiz' option of the 'Operations' menu is



Edge's been added between B and E

Fig. 1. A graph with six vertices and nine edges.



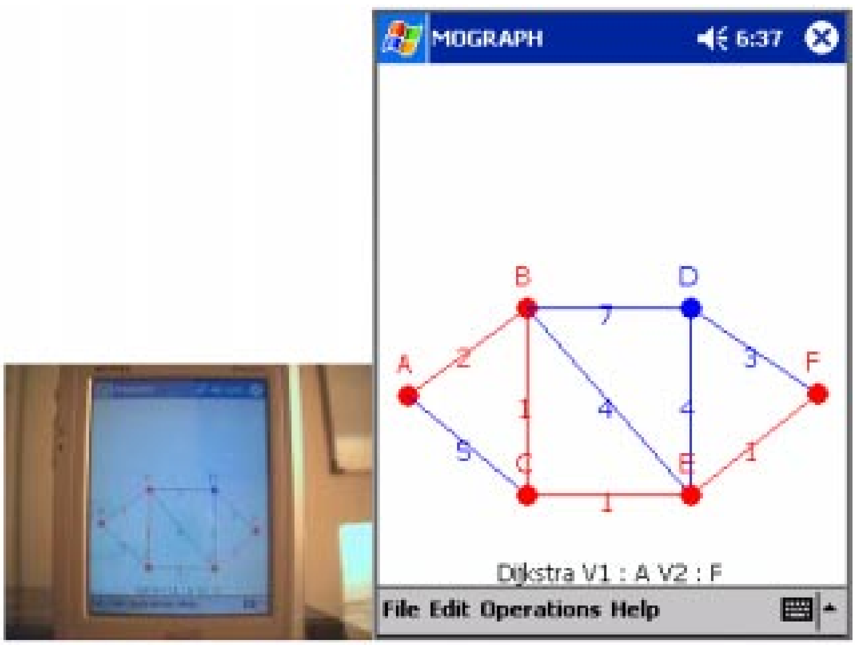
DFS : A-B B-C C-E E-D D-F

BFS : A-B A-C B-D B-E D-F

Fig. 2. DFS and BFS examples.

the third basic component of the MOGRAPH application. Each quiz session contains three consequent questions. When 'Quiz' option is selected, the application connects to the server and the user is asked the first question immediately when the server connection is established. When the user wants to quit the quiz session, he/she may click the OK button at the top left of the screen. The user is supposed to answer the questions within the given time. If the time runs out, the application automatically takes the content of the answer area as the user's answer. The user may click the OK button if he/she answers the question before the time runs out. After the first question, the second question is retrieved from the server and displayed on the client's screen. The same procedure is followed for the third question. When the user answers all the questions, the result of the quiz is displayed to the user with his/her answers and the correct answers. Figure 6 shows an example quiz screen.

The 'Help' menu contains the help documentation for algorithms and usage of the MOGRAPH application.



Dijkstra V1 : A V2 : F

Fig. 3. Shortest path example.

METHODOLOGY OF THE DEVELOPMENT PROCESS OF MOGRAPH

In this section, the infrastructure of MOGRAPH is explained in detail with the use of MOGRAPH in the Discrete Mathematics course.

MOGRAPH is an open source mobile application implemented in Microsoft Visual Studio.NET environment with C# language.

Different data structures, such as two-dimensional matrices and linked lists, can be used to represent graphs [10–13]. For MOGRAPH, since the user will be able to add, delete or edit vertices and edges dynamically at run time, using a predefined number of vertices will not be feasible. Therefore, a dynamic data structure is needed. In C#, using array lists is the most efficient solution available. Thus, in MOGRAPH, vertex and edge classes are defined with the graph class containing two array lists to store vertex and edge objects.

