

Finding the Way

Interactive Tutoring of the Dijkstra's Algorithm

Umang Sehgal

Information School, University of Washington
Seattle, Washington
umang93@uw.edu

Rui Wu

Information School, University of Washington
Seattle, Washington
rw420@uw.edu

Abstract—Finding the Way is an interactive story based tutor for helping students learn the Dijkstra's algorithm. The algorithm is a widely taught concept in computer science courses which usually encompasses most of CS101 lectures. This interactive visualization helps the students uncover the concepts behind the shortest path and learn the basic concepts in an accurate manner. This tool can also be categorized under the umbrella of 'Computer Aided Education'. The tool we developed presents the author with interactively seeing well connected edges and helps him/her make decisions on individual steps that help connect and complete the implementation table for the algorithm. Finding the Way develops around a themed story that helps the user comprehend the challenge to help Jack's family get to their destinations via the shortest route.

Index Terms— Dijkstra's, Shortest Path, Data Visualization, Computer Aided Education

I. INTRODUCTION

In order to help explain students/novices the concepts of graph theory in introductory computer science courses, graphical and web based tools have proven to be remarkably more effective. As a pedagogical tool, visualizations have proven to be more effective and educators have been widely known to enhance their lectures and teaching material through storytelling and visual informatics. [1] This particular project/tool would help students not fall prey to passive learning but help them through active and interaction based learning. The project makes sure to avoid a passive attitude, during the execution, it should interact continuously with the users, forcing them to predict the following step of the algorithm.

This particular project makes use of the Force Layout provided by D3 and allows room for iterative feedback back mechanism from the user. Use of such open source frameworks promise minimal effort to implement a good set of visualizations. In this paper, the analysis, storyboard, interactions have all been centered around the effectiveness of visual teaching that gets boosted with the flexibility and technical capabilities of online learning, featuring the students as drivers of their own learning progression.

A number of steps went into framing the storyboard to illustrate the tutoring process in helping understand and deliver what works best for understanding of the concepts of Dijkstra's algorithm.

Key Points of StoryBoard Design & Implementation

1. Step by step inquiring: At every single process step, the student must provide the solution while the application waits in a stand by mode, expecting the user's input to proceed to the next step.
2. Step by step evaluation: Upon the user's entry, the input is evaluated against the actual expected answer, marking the option as red if it is found to be inaccurate. Upon the correct choice selection the tool would then populate the implementation table accordingly as seen clearly in Figure 1.

			A		B	C		
1		20		80			90	
2		20		80		30	90	
3	C							
4								
5								
6								
7								

Figure 1 - Storyboard depicting the evaluation.

3. Highlighting: As visible in Figure 2 - Changing the opaqueness of the connected links and nodes to the 'Node in Action' within the Force Layout Graph itself to help the student seek clarity on available options.
4. Draggable: The student should be able to move the graph around as per his or her preference and be able to understand various limitations it offers with respect to path lengths, path directions, and possible routes.

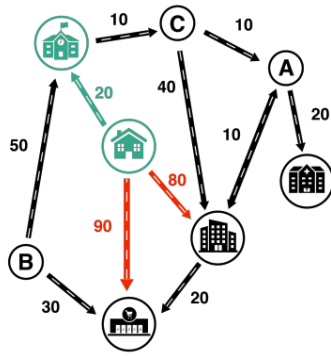


Figure 2 - StoryBoard depicting the highlight

5. Friendly: As shown in Figure 3 - The choice of colours and theme should be user friendly, all the text and themed story and interactive buttons would run to left half of the screen while the result of the interactions take place on the right half of the screen.
6. Story Based: The conceptualization of the complete concept should be a solid narrative of the story and would help the student understand and be a part of a real world scenario in solving the problem.
7. Platform Independent: The visualization should have cross-browser functionality and should pose no particular issues on different machine environments.
8. Quiz: The tool should eventually be quizzing the concept taught to the student with Multiple Choice Questions posed to him based on the context explored in the exploration.

Finding the Way

Make Dijkstra's Algorithm Easy to Understand
by Bafiz (Rui & Umang)

Have you ever heard about Dijkstra's Algorithm? Do you know how Dijkstra's Algorithm can be used to solve problems in our daily life? Now thinking back to a busy morning when you and your family members are all in a hurry and need to go to different places at a certain time, how did you and your family members decide the most efficient way to share the ride? Dijkstra's Algorithm can help you find the answer.

