FIND THE WAY": A SERIOUS GAME FOR THE MINIMUM SPANNING TREE PROBLEM

I. Sarris[1], E. Papaioannou[2], N. Karanikolas[1], C. Kaklamanis[2]

[1]University of Patras (GREECE)
[2]University of Patras and CTI "Diophantus" (GREECE)

Abstract

"Find the Way" is a serious game we developed with the intention to help mainly university students but also all interested persons to understand the Minimum Spanning Tree (MST) problem and become familiar with the application of the algorithms suggested by R. Prim and J. Kruskal. Both algorithms are discussed as part of the curriculum in the context of courses on Introduction to Algorithms and are used in a wide range of practical real-world applications.

In the context of Graph Theory, a graph is a collection of vertices (i.e., points) and edges (i.e., lines connecting points). Vertices correspond to entities. An edge between two vertices implies that the corresponding entities are somehow related. Edges can bear weights which reflect some sort of cost required for their establishment. Weights can reflect physical or conceptual distance, time, budget and so on. Trees are special cases of graphs which are acyclic and connected. A Spanning Tree for a given graph is a tree containing all vertices of this graph. When weights are assigned to the edges of a graph, a Minimum Spanning Tree (MST) for this graph is a spanning tree of minimum total edge-cost. The MST problem aims at finding spanning trees of minimum total (edge) cost for given (connected) graphs and has many practical applications. Imagine for example that graph vertices correspond to houses, airports, close by islands, gates, and we seek for minimum-cost solutions for keeping all of them connected. Given a connected graph, we can efficiently compute a Minimum Spanning Tree for this graph using two well-known algorithms from the literature, one suggested initially by Vojtěch Jarník and later by Robert Prim and one suggested by Joseph Kruskal, which return similar results though in a slightly different fashion.

In "Find the Way", players can either create or are presented with a newly created village where pavements must be constructed so that all houses are connected with the lowest possible budget. Only the minimum-cost proposals are qualified for funding. Players can engineer their proposal about which houses must be connected with a pavement so that all houses are eventually connected at a minimum total budget using one of two available methods, Prim's algorithm or Kruskal's algorithm. According to the method suggested by Prim, players are given the house where the work will start and the cost for all individual pavements between any two houses in the village. Then, players build their proposal by gradually adding houses connected by minimum-cost pavements to houses already included in their solution so far until all houses are connected. According to the method suggested by Kruskal, players are given the cost for all individual pavements between any two houses in the village. Then, players build their proposal by gradually adding to their proposal minimum-cost pavements until all houses are connected. Player scores are calculated for each method taking into account elapsed time until the submission of their proposal, village size (i.e., number of vertices in the given graph) and possible pavements (i.e., number of edges in the given graph) and appear under player profiles.

"Find the Way" is a responsive application, currently supporting greek and english; it was developed using state-of-the art web technologies and software including html, css and javascript for the front-end as well as the node.js run time environment and the express.js framework for the back-end.

Keywords: Serious game, educational web app, responsive, Minimum Spanning Tree (MST) problem, Prim's algorithm, Kruskal's algorithm, node, js, express, js, html, css, javascript.

1 INTRODUCTION

Optimization problems, especially those related to networks and efficient connectivity, are fundamental in both theoretical studies and practical applications. Among these, the Minimum Spanning Tree (MST) problem [2] has a prominent place due to its widespread application across many fields such as logistics, telecommunications, electrical networks, and urban planning. The MST problem involves

finding the most cost-efficient way to connect a set of points or nodes, minimizing the total cost of the connections.

Despite its relevance and practical importance, the MST problem and its algorithmic solutions can often be considered as challenging and abstract by university students and novices in algorithmic theory. This complexity underscores the need for innovative educational methods to simplify and effectively communicate these concepts, making them accessible and engaging to learners.

Serious games [1], defined as games designed primarily for educational purposes rather than mere entertainment, have proven highly effective in facilitating active learning and deeper understanding of complex concepts. Properly designed serious games provide interactive and immersive experiences, making them ideal educational tools to address topics traditionally considered complex or inaccessible to beginners. Through interactive gameplay, serious games encourage critical thinking, problem-solving skills, and sustained engagement.

Some notable serious game examples include "DragonBox Algebra" [10], an intuitive and engaging approach to algebra, with a goal to excite young learners about math and maximize exposure to pre-algebraic concepts. Also "Foldit" [12] is a revolutionary crowdsourcing computer game enabling players to contribute to scientific research. It uses gamification to advance research in protein folding. "Lightbot" [13] is a puzzle game based on coding which teaches the players programming logic as they play. “Fair ‘n Square” [7] is a serious game for fair division algorithms focusing on “the bankruptcy problem” and “the problem of sealed bids”.