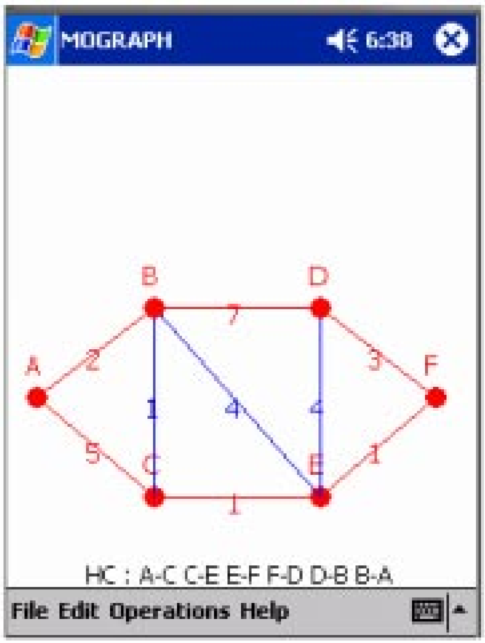
Vertex objects contain a unique ID assigned to each vertex, a vertex name and a point object to store the coordinates of the vertex on the drawing area of the device screen. Edge objects contain two vertex IDs associated with the start and end vertices of the edge and the distance value between the vertices. This is also the weight value of the edge. Vertex and edge class definitions are given in Fig. 7.

Since there is no built-in method supported by C# to detect whether a clicked point on the screen is on a vertex, on an edge or neither, two methods are added to the graph class. These methods are ‘IsClickedOnVertex’ and ‘IsClickedOnEdge’.

To detect vertex areas, Equation (1) of analytic geometry, which calculates the distance between two points is used by the ‘IsClickedOnVertex’ method. This formula is applied to all vertices on the graph. If the calculated distance is less than or equal to five pixels for a vertex, then that vertex is determined as the clicked vertex.

$$\text{Distance} = \sqrt{(X_{\text{Vertex}} - X_{\text{ClickedPoint}})^2 + (Y_{\text{Vertex}} - Y_{\text{Clickepoint}})^2} \qquad (1)$$

To detect whether a clicked point is on an edge, ‘IsClickedOnEdge’ method examines each edge one by one with the clicked point data. This process contains two steps. At the first step,



HC : A-C C-E E-F F-D D-B B-A

Fig. 4. Hamilton circuit example.

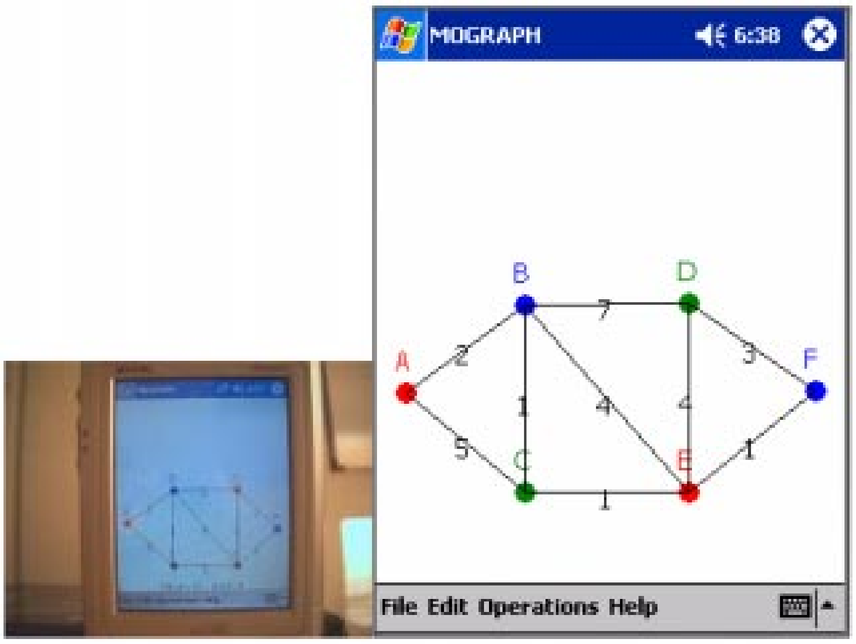
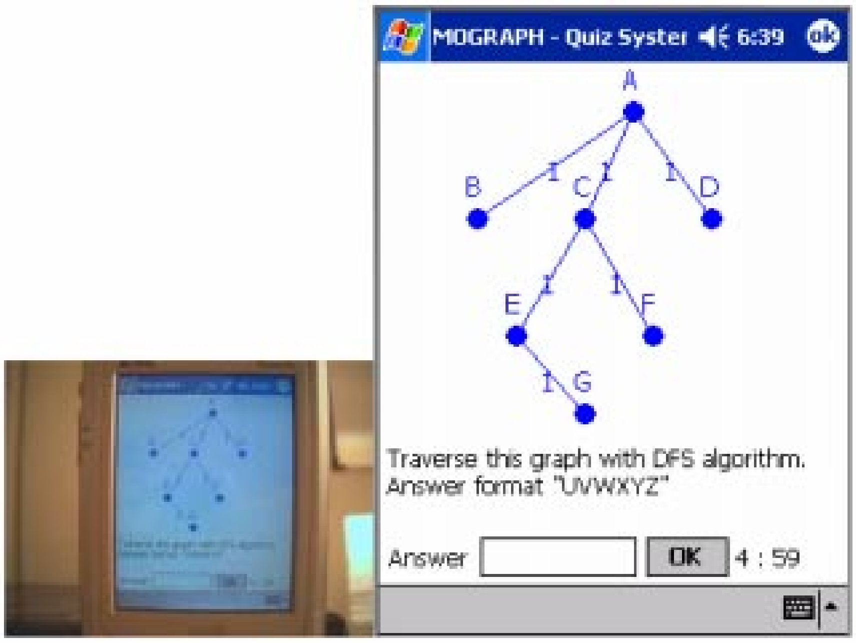