Jack's Family Needs Some Help

Now, let's work together to help Jack's family make their busy morning easier.

Jack is an elementary school student, and he lives with his Mom, Dad, and Grandpa. On a Tuesday morning, Jack needs to go to school as usual. His Dad needs to go to the office for an important meeting. His Mom is planning to go to the supermarket to buy some fresh vegetables to cook lunch. His Grandpa has a doctor appointment in the nearby hospital. They decide to leave the house together and share the ride. What is the fastest way for them to arrive at their destinations and how many cars do they need at minimum?

To solve this problem, we need to find the shortest paths between home and different destinations, and Dijkstra's Algorithm can help us find the answer. Let's use the directed edge-weighted graph at right as the map to solve this problem. You can click on the icon buttons below to find out where are the destinations located.

Jack → School
 Dad → Office
 Mom → Supermarket
 Grandpa → Hospital

Learn the Concept & Solve the Problem

Dijkstra's Algorithm is an algorithm for finding the shortest paths between nodes in a graph (Wikipedia). The starting node is called initial node, and the end node is called destination node. The algorithm first marks all nodes in the graph unvisited. Then, it picks the unvisited node with the lowest distance, calculates the distance through it to each unvisited neighbor, and updates the neighbor's distance if smaller. Mark visited when done with neighbors. Finally, when the destination node has been marked visited, the algorithm stops and the shortest path is found.

Let's work on the problem step by step:

	Home	A	B	C	Supermarket	Hospital
1	Home	20	50	10	90	40
2	Home	20	50	10	30	90
3	C	20	40	C	30	90
4	A	20	40	50	30	90
5	Home	20	40	50	30	70
6	Home	20	40	50	A	30
7	Home	20	40	50	30	70

Figure 3 - StoryBoard of the Interaction.

II. RELATED WORK

Torrubia et. al. propose an active learning approach called 'PathFinder' based on eMathTeacher to help students approach Dijkstra's algorithm in structured and visual manner [2]. The work highlights features provided by this application is an animated algorithm visualization panel. It shows the code, the current step the student is executing, and also where there is a user's mistake within the algorithm running. The application runs in a Java Web Start window, and this technology allows it to read and write in the client's hard disk, which gives users the feasibility of saving and retrieving graphs.

Some limitations that the work above poses are that it is oriented towards helping the student run through the code or essentially help him debug the complete pseudo-code line by line and not adhering the concepts behind the calculations and avoiding the use of interactive path selection. The work also poses a limitation by being deployed as a Java Application thus requiring every student to install and run it. Both limitations have been effectively tackled in our paper by avoiding interaction with the code but providing interaction with the actual graph. Also, our project is based on D3, which is a lightweight JavaScript library to work across multiple platforms without the need of installing anything or writing to the disk.

Makohon et. al have designed the tool for 'Transportation Engineering' students to help them aid with the understanding of the Dijkstra's algorithm [3]. The visualization makes use of colour coding techniques to help the student understand the complete algorithm. The application is again Java based and allows the student to select the source and the destination node thus formulating the route to figure out the shortest path.

One major drawback of this visualization technique as a tool for 'Computer Aided Education' is that although it allows for the user to select the source and destination node, it essentially runs through the complete algorithm by itself depicting its state in a colour coded format on the graph. This does not leave room for the student to be able to completely understand the process and think behind the algorithm. It also prevents the student from taking his/her own time in understanding the mechanics of the algorithm. It takes away the feasibility of 'self-paced' understanding. Also, this again is a Java based Desktop app and cannot be centrally changed or easily maintained by the instructor.

Lastly, Borissova and Mustakarov explore the development of developed e-learning tool allows creating, editing and saving graph structure and visualizes the algorithm steps execution. [4] The developed software system could be used also from researchers for fast visual creating of graphs and networks and for solving of implemented combinatorial graph

problems but poses certain limitations while exploring and enabling students to test and “try out” different types of graphs modifications and constructions effectively. The tool serves as a full-fetched suite for the following areas: 1) graphical area for visual processing of the graph; 2) text pane for numerical displaying of the graph structure; 3) status bar for operations and results; 4) menu and buttons toolbars as marked.

Limitation being posed is certainly the overwhelming nature of graph production and multiple operations to conduct combinatorial algorithms. It could distract/ hinder the students’ learning process efficiency and also not provide space for effective learning. In visualization design, this is generally termed as ‘Chart Junk’ and even though it may be more suited towards industry demand it does not particularly perform well for students and pedagogical purposes.