There are several web applications that deal with the MST problem. “Prim’s Algorithm” [6][22] and “Kruskal’s Algorithm” [5][20] are applications developed at “Technische Universität München” [16], where the MST problem is formulated as a game where the goal is to connect houses with cables at the lower possible cost. “Minimum Spanning Tree Calculator” [21] is a web application that lets users upload or draw a graph and then computes a Minimum Spanning Tree for this graph using Prim's or Kruskal's algorithm. The application returns the resulting MST visually so that users can see the optimized connections. The “GraphOnline” [18] tool lets users create graphs manually and then finds a Minimum Spanning Tree for these graphs using Prim’s algorithm. The application highlights the MST directly on the graph for easy visualization.

"Find the Way" is designed to bridge the gap between graph theory and games, providing learners with an interactive platform to visualize, explore, and practically apply MST algorithms. "Find the Way" is a serious game developed with the intention to help with understanding the Minimum Spanning Tree (MST) problem and familiarizing with the application of the algorithms suggested by R. Prim and J. Kruskal. Through gameplay that simulates a real-world scenario, students gain hands-on experience in algorithmic thinking and practical problem-solving, significantly enhancing their understanding and retention of the MST problem.

The rest of the paper is structured as follows. In Section 2, we provide background information and present practical applications of MST, including a comparative overview of Kruskal's and Prim's algorithms and the introduction of Erdős–Rényi random graphs. Section 3 outlines the detailed design, user interface, functionalities, software, and technologies used in the development of the "Find the Way" game. Finally, Section 4 concludes with a summary of the work conducted and presents future directions and potential enhancements for the game.

2 THE MINIMUM SPANNING TREE PROBLEM

In the context of Graph Theory a graph is a collection of vertices (i.e., points) and edges (i.e., lines connecting points). Vertices correspond to entities. An edge between two vertices implies that the corresponding entities are somehow related or associated. Usually edges are assigned weights which reflect some sort of cost required for their establishment. Weights can reflect physical or conceptual distance, time, budget and so on. Trees are special cases of graphs which are acyclic and connected. A Spanning Tree for a given graph is a tree containing all vertices of this graph. When weights are assigned to the edges of a graph, a Minimum Spanning Tree (MST) for this graph is a spanning tree of minimum total edge-cost for this graph. For example, for the weighted graph G depicted on the left of Fig. 1, the subgraph containing the edges marked in blue is a spanning tree of weight 26 for G, while the subgraph containing the edges marked in green is a spanning tree of minimum weight 18 for G

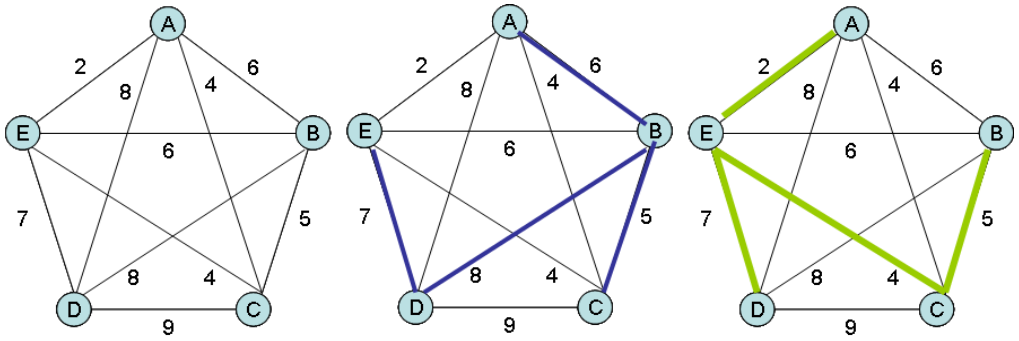

Figure 1. A edge-weighted graph G (left), a spanning tree for G (middle) and a minimum spanning tree for G (right).

The MST problem aims at finding spanning trees of minimum total (edge) cost for given (connected) graphs and has many practical applications. Imagine for example that graph vertices correspond to houses, airports, close-by islands, gates, and we seek for minimum-cost solutions for keeping all of them connected, i.e., for guaranteeing the existence of a (not necessarily direct) connection between any two of them at a minimum total cost. For instance, MST solutions enable effective planning and cost minimization for road construction, power grid layouts, water distribution networks, telecommunications, and logistical routing.

Given a connected graph, we can efficiently (i.e., fast) compute a Minimum Spanning Tree for this graph using two well-known algorithms from the literature: Prim's algorithm and Kruskal's algorithm, which return similar results though in a slightly different fashion, Prim's algorithm starts from a particular vertex and computes a Minimum Spanning Tree by gradually augmenting a partial spanning tree by adding vertices at minimum additional cost. Kruskal's algorithm computes a Minimum Spanning Tree by gradually adding minimum-cost edges to in principle disconnected components/subtrees until all are connected.

2.1 Prim' s algorithm

Prim’s algorithm [6], originally suggested by Vojtěch Jarník in 1930 [4] and then independently by Robert Prim in 1957, constructs the MST incrementally starting from a single vertex. At each step, Prim’s algorithm adds the lowest-weight edge connecting the current MST to a new vertex, continuously expanding the MST until all vertices are included. The input of Prim’s algorithm is a connected graph with weighted edges and a selected starting vertex. The output is a set of edges forming the MST. Prim's algorithm evolves as follows. Starting from an arbitrary (or indicated) vertex, an MST under-construction is initiated. Then, all edges connecting the vertices in the current MST to vertices outside the MST are identified. The edge with the minimal weight is added to the MST. This process is repeated until all vertices of the given graph are included in the MST. Prim's algorithm typically employs a priority queue, leading to a computational complexity of O(|E| log |V|), where V is the set of vertices and E is the set of edges of the initial graph. Prim’ algorithm performs efficiently on dense graphs with numerous edges.