Fig. 5. Graph coloring example.

Equation (2), which calculates the distance of a point to a line is used. Thus, it calculates the distance of the clicked point to the line that represents the currently examined edge. In Equation (2), StartVertex and EndVertex represent the two vertices of the current edge. If the calculated distance is less then a predefined value (two pixels), the second step is applied. In the second step, the current edge's start and end vertices' points are checked with the clicked point coordinates to see if the clicked point is on the line segment restricted with the start and end vertices of the edge.

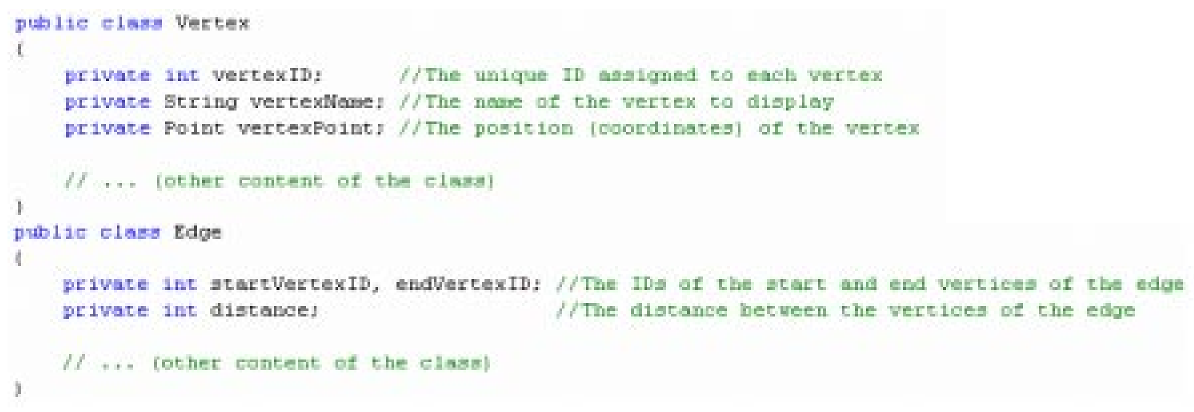
$$\text{Distance} = \frac{|a * X_{ClickedPoint} + b * Y_{ClickedPoint} + c|}{\sqrt{a^2 + b^2}}$$

$a = Y_{StartVertex} - Y_{EndVertex}$
$b = X_{EndVertex} - X_{StartVertex}$
$c = (X_{StartVertex} * Y_{EndVertex}) - (Y_{StartVertex} * X_{EndVertex})$



Traverse this graph with DFS algorithm.
Answer format "UVWXYZ"

Fig. 6. A quiz screen.



```
public class Vertex
{
    private int vertexID;      //The unique ID assigned to each vertex
    private String vertexName; //The name of the vertex to display
    private Point vertexPoint; //The position (coordinates) of the vertex

    // ... (other content of the class)
}
public class Edge
{
    private int startVertexID, endVertexID; //The IDs of the start and end vertices of the edge
    private int distance;                   //The distance between the vertices of the edge

    // ... (other content of the class)
}
```

Fig. 7. Vertex and edge class definitions.