III. METHODS

We begin with defining the Dijkstra's algorithm as the underlying basis of our research. For a given source node in the graph, the algorithm finds the shortest path between that node and every other.[4]:196–206 It can also be used for finding the shortest paths from a single node to a single destination node by stopping the algorithm once the shortest path to the destination node has been determined. For example, if the nodes of the graph represent cities and edge path costs represent driving distances between pairs of cities connected by a direct road, Dijkstra's algorithm can be used to find the shortest route between one city and all other cities. As a result, the shortest path algorithm is widely used in network routing protocols, most notably IS-IS (Intermediate System to Intermediate System) and Open Shortest Path First (OSPF) [5]

The application runs on a web hosted server and the page is split into two sections: The Narrative and The Interactive respectively. The first one which is the Narrative panel hosts the complete story that is being guided through the visualization to help the student adopt and learn the concepts conveyed in the process. The second one which is the Interactive panel hosts the visualization and the implementation table for the student to be able to visualize the changes owing to his answers through the buttons.

The graph layout is implemented through using the ForceLayout of the D3 library and allows moving/ dragging the complete graph with the help of the mouse. The initial graph display enables the student to have a complete look at the problem statement being displayed to him and the graph corresponding to the content in the narrative. These nodes can be observed by double clicking on them and are observed to hold directional properties. We also observe the initial table to be empty and consisting of now rows or values updated until the student starts selecting the options available to him. Once, the graph has been introduced and the problem statement been laid out, the student is given an array of questions to build the

complete implementation table by himself. On every question being iterated upon the ‘operational node’ or the ‘node of observation’ changes. This ‘operational node’ is reflective of the current implementation tables’ row value being worked on and thus highlights the connections from that node to adjacent first degree nodes and first degree links supplementing the user with enhanced visual cognition and knowledge of the choices being focussed upon.

For every verification, when any of the updates is incorrect the button turns red indicating to the student to proceed towards selecting another option which shall help proceed with further steps into the algorithm. If the updates from the student is correct, the implementation table gets another row updated and the ‘operational node’ changes to the next operated node with the lowest distance value. At this step, the application waits for the user’s next entry to evaluate further course of action by him/her.

The Force Directed layout implementation was particularly the best suited option for our case as it allowed simple constraints to be implemented easily. This implementation helps repulsive charge force, a pseudo-gravity force keeps nodes centered in the visible area and avoids expulsion of disconnected subgraphs, while links are fixed-distance geometric constraints. Additional custom forces and constraints were applied on the “tick” event, simply by updating the x and y attributes of nodes. The layout helped point out effectively the edge weights and helped create custom attributes like image addition, arrow manipulation, stroke width and node size of the graph being worked upon. The layout follows the method chaining pattern where setter methods return the layout itself, allowing multiple setters to be invoked in a concise statement. Unlike some of the other layout implementations which are stateless, the force layout keeps a reference to the associated nodes and links internally; thus, a given force layout instance can only be used with a single dataset which suited our case for tutoring the Dijkstra’s algorithm.

IV. RESULTS

This work takes into account various design components that led to a successful implementation of the concept being explored. Some of them were as follows:



1. The Story/ The Theme



Figure 4 showcases the storyline being set up to help user create a context around the concept. This particular storytelling technique has been famously used by universities and is known to help improve pedagogical understanding of the concept.



Jack's Family Needs Some Help



Now, let's work together to help Jack's family make their busy morning easier.

Jack is an elementary school student, and he lives with his Mom, Dad, and Grandpa. On a Tuesday morning, Jack needs to go to school as usual. His Dad needs to go to the office for an important meeting. His Mom is planning to go to the supermarket to buy some fresh vegetables to cook lunch. His Grandpa has a doctor appointment in the nearby hospital. They decide to leave the house together and share the ride. What is the fastest way for them to arrive at their destinations and how many cars do they need at minimum?


 Jack

 School


 Dad

 Office


 Mom

 Supermarket


 Grandpa

 Hospital

To solve this problem, we need to find the shortest paths between home and different destinations, and *Dijkstra's Algorithm* can help us find the answer. Let's use the directed edge-weighted graph at right as the map to solve this problem. *[Please play around with the graph. You are able to drag the nodes to different places]*.

Figure 4 - Setting up the Background Story.