As an example, we are given the graph depicted on the left of Fig. 2 and we are requested to compute an MST for this graph by applying the Prim’s algorithm starting at vertex A. First, vertex A is included in the MST under construction. Then, from the 2 edges incident to A, AB of a weight of 2 and AC of a weight of 1, the one of minimum weight (ties are broken arbitrarily), i.e., AC, is added to the MST. Next, from the not yet selected edges incident to vertices A and C, which are currently included in the MST, namely AB of a weight of 2, CB of a weight of 3 and CD of a weight of 4, the one of minimum weight, i.e., AB, is added to the MST. Next, from the not yet selected edges incident to vertices A, B and C, which are currently included in the MST, namely BD of a weight of 1 and CD of a weight of 4, the one of minimum weight, i.e., BD, is added to the MST; BC is ignored because its addition would cause a cycle (ABC) to form. Since all vertices of the graph have been included in the MST the algorithm terminates returning an MST of total weight of 4 for the given graph. The procedure described above is depicted in Fig. 2.

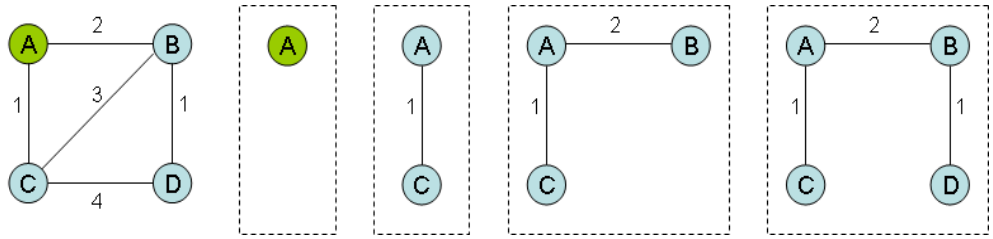

Figure 2. Prim's algorithm: example.

2.2 Kruskal' s algorithm

Introduced by Joseph Kruskal in 1956, Kruskal's algorithm [5] solves the MST problem by systematically selecting edges based on ascending weight values. The algorithm starts with an empty edge set, progressively adding the smallest available edges that do not create cycles until all vertices are connected. The input of this algorithm is a connected graph with weighted edges. The output is a set of edges forming an MST for the given graph. Kruskal's algorithm evolves as follows. First all edges are sorted in ascending order according to their weights. Then, an empty edge-set is initialized for the MST under-construction. Iterating through the sorted edges, each edge is either added to the MST set or discarded if its addition would create a cycle. This process continues until all graph vertices are included in the MST. Kruskal's algorithm has a computational complexity of O(|E| log |E|), where E is the set of edges of the initial graph. It performs optimally in sparse graphs where the number of edges is relatively low.

As an example, we are given again the graph of Fig. 2 with 4 vertices labeled A, B, C, D, and 5 edges which are AB (of a weight of 2), AC (of a weight of 1), BC (of a weight of 3), BD (of a weight of 1), CD (of a weight of 4). We are now requested to compute an MST for this graph by applying the Kruskal's algorithm. We first sort the edges of the graph according to their weight in ascending order, obtain the sorted list AC, BD, AB, BC, CD. Now, starting from the top of this list, we examine each edge and include it to the MST as long as it does not cause a cycle to form until all vertices of the graph are included in the MST. In particular, adding the edges AC, BD and AB results in the formation of an MST of total weight of 4 for the given graph. The procedure described above is depicted in Fig. 3.

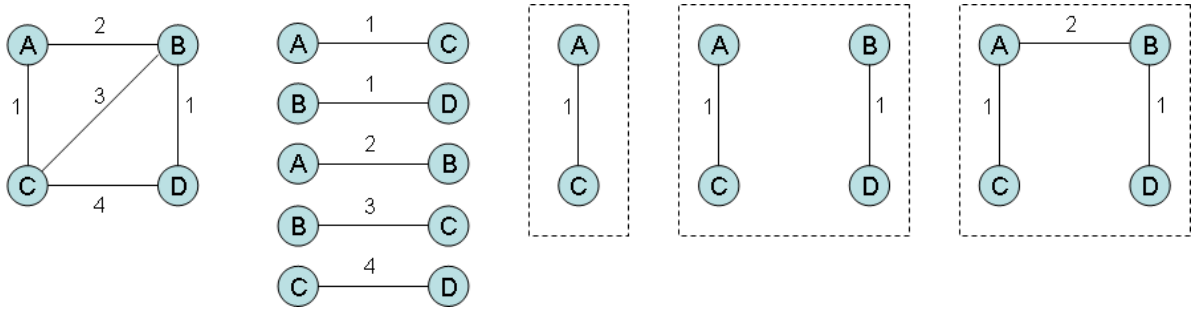

Figure 3. Kruskal's algorithm: example.