where a, b and c are calculated from:

$$\frac{X - X_{StartVertex}}{X_{StartVertex} - X_{EndVertex}} = \frac{Y - Y_{StartVertex}}{Y_{StartVertex} - Y_{EndVertex}} \tag{2}$$

When the user adds a new vertex or edge, a new vertex or edge object is created and added to the vertices or edges array list respectively. When the user decides to delete an edge, the clicked point is examined by the 'IsClickedOnEdge' method explained above and if an edge is clicked over, the detected edge is removed from the edges array list. The deletion of vertices is implemented in a similar way. 'IsClickedOnVertex' method is used and if a vertex is detected on the clicked area, all of the edges associated with the vertex are removed from edges array list first and then the vertex itself is removed from vertices array list. For renaming a vertex, the selected vertex is determined in the same way as in the delete vertex operation. After that, the name field of the selected vertex is updated on the vertices array list. 'Assign/update edge weight' operation first detects the clicked edge and assigns/updates the weight value of the detected edge. 'Move Vertex' option updates the coordinates of the selected vertex on the vertices array list after detecting it.

The 'Open' and 'Save' options of the 'File' menu are used to load and save graphs in two formats; binary and XML. Binary files occupy less storage space than XML files, however, this is not a crucial advantage because the output graph file sizes are not large enough to be of consideration. Conversely, XML file format has the advantage of being a commonly used standard that creates application-independent documents and data. Users may also create new XML files or edit previously created XML files as an alternative way of graph creating. Thus, both of this file formats are supported by MOGRAPH to let the user select the appropriate one for himself/herself.

General infrastructure of the implementation of the first five algorithms (DFS, BFS, Dijsktra, Euler & Hamilton Path/Circuit) is the same. Since Graph Coloring's concept is different from these five algorithms, it is implemented with other data structures as explained below.

As the vertices and edges are stored in array lists, when any of the first five algorithms is to be applied, the adjacency matrix of the current graph is created with the data in array lists. The adjacency matrix is created before applying the algorithms, because all five algorithms use the adjacency matrix as an input to operate on. Since the adjacency matrix of the graph is created from the vertices array list, the order of the adjacency matrix is the addition order of the vertices to the array list. Thus, in MOGRAPH, these five algorithms are applied according to the addition order of the vertices on the graph. During the execution of these algorithms, the results are stored in a newly created array list called 'algorithmResult' that stores the path data. When the execution of an algorithm finishes, a timer for displaying the results of the algorithm step by step is enabled. This timer ticks with 1000-millisecond intervals and takes an edge of the path data stored in 'algorithmResult' at each tick and calls the paint method of the form to display the path (algorithm result) in a step-by-step manner.

DFS and BFS algorithms use the basic algorithm structure of [12], where a stack and a queue are used to traverse the graph respectively. These algorithms are extended in MOGRAPH to store the results on the 'algorithmResult' array list explained above. During the execution of the whole graph, 'algorithmResult' array list is filled with the information about the route to be followed. Then, in the paint method of the form, the route is displayed with the help of the timer.

Dijkstra's shortest path algorithm is implemented by using the algorithm in [10–12]. The algorithm is extended to store the shortest path values to each vertex on an array and the shortest path information to each vertex on an array list with the vertices of the shortest path. Thus, this algorithm also calculates the shortest path from the starting vertex to all of the other vertices. After calculation, this algorithm also puts the shortest path information on the 'algorithmResult' array list.

The Euler Path/Circuit algorithm aims to traverse the graph by visiting each edge only once. Euler Path/Circuit algorithm first checks if the graph meets the Euler Path/Circuit conditions described in [10]. According to these conditions, an Euler Circuit exists only if all vertices have an even number of adjacent vertices and an Euler Path exists only if, at most two vertices have an odd number of adjacent vertices with the rest of the vertices having even number of adjacent vertices. If the graph meets one of these conditions, a modified BFS algorithm is applied to determine the traverse order of the path/circuit detected. The Hamilton Path/Circuit algorithm aims to traverse the graph by visiting each vertex only once. The implementation infrastructure of this algorithm is the same as that of the Euler algorithm, but the checked conditions and the modified BFS algorithm are special to Hamilton.