2. The Graph Visualization Produced

Our primary dataset behind this particular graph was self-curated, and it catered to a wide range of corner cases as well. The graph was intended to have the following properties:

1. Directed Graph - Showcase possible directions of roads.
2. Draggable - Allow drag the graph's elements on the canvas.
3. Pictorial - Depicts pictures as laid out in the storyline.
4. Highlightable - Allows for highlighting of the node, its connected nodes, and the edges that are connecting the first degree connections for the node in operation.

The first three properties are depicted in Figure 5 which help the students have a better understanding of the problem statement.

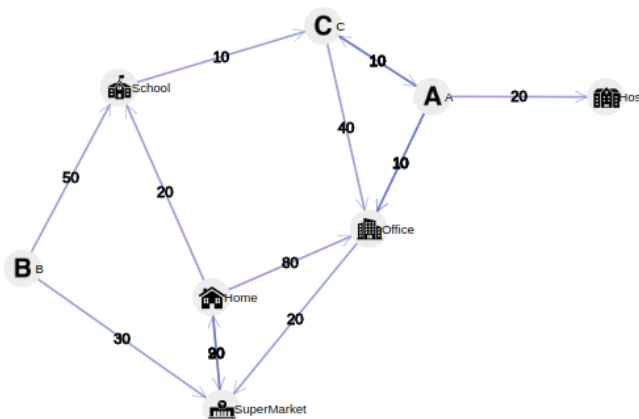


Figure 5 - The complete graph of operation.

As discussed above the visualization also features selective highlighting of the edges and nodes that form first degree connections to help the user make effective choices from the options he/she is presented with. The visualization also supports rendering of this effect on double-clicking the node. This effect can be visualized in Figure 6 below.

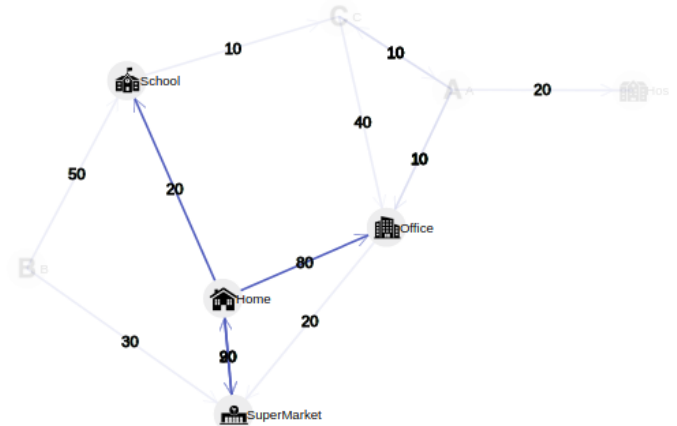


Figure 6 - Highlighting connections from 'Home'

3. Interactions and Teaching Concepts

This visual interaction through the questions and available 'button' options formed the core interaction medium with the graph and the implementation table. It exhibited the following properties:

1. Update of the implementation table on selection of every correct answer. Figure 7 below highlights the gradual answer based update of the implementation table.

Current Node	School	A	Office	B	C	Supermarket	Hospital
Home	20 (Home)	Infinity	80 (Home)	Infinity	Infinity	90 (Home)	Infinity

Figure 7 - Gradual update of the implementation table.

2. Red-Signalled warning on the button on selection of every wrong answer. This can be clearly observed in Figure 8. The user cannot proceed the steps without the correct answers in place.



Figure 8 - Interaction with Incorrect Prompt from User.

3. Highlighting on the graph based on the 'Operational Node' everytime we change the current node on a correct answer.
4. Start and End buttons for the user to know the end to end pipeline as depicted in Figure 9.



Figure 9 - Buttons that indicate the start and end of the process

4. The Data - Curation and Format

The dataset we curated comprised of 8 nodes and 11 edges. It comprises of custom ids, images, source, target and edge values contained within the json object. A snapshot of the data can be visibly observed in Figure 10.

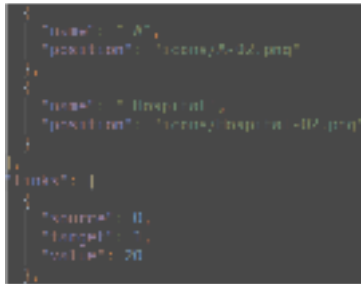


Figure 10 - The data attributes and use.

4. The Quiz - Validation of Users' Concepts

The quiz serves as the last section of the webpage and helps guide the user to validate his concepts. It also notifies the users of correct/ wrong answers based on 'Red' and 'Green' Flags as depicted in Figure 11.