Note that the same MST is computed by both Kruskal's and Prim's algorithms for the simple graph in our example. However, the MST is constructed in a slightly different fashion by each of the two algorithms, with one prevailing detail being that the final MST is computed by Kruskal's algorithm via joining initially disconnected components while the final MST is computed by Prim's algorithm via gradually augmenting a single connected component.

2.3 Erdős–Rényi random graphs

Erdős–Rényi random graphs [3], introduced by mathematicians Paul Erdős and Alfréd Rényi in 1959, constitute a fundamental model in random graph theory for the generation of random graphs. In the relevant literature, the definition of the Erdős-Rényi model can be found in two variants, G(n,p) or G(n,m). According to the G(n,p) variant (Fig.4, left), for generating a random graph, we start with n vertices and generate an edge between each pair of vertices with a probability p, independently of all other pairs. According to the G(n,m) variant (Fig. 4, right), which is suitable for generating graphs with a particular number of edges, for generating a random graph, we start with n vertices and randomly select m from all possible edges to connect the vertices.

The Erdős–Rényi model is extensively used to simulate and analyze the properties of random networks affecting algorithmic performance, network robustness, or network connectivity which can be tested against real-world scenaria in the context of communication, social or even biological networks. In the context of our "Find the Way" game, we exploit a combination of the two variants of Erdős–Rényi random graphs in order to provide a versatile and realistic approach to generating diverse problem scenarios. They simulate real-world uncertainties and complexities, enabling players to test and enhance their understanding of MST algorithms under varying conditions. By exposing players to randomly generated graphs, the game effectively illustrates algorithm behavior across multiple practical scenarios, significantly reinforcing educational outcomes. Overall, the inclusion of Erdős–Rényi random graphs in the game design expands the educational scope, allowing users to gain insight into algorithmic efficiency and effectiveness within realistic and unpredictable environments.

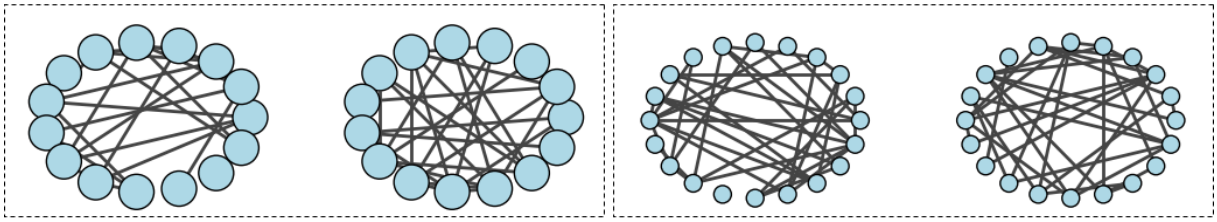

Figure 4. Erdős–Rényi random graphs: G(n,p) for n=15, p=0.2 (left) and G(n.m) for n=20, m=35 (right) [17].

3 OUR “FIND THE WAY” GAME

“Find the Way” is an educational serious game developed with the intention to help mainly university students but also all interested persons to understand the Minimum Spanning Tree (MST) problem and familiarize with the application of the algorithms suggested by R. Prim and J. Kruskal. Both algorithms are discussed in the context of courses on Introduction to Algorithms as typical part of their curriculum and are used in a wide range of practical real-world applications.

The “Find the Way” game is currently available in greek and english and can be freely accessed online at https://findtheway-app-553a38324d72.herokuapp.com. The scenario implemented in the context of the game is as follows. Players are presented with a newly created village (Fig. 5) where pavements must be created so that all houses are connected with the lower possible budget. Only the minimum-cost proposals qualify for funding. In order to help the residents to obtain support for improving their village, players can engineer their proposal about which houses must be connected with a pavement so that all houses are connected at a minimum total budget using one of two available methods: one suggested initially by Vojtěch Jarník and later by Robert Prim and one suggested by Joseph Kruskal.

According to the method suggested initially by Vojtěch Jarník and later by Robert Prim, players are given the house where the work will start and the cost for all individual pavements between any two houses in the village (Fig. 5, left). Then, players build their proposal by gradually adding houses connected by minimum-cost pavements to houses they have already included in their solution so far until all houses are connected. According to the method suggested by Joseph Kruskal, players are given the cost for all individual pavements between any two houses in the village (Fig. 5, right). Then, players build their proposal by gradually adding to their proposal minimum-cost pavements until all houses are connected.