Graph Coloring is a very detailed and active working area of the graph theory, which includes subjects like vertex coloring, edge coloring, four-color theorem and many more [14]. Since the scope of MOGRAPH is to offer an introduction to Graph Coloring, the four-color theorem is selected for implementation. If a graph is planar, then it can be colored using the four-color theorem. The implemented Graph Coloring algorithm takes the first vertex of the vertices array list and starts the coloring process from that vertex. Then its adjacent vertices are taken into account in sequence. This process continues until all vertices are colored. During the color determination process, each vertex is colored with an appropriate color by checking its adjacent vertices' colors.

The quiz section of MOGRAPH is implemented using a web service. When the user starts a quiz, MOGRAPH connects to the server machine using the 'MographSS' web service running on it. The quiz data are stored on an SQL Server database at the same server machine. Graphs are stored as adjacency matrices in the database. 'MographSS' web service has five web methods: TakeQuiz, Vertices, Edges, Question and Answers that connect to the database through stored procedures, get/set appropriate data and serve the data to MOGRAPH clients. These web methods send data to MOGRAPH clients in XML format. When the first server connection to the web service gets the first question, web methods are called in the following order: TakeQuiz, Vertices, Edges and Question. TakeQuiz web method selects three random questions from the SQL Server database. A unique ID is also assigned to each user in TakeQuiz. Then, the Vertices, Edges and Question web methods are executed in order to send the vertices, edges and question data respectively to the user. For the second and third questions, Vertices, Edges and Questions web methods are invoked again with the same order. At the end of the third question, Answers web method is called. This web method stores the user's results on the database for future use and sends the answers to the user.

The Discrete Mathematics course in the Ege University Computer Engineering Department lasts 14 weeks and is taught for 3 hours per week. Course contents are: propositions (1 week), sets and set theory (1 week), functions (1 week), matrices (1 week), algorithms for sets, functions and matrices (2 weeks), mathematical reasoning (2 weeks), counting techniques (1 week), relations (1 week), graph theory, applications and algorithms (3 weeks) and trees (1 week). Boolean Algebra and Modeling Computation subjects present in the textbook [10] are not included in this course, but they are studied in more detail in Logic Design and Automata Theory courses.

A group of 40 student volunteers were selected from 154 students who had taken the Discrete Structures course in order to test the developed application. These students were given an hour of education on the usage of MOGRAPH using a computer, PDA and a projector. Then, with the aid of an instructor, the students have downloaded and executed the application on the PDA devices given to them, they have used and tested several features of MOGRAPH and they had answered the quiz questions.

The MOGRAPH application was downloaded to a PDA only once and only during the quiz section, The PDA connected to the server and stayed online using WLAN or GPRS. For the quiz section, PDAs connect to the server using WLAN in the campus and by using GPRS from outside of the university. In the study made with the students, ASUS MYPAL (for WLAN support) and I-mate (for GPRS support) brand of PDA devices are used.

The quiz study is realized to test the performance of the application when a group of students are connected to the server simultaneously. In the quiz study, all students were asked to use the quiz section at the same time. Each student had been checked to see if they were online. In the observation, all students except three, whose devices had problems, could connect to the server for the quiz section.

After the 1 hour laboratory study, the students were asked to answer a questionnaire composed of seven questions with Likert type answers and two open-ended questions to evaluate the advantages and disadvantages of MOGRAPH. A total of 37 out of 40 questionnaire forms were returned, so that the participation ratio was calculated as 92.5%. Answers to the questionnaire are summarized in Table 1 (five totally agree, four partially agree, three neither agree nor disagree, two partially disagree, one totally disagree).

Results derived from the questionnaire are: students were satisfied with the graph theory education at the class; they had understood the topics taught about graphs and the examples given, they had agreed that the MOGRAPH application could be easily used as an education tool and they had recommended the use of MOGRAPH in the teaching of graph theory.

