



Graph Tea: Simulating Tool for Graph Theory & Algorithms

Bala Bhaskara Rao Emani¹ | Satya Srinivas Maddipati² | Reddy Krishna Rao³

¹Computer Science & Engineering, Sasi Institute of Tech. & Engg., Tadepalligudem, AP, India

² Computer Science & Engineering, Sasi Institute of Tech. & Engg., Tadepalligudem, AP, India

³ Computer Science & Engineering, Sasi Institute of Tech. & Engg., Tadepalligudem, AP, India

ABSTRACT

Simulation in teaching has recently entered the field of education. It is used at different levels of instruction. The teacher is trained practically and also imparted theoretical learning. In Computer Science, Graph theory is the fundamental mathematics required for better understanding Data Structures. To Teach Graph theory & Algorithms, We introduced Simulation as an innovative teaching methodology. Students can understand in a better manner by using simulation. Graph Tea is one of such simulation tool for Graph Theory & Algorithms. In this paper, we simulated Tree Traversal Techniques like Breadth First Search (BFS), Depth First Search (DFS) and minimal cost spanning tree algorithms like Prims.

KEYWORDS: Simulation, Graph theory, Breadth First Search, Depth First Search, Spanning tree.

Copyright © 2016 International Journal for Modern Trends in Science and Technology
All rights reserved.

I. INTRODUCTION

Graph is a collection of nodes along with the edges that connects the nodes. The study of properties, operations and applications of graphs is called Graph Theory. Graph theory is a fundamental area of mathematics required for computer scientist. It covers the basic knowledge in Data structures. In computer science, graphs are used to represent networks of communication, data organization, computational devices, the flow of computation, etc.

The aim of this paper is the comprehensive understanding of Graph theory in class room. Graph Tea is a Graph Editing framework developed under the name of GraphLab in the department of Mathematics in Sharif University of Technology. CABRI-Graph [1] and Gato/CATBox [2] are well-designed for graph algorithm visualization and animation. However, the interactively editing power of the graph structure is very limited. Tulip [3], [4] is another tool for information visualization. This tool could be used efficiently when the user knows how to program Python language.

II. GENERATING GRAPH

Generating Graphs can be done in two ways. First, Graphs are generated by clicking on Generate Graph menu item and select Random Graph. A sub window will be opened with default edges and number of vertices values. You can change the values according to your graph structure. A Graph will be displayed on working area of current Tab. To change the default properties of nodes and edges, select the node or edge. Then, In the properties window change the values of the attributes. Second, Graphs can also be generated by Clicking on new graph option. It will prompt for type of graph, Directed, Un directed. Select the type of graph. A new tab will be opened. To create a vertex, click on the working area of new tab. In this manner create number of vertices. To draw an edge between the vertices, select the source vertex, and then drag the mouse towards destination vertex. In the properties window change the weight of the edge. Figure 1 shows the properties window. Figure 2 shows Generated Graph.

Properties	
Name	Value
label	G4
directed	Yes
Vertex Labels	Yes
Edge Labels	No
Font	Arial Bold
Background	
Zoom	100%
Curved Edges	No
Allow Loops	No

Fig 1 : Properties Window

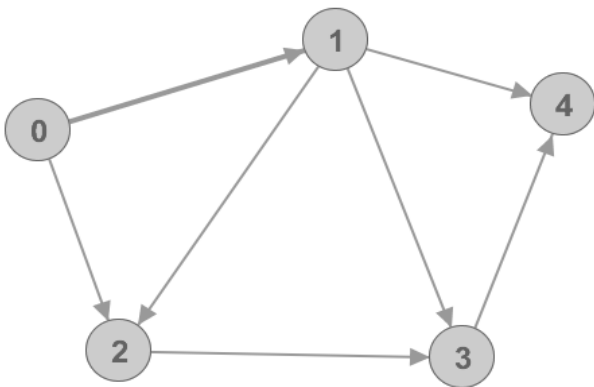


Fig 2: Generated Graph

III. ALGORITHM VISUALIZATION

A. Teaching Breadth First Search with Graph Tea

Breadth-first search (BFS) is an algorithm for traversing or searching tree or graph data

structures in the order of neighbor nodes first, before moving to the next level neighbors. The applications of BFS are finding the shortest path, bipartition and Garbage collection.