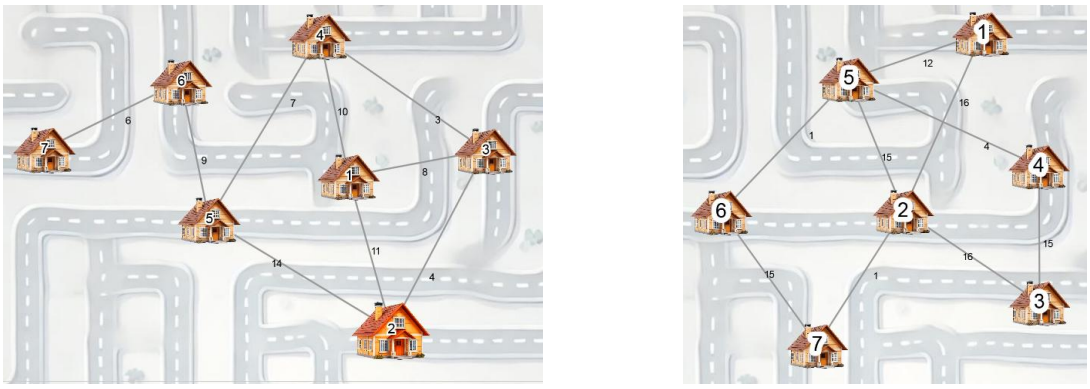

Figure 5. Initial instances of the game for Prim’s (left) and Kruskal’s (right) algorithms.

3.1 Design

Our “Find the Way” game follows a clear and rather intuitive design. Houses and pavements have been used as graphical representations for vertices and edges, respectively. Visualizing instances of the MST problem makes it easier to understand even by learners unfamiliar with graph theory concepts. It provides real-time feedback, while the general styling idea makes it a modern and pleasant game, able to target not only students. The application is fully responsive (Fig. 6), meaning that the players can play through any device and screen dimension, like computer, tablet and smartphone.

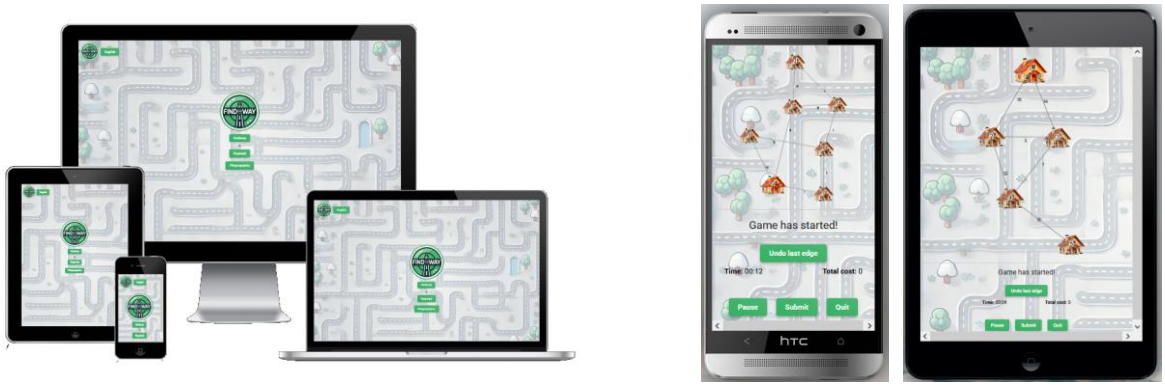

Figure 6. “Find the Way” on various devices.

3.1.1 User Interface

Our game features a simple and easy-to-follow user interface. As depicted in the left part of Fig. 7, the start screen of the game is composed of a header and its main body. The header, which remains visible on all screens of the game, contains the logo of the game which also serves as a home button and a language selector. Currently, our game supports the greek and english languages. The main body of the start screen contains the logo of the game, and 3 buttons: Signup and Login as well as an Information button through which a short description of the game is provided. For signing up, users must give a username, a nickname and a password. Then, users can use their username and password as credentials for signing in (Fig. 7, middle). After logging in, players are presented with a menu containing 3 items depicted on the right part of Fig. 7: a box entitled "Pick an algorithm" with 2 options in the form of buttons labeled “Prim's” and “Kruskal's”, a button labeled “Scores” and a logout button.

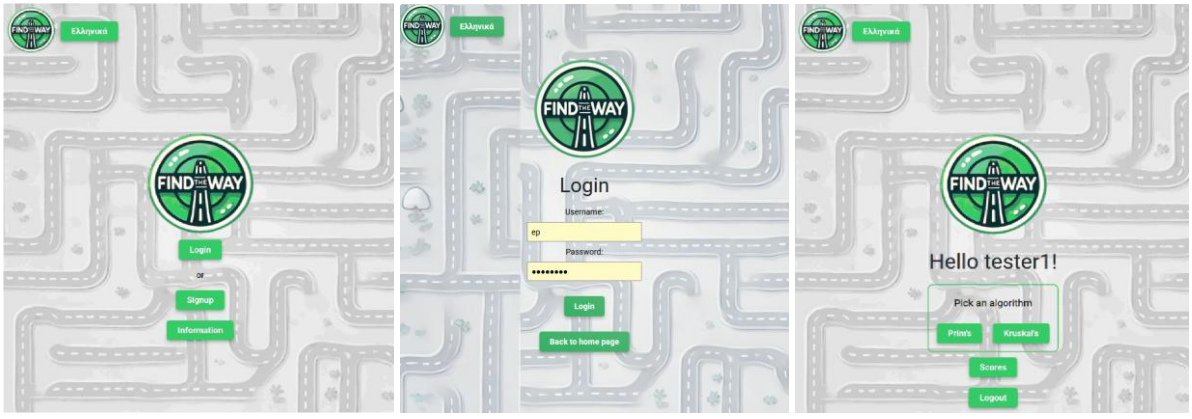

Figure 7. Starting the game.