Table 1. Questionnaire results for 37 students

Question number	Question	Mean (A)	Satisfaction per cent A / 5 (%)	Standard deviation
1	Are you satisfied with the teaching about graphs?	4.054	81	0.468
2	Did you understand what is taught about graphs?	4.405	88	0.551
3	Could you easily install the MOGRAPH application?	4.297	86	0.878
4	Is it easy to use to the MOGRAPH application?	3.865	77	0.855
5	Did you find the MOGRAPH application useful?	3.930	79	0.759
6	Can you say that MOGRAPH application is helpful in learning the topics about graphs?	4.189	84	0.739
7	Do you recommend using MOGRAPH application in the next years?	4.325	87	0.747

Seven students had problems in the download and installation of the MOGRAPH package and stated these in the questionnaire. Some positive comments stated in the open-ended questions are: 'Move vertex feature is very good', 'A fun application', 'This application should include all other topics in the course', 'These kinds of experiments that we have participated in, encourages us', 'Visual simulations of the algorithms were nice', 'Visual usage of the application increases my interest and will to learn'; while the negative comments are: 'There has to be shortcuts for the menus', 'It should show graphs like hypercube and complete graph automatically', 'A little bit slow to load', 'It takes some time to download and install the application'. All positive and negative comments were evaluated and work on MOGRAPH was continued to make it more efficient. Recent additions, new features, user documentation and the latest version of the MOGRAPH package can be found at the following URL: http://efe.ege.edu.tr/~birol/mograph.

CONCLUSION

MOGRAPH is an open source, improvable package developed to support the teaching of graph theory topics in discrete mathematics courses with practices and quizzes, in order to help the students study on their own and to help the lecturers test the students' levels of understanding.

The first version of MOGRAPH package was tested by a group of selected students, as there were only a limited number of devices, and a questionnaire was applied to test the benefits. Results show that at least 79% of students liked the educational features of the MOGRAPH package and thought it would be beneficial in the future.

REFERENCES

1. M. J. Wang, R. M. Shen, R. Tong, F. Yang and P. Han, Mobile learning with cell phones and pocket PCs. *Lecture Notes in Computer Science* **3583**, 2005, pp. 332–339.
2. G. Walton, S. Childs and E. Blenkinsopp, Using mobile technologies to give health students access to learning resources in the UK community setting. *Health Information and Libraries Journal* **22**, 2005, pp. 51–65.
3. P. Thornton and C. Houser, Using mobile phones in English education in Japan. *Journal of Computer Assisted Learning* **21**, 2005, pp. 217–228.
4. P. Seppala and H. Alamaki, Mobile learning in teacher training. *Journal of Computer Assisted Learning* **19**, 2005, pp. 330–335.
5. K. W. Leeand J. H. Lee, Design and implementation of mobile-learning system for environment education. *Lecture Notes in Computer Science* **3480**, 2005, pp. 856–862.
6. B. P. Heath, R. L. Herman, G. G. Lugo, J. H. Reeves, R. J. Vetter and C. R. Ward, Project Numina: Enhancing student learning with handheld computers. *Computer* **38**, 2005, pp. 46–53.
7. M. M. Inceoglu, A discrete mathematics package for computer science and engineering students. *Lecture Notes in Computer Science* **3482**, 2005, pp. 538–546.
8. L. Bitincka and G. E. Antoniou, PDA-based Boolean function simplification: a useful educational tool. *Informatica* **15**(3), 2004, pp. 329–336.
9. G. Vavoula and C. Karagiannidis, Designing mobile learning experiences. *Lecture Notes in Computer Science* **3746**, 2005, pp. 534–544.
10. K. H. Rosen, *Discrete Mathematics and Its Applications*. McGraw-Hill Inc., New York, (1995).
11. R. Johnsonbaugh, *Discrete Mathematics*, Prentice-Hall, 4th edn, New York, (1997).
12. M. Waite and R. Lafore, *Data Structures and Algorithms in Java*, Waite Group Press, Corte Madera (1998).
13. R. Kruse, C. L. Tondo and B. Leung, *Data Structures and Program Design in C*, Prentice Hall International Editions, New Jersey, (1997).
14. *Graph Coloring Algorithm*, Available at: http://en.wikipedia.org/wiki/Graph_coloring, accessed: 16th Jan. 2006.

M. M. Inceoglu is an associate professor at the Ege University Computer Education and Instructional Technology Department. He received his Ph.D. degree from Ege University in 1998. His research interests include computer science education, educational technology and distance learning. He is a member of the IEEE.

B. Ciloglugil is a research assistant at the Ege University Computer Engineering Department. He received his M.Sc. degree from Ege University in 2006. His research interests include computer science education, educational technology and distance learning.

K. Karabulut is an assistant professor at the Yasar University Computer Engineering Department. He received his Ph.D. degree from Ege University in 2004. His research interests include computer science education, genetic algorithms and artificial intelligence.