The working of BFS can be visualized by using Graph Tea. To Visualize or simulate the BFS, Select Algorithms on menu bar. It will popup all the algorithms that are simulated in Graph Tea tool. Selecting BFS generates a pop up window. In that window select Play or Play one step. Play traces the entire algorithm. Play one step trace the algorithm step by step.

In Step 1 Algorithm prompt for root vertex. Select the root vertex in the graph. The algorithm will explore the root vertex and the visited vertices are displayed in the pop up window. Once all the vertices from the root vertex are visited, the next node will be explored. In this manner all the the vertices are visited in Breadth First Order.

Given the example above, here are the steps plus a visualization to play through each step:

- Start by exploring vertex 1.
- Then visit vertices 2, 3 & 4.
- Explore vertex 2
- Then visit 0
- Exploring 3,4 & 0 does not visit any new node. Hence the algorithm will be terminated.

Fig 3: Visualizing Breadth First Search in Graph Theory

B. Teaching Depth First Search With Graph Tea

Depth-first search (DFS) is an algorithm for traversing or searching tree or graph data structures that explores as far as possible along each branch before backtracking. The applications

of DFS are finding Bridges of a graph, topological sorting and connected components. As similar to BFS, to trace DFS Select the root vertex and play

the algorithm.

```

procedure DFS(G,v):
    label v as discovered
    for all edges from v to w in G.adjacentEdges(v)
    do
        if vertex w is not labeled as discovered then
            recursively call DFS(G,w)
    
```

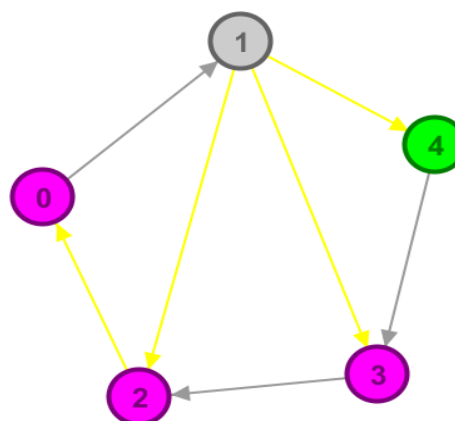
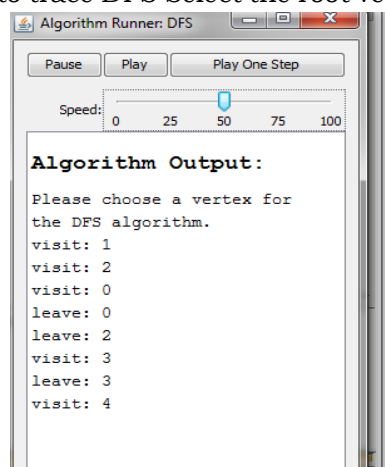


Figure 4 : Simulating DFS algorithm in Graph Theory

C. Simulating Prim's algorithm for Minimal spanning tree:

Let $G = (V, E)$ be a connected graph in which each edge $(u, v) \in E$ has an associated **cost** $C(u, v)$. A Spanning Tree for G is a subgraph of G that it is a free tree connecting all vertices in V . The cost of a spanning tree is the sum of costs on its edges.

In computer science, Prim's algorithm is a greedy algorithm that finds a minimum spanning tree for a weighted undirected graph. This means it finds a subset of the edges that forms a tree that includes every vertex, where the total weight of all the edges in the tree is minimized.

Algorithm

1) Create a set *mstSet* that keeps track of vertices

already included in MST.

2) Assign a key value to all vertices in the input graph. Initialize all key values as INFINITE. Assign key value as 0 for the first vertex so that it is picked first.