Using the "Scores" button, players are presented with a new screen which contains top 10 scores for each algorithm in descending order, with nickname and total score for each top-player, summary of their total individual scores for each algorithm and a back button (Fig. 8, left). "Prim's" and "Kruskal's buttons serve as selectors for the player's preferred algorithm and lead to new screens of similar setup (Fig. 8, middle). The screen for each algorithm contains a menu with 3 parts: upper, middle and lower.

The lower part of the menu contains a "Back" button. The middle part of the menu provides 3 options for the difficulty level, Beginner, Intermediate and Expert, and a “Create custom level” option.

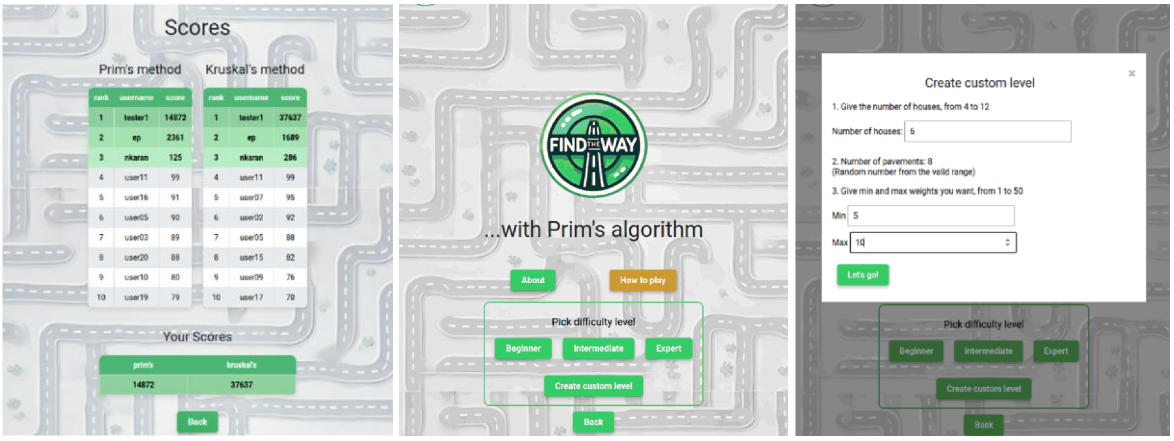

Figure 8. Overviewing scores and selecting a game level or setting up an instance.

The "Create custom level" button enables players to setup their own instance, i.e., players choose the number of houses for the village and min/max values for the pavement cost which can range from 1 to 50 (Fig. 8, right). The upper part of the menu contains the buttons "About" and "How to play" (Fig. 9). The "About" button enables players to view comprehensive information about the MST problem and the algorithms suggested by Prim and Kruskal, as well as a short description of the game and its objectives. The “How to play” button provides playing instructions.

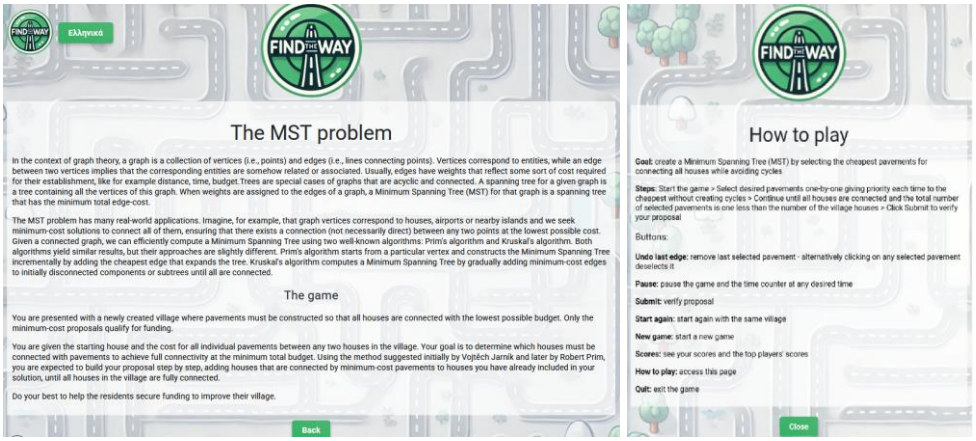

Figure 9. “About” and “How to play” buttons.