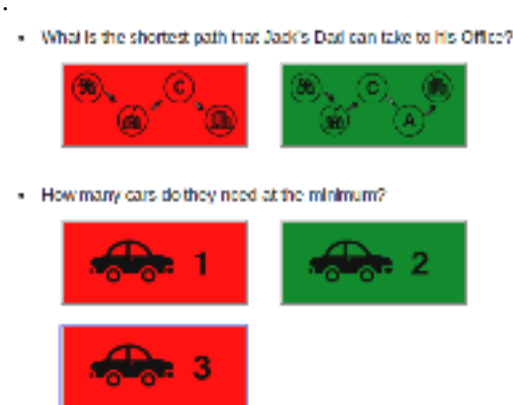


Figure 11 - The Quiz.

V. DISCUSSION

Some of the most challenging sub-problems observed as a part of this particular work involved charting out a story based interaction. Every piece of interaction should be guided with an effective change in the graph as well as in the implementation table. This involved framing a story that would very carefully cater to both the needs. It was essential that the story had multiple elements and multiple questions to be answered for the students to completely grasp the algorithm from multiple angles. Unless the story questions sub-concepts like - multiple common paths, un-visited edges, explainable explorations it would not have catered to the curiosity of the student. Efforts were taken to ensure that no edge cases were left out in defining the themed story of Jack and his Family. The story was made such that the student is forced to think about corner cases and make decisions where the outcome is not too easily predicted.

Particularly challenging design problems that we faced were to be able to navigate the users focus to the node being operated on. It was particularly challenging to implement the changing degree of opaqueness with the connected nodes and link to the 'node in action' for the particular step. Much of design and implementation considerations went into getting the feature in effect.

The audience (students) in particular gained step wise knowledge of the implementation of the Dijkstra's algorithm. They surely were also able to question themselves on every step, see every node in action, see the available options, brainstorm on the edge cases, work around unforeseeable options like that of undirected links and in the process be able to appreciate the efforts that went behind the construction of this algorithm and its visualization.

VI. FUTURE WORK

In conclusion, we present a novel way to improve the pedagogical understanding of the Dijkstra's algorithm through interactive visualization. This work caters to step by step evaluation and continuous monitoring of a student's understanding of the algorithm. It also helps the students reason about the implementation table population and discover the steps that lead to formation of the shortest paths from a particular point.

Some of the future work would definitely involve to have more control over the visualization. In our current implementation the data is fixed and the narrative is described by us. We would like to extend our work where the user has the charge to play around with the route length and add/delete nodes to further strengthen his comprehension of the subject being discovered and worked upon.

We would also like to conduct usability test to observe performance of users and gain feedback to seek any visual improvements and gauge their comprehension of the algorithm being taught to them. It would also help the user answer particular questions about the his interaction with the algorithm and help us uncover new perspectives in computer science pedagogy.

ACKNOWLEDGMENT

This research was supported by the iSchool, University of Washington. We thank our colleagues from course INFX 562 who provided insight and expertise that greatly assisted the research. We particularly thank Jessica Hullman, Course Instructor and Wang Chen, Course Teaching Assistant for their continuous support and guidance that greatly improved the project and this paper.

REFERENCES

- [1] Naps, T., G. Rößling, V. Almström, W. Dann, R. Fleischer, C. Hundhausen, A. Korhonen, L. Malmi, M. McNally, S. Rodger and J. A. Velazquez-Iturbide, Exploring the Role of Visualization and Engagement in Computer Science Education, SIGCSE Bull. 35(2) (2003), pp. 131–152
- [2] Torrubia, Torres-Blanc, López-Martínez, PathFinder: A Visualization eMathTeacher for Actively Learning Dijkstra's Algorithm, Electronic Notes in Theoretical Computer Science 224:151-158 (Jan 2009)
- [3] Makohon et al. Java Based Visualization And Animation For Teaching The Dijkstra Shortest Path Algorithm In Transportation Networks. International Journal of Software Engineering & Applications (IJSEA), Vol.7, No.3, (May 2016)
- [4] Borissova et al. E-learning Tool for Visualization of Shortest Paths Algorithms. Trends Journal of Sciences Research (2015) 2(3):84-89
- [5] Dijkstra's algorithm, From Wikipedia, the free encyclopedia:
https://en.wikipedia.org/wiki/Dijkstra%27s_algorithm