3) While *mstSet* doesn't include all vertices

....a) Pick a vertex *u* which is not there in *mstSet* and has minimum key value.

....b) Include *u* to *mstSet*.

....c) Update key value of all adjacent vertices of *u*. To update the key values, iterate through all adjacent vertices. For every adjacent vertex *v*, if weight of edge *u-v* is less than the previous key value of *v*, update the key value as weight of *u-v*

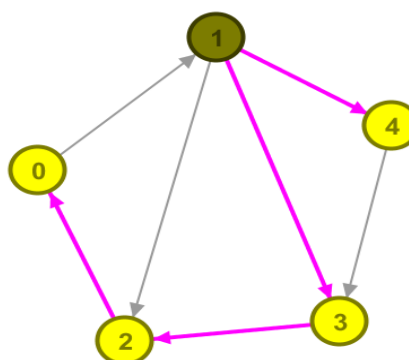
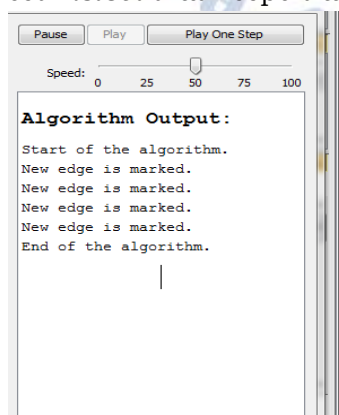


Fig 5: Simulating Prim's algorithm in Graph Theory

IV. CONCLUSION

Graph Theory & Algorithms are simulated by using the tool Graph Tea. This Tool is helpful for delivery of lectures in Graph theory & Algorithms. We observed, using Simulation as a teaching methodology improved understanding level of students

ACKNOWLEDGMENT

We Would like to thank Management of Sasi Institute of Tech. & Engg., for providing continuous support in using & delivering the classes using Graph Tea. It also encourages for preparation & publication of this paper.

REFERENCES

- [1] Y. Carbonneaux, J.-M. Laborde, and R. Madani, CABRI-Graph: A tool for research and teaching in graph theory, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 1996, vol. 1027.
- [2] A. Schliep and W. Hochstättler, "Developing Gato and CATBox with Python: Teaching graph algorithms through visualization and experimentation," *Multimedia Tools for Communicating Mathematics*, pp. 291–310, 2002.
- [3] D. Auber, D. Archambault, R. Bourqui, A. Lambert, M. Mathiaut, P. Mary, M. Delest, J. Dubois, and G. Melançon, "The Tulip 3 Framework: A Scalable Software Library for Information Visualization Applications Based on Relational Data," INRIA, Research Report RR-7860, Jan. 2012. [Online]. Available: <http://hal.archives-ouvertes.fr/hal-00659880>
- [4] A. Lambert and D. Auber, "Graph analysis and visualization with Tulip-Python," in *EuroSciPy 2012 - 5th European meeting on Python in Science*, Bruxelles, Belgique, 2012-08. [Online]. Available: <http://hal.archives-ouvertes.fr/hal-00744969>.
- [5] M. Lulfesmann, S. R. Lessenich, and H. M. Bucker, "Interactively exploring elimination ordering in symbolic sparse cholesky factorization," *International Conference on Computational Science, ICCS 2010*, vol. 1, pp. 867–874, 2010.
- [6] H. Bucker, M. Rostami, and M. Lulfesmann, "An interactive educational module illustrating sparse matrix compression via graph coloring," in *Interactive Collaborative Learning (ICL)*, 2013 International Conference on, 2013, pp. 330–335.
- [7] M. T. Heath, *Scientific Computing: An Introductory Survey*, 2nd ed. McGraw-Hill, 2002.
- [8] R. Tamassia, *Handbook of Graph Drawing and Visualization (Discrete Mathematics and Its Applications)*. Chapman & Hall/CRC, 2007.
- [9] A. Bondy and U. Murty, *Graph Theory*, ser. Graduate Texts in Mathematics. Springer, 2008.