Reaching the main screen of the game (Fig. 10, left), users are presented with an instance of the village, containing houses and pavements together with their cost. On this interactive image component, players can select pavements by clicking with on the corresponding edges in order to include them in the MST under construction. Below the village component, an "Undo last edge" button enables players to unselect the most recently selected pavement, there is a timer and a "Total weight" counter for the MST under construction, 3 control buttons, namely Pause, Submit and Quit. Pause and Quit buttons allow players to temporarily stop or leave the game, respectively. Using the Submit button, players are presented with a new screen (Fig. 10, right) where they receive an evaluation of their solution, a comparison with an indicative correct solution and options for starting over or proceeding with a new instance via corresponding buttons labelled respectively "Start again" and "New game". Regarding the evaluation of submitted solutions, with an intent to enhance the learning experience, colours have been exploited as a means to visualize whether pavements (i.e., edges) have correctly been included in the solution. In particular, green indicates correct edge-choices while red indicates errors. In addition, players can have access to instructions via the "How to play" button and can also view the scores screen via the "Scores" button.

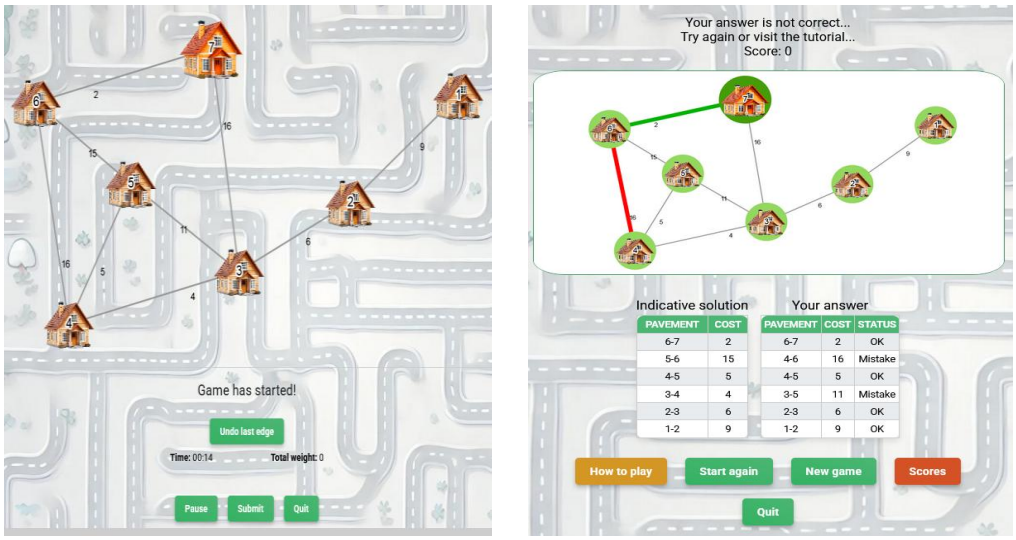

Figure 10. Solution submission and evaluation.

3.1.2 Functionalities

"Find the way" is addressed to registered players. Thus, functionalities for signing up and signing in are supported. Players must first create their account and then log in with their credentials, so that the scores can be saved and compared to scores of other players. For signing up, players must provide a nickname, username and password. The password is hashed, so that nobody can see their password in the database, in case someone uses a personal password in many cases.

Available functionalities also include the selection of game mode and game level. In particular, after connecting to the game, players must select the game mode, i.e., which of the two available algorithms, Prim's and Kruskal's, they wish to play with. Then, players must select a difficulty level out of three available options: beginner, intermediate and expert. After a game mode and difficulty level is picked, the game begins and players build their MST solutions by progressively selecting pavements that connect houses at minimal total cost. To this aim, a functionality for edge selection is provided. After submitting a solution, an evaluation functionality has been implemented which rewards players with a score using a custom point system that takes into account time spent for the MST computation and the complexity of the initial instance in terms of total number of houses and pavements given. Player performance is tracked and scored based on accuracy, efficiency, and the complexity of the provided graphs. Scores are displayed on player profiles, motivating continuous learning and improvement.

Player decisions are dynamically validated, and the game accommodates multiple correct MST solutions, when edge-weight ties exist. From a technical point of view, the related functionality works as follows. When a solution is submitted, a script runs having the original graph as an input. According to the method/algorithm the player is training on, all the possible MST solutions are computed and compared with the solution submitted by the player. If a match is found, the solution is correct and the points are granted.

3.2 Development

"Find the Way" was developed using state-of-the-art web technologies to ensure broad accessibility, responsiveness, and ease of maintenance. We have used HTML (HyperText Markup Language) [23] as the markup language, which provides the structural backbone for web pages and interfaces, facilitating clear content organization and readability. For styling the pages we used CSS (Cascading Style Sheets) [15], which is used extensively for styling the game's interface, ensuring aesthetic appeal and responsive design across various devices. Functionality comes with JavaScript [19], which powers interactive features, real-time feedback, and dynamic game logic. In the main game pages we used Cytoscape.js (https://js.cytoscape.org) [9], an open-source JavaScript library specialized in graph visualization. This technology was integral to visually representing interactive MST problems. The back-end is handled by Node.js [14], an efficient runtime environment that handles server-side operations, user authentication, and data processing in real-time. Also Express.js [11], which is a minimalist web application framework used with Node.js to manage routing, request handling, and

server-side application logic effectively. Additionally, the application is deployed from Heroku (https://www.heroku.com/) and also uses a Postgres Database which is hosted in Neon (https://neon.tech/).

4 CONCLUSIONS AND FUTURE PLANS

"Find the Way" is a serious game we developed with the primary objective of providing an effective educational tool for learning and understanding the Minimum Spanning Tree (MST) problem and its associated algorithms, specifically Kruskal's and Prim's algorithms. Our intention was to bridge the gap between theoretical algorithm concepts and practical application, facilitating an engaging, interactive, and intuitive learning experience for university students and other interested learners. Through careful design, user interface considerations, and robust technological implementation, the game has successfully achieved its primary educational objectives. Players interactively engage with realistic problem scenarios, allowing them to visually and practically understand the algorithmic procedures and complexities involved in constructing optimal MST solutions. Dynamic feedback and interactive gameplay have significantly enhanced user understanding, algorithmic proficiency, and sustained interest in graph theory concepts.

Future directions and improvements of the "Find the Way" application include potential enhancements to broaden its educational impact and usability. One of them is expanded translation. More languages available would make the game attractive to a wider audience. The algorithm coverage can also be expanded, introducing additional fundamental graph algorithms, such as Dijkstra's shortest-path algorithm, Bellman-Ford algorithm, and network flow algorithms, enriching the learning scope. Also, by developing advanced analytical and visual feedback tools we could help users evaluate their experience and provide information and ideas that could lead to significant updates. Additionally, multiplayer functionality could be considered, incorporating competitive and collaborative multiplayer modes, allowing learners to engage with peers, promoting teamwork, and facilitating collaborative problem-solving experiences. Also, conducting structured user surveys to systematically evaluate the game's effectiveness in educational settings, could lead to gathering feedback, able to inform future developments and improvements.

REFERENCES

[1] D. Chandross and E. DeCourcy, "Serious Educational Games in Education for Online Learning," *International Journal on Innovations in Online Education*, vol. 2, no. 3, pp. 1-27, 2019.

[2] T. H. Cormen, C. E. Leiserson, R. L. Rivest and C. Stein, *Introduction to Algorithms*, *3rd ed.* Cambridge, MA: MIT Press, 2009.

[3] P. Erdős and A. Rényi, “On random graphs,” *Publicationes Mathematicae*, vol. 6, pp. 290–297, 1959.

[4] V. Jarník, “O jistém problému minimálním [About a certain minimal problem],” *Práce Moravské Přírodovědecké Společnosti*, vol. 6, pp. 57–63, 1930.

[5] J. B. Kruskal, “On the shortest spanning subtree of a graph and the traveling salesman problem,” *Proceedings of the American Mathematical Society*, vol. 7, no. 1, pp. 48–50, 1956.

[6] R. C. Prim, “Shortest connection networks and some generalizations,” *Bell System Technical Journal*, vol. 36, no. 6, pp. 1389–1401, 1957.

[7] P. Stavropoulos, E. Papaioannou, C. Kaklamanis and D. Tsolis, “Fair ‘n Square: A serious game for fair division,” *Proceedings of the 16th Annual Int. Conf. Education, Research and Innovation (ICERI 23)*, pp. 1190–1199, 2023.

[8] M. Zyda, “From visual simulation to virtual reality to games,” *Computer*, vol. 38, no. 9, pp. 25–32, 2005.

[9] Cytoscape Consortium. (2023). Cytoscape.js. Retrieved from https://js.cytoscape.org

[10] DragonBox Algebra. (2023). Retrieved from https://dragonbox.com

[11] Express.js Framework. (2023). Express. Retrieved from https://expressjs.com

[12] Foldit. (2023). Retrieved from https://fold.it

[13] Lightbot. (2023). Retrieved from https://lightbot.com

[14] Node.js Foundation. (2023). Node.js. Retrieved from https://nodejs.org

[15] CSS: Cascading Style Sheets. Retrieved from https://developer.mozilla.org/en-US/docs/Web/CSS

[16] Discrete Optimization, Department of Mathematics, Technische Universität München. Retrieved from https://www.math.cit.tum.de/math/forschung/gruppen/discrete-optimization

[17] Erdős-Rényi Graph. Retrieved from https://python.igraph.org/en/main/tutorials/erdos_renyi.html

[18] Graph Online. Retrieved from https://graphonline.ru/en/wiki/Help/FindMinimumSpanningTree

[19] JavaScript. Retrieved from https://developer.mozilla.org/en-US/docs/Web/JavaScript

[20] Kruskal's Algorithm. Retrieved from https://algorithms.discrete.ma.tum.de/graph-algorithms/mst-kruskal/index_en.html

[21] Minimum Spanning Tree Calculator. Retrieved from https://mst-calculator.vercel.app/

[22] Prim's Algorithm. Retrieved from https://algorithms.discrete.ma.tum.de/graph-algorithms/mst-prim/index_en.html

[23] Structuring content with HTML. Retrieved from https://developer.mozilla.org/en-US/docs/Learn_web_development/Core/Structuring_content