JOHANNES KEPLER
UNIVERSITY LINZ

1. Graph Theory

This section provides an introduction to the key concepts of graph theory. It is intended for teacher to understand the mathematical fundamental principles underlaying the Dijkstra algorithm. Graph theory is part of discrete mathematics. For a detailed description, books about graph theory as [19], [**bhargava2024algorithmen**], [**dorfler2011graphentheorie**], [20], [21], or [**trudeau2013introduction**] are recommended. An introduction in the field of set theory is presented in the teachers education book of **schichl2009einfuhrung**[**schichl2009einfuhrung**] In general graph theory describes the relationship between objects. In a more mathematical definition, a graph consists of nodes which are connected by edges. Graphs are mathematical models and have many useful applications in the modern world. It is important to notice that they have nothing in common with graphs of functions. Graph theory is an effective tool having applications in a variety of domains, including computer science, biology, sociology, and transportation.

- Network analysis: modeling and optimization of transportation networks (road, rail, air travel) [**peranginangin2024analysis**]
- Electronics: design of digital circuits [**dorfler2018electrical**]
- Biology: analyze biological networks [**pavlopoulos2011using**]
- AI: graph neural networks (GNN) [**li2023graph**] and explainable AI through graph theory [**ucer2022explainable**]
- Sports analytics: analyze player interactions, and strategies [**sports**]
- Automata theory: state diagrams [**dutta2022graphs**]

1.1. Historical Remarks

The Seven Bridge Problem goes back to the year 1736. It is also called 'Königsberger Brücken Problem'. The problem statement is to find a tour crossing every bridge exactly once.

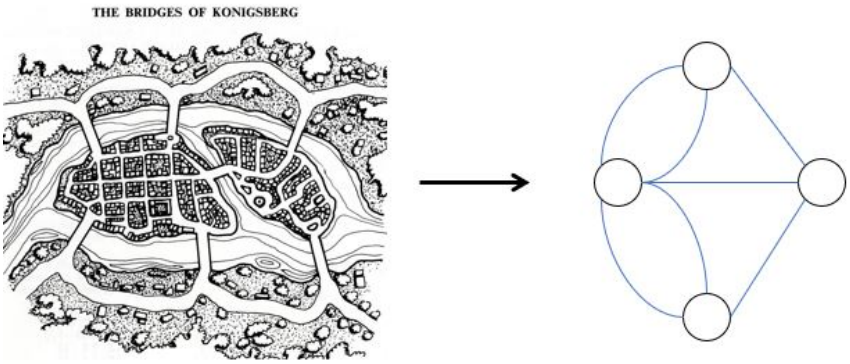

Figure 1.1.: Königsberg Bridge Problem, taken from [**kbp**]

The medieval city Königsberg is now Kaliningrad. The Pregel river goes through this town, which has two large islands (named Lomse and Kneiphof) in the middle. These islands are surrounded by the river. Figure **??** illustrates the seven bridges that connect the islands. Karl Leonhard Gottlieb Ehler came up with the question, which route has one to take to cross all seven bridges without crossing any of them more than once? Karl struggled with this problem and decided to write the famous mathematician Leonard Euler for help. Euler simplified the problem in a graphical representation of the four landmasses by four nodes named A,B,C and D, and each bridge as an edge between the nodes. He demonstrated in his article **euler1741solutio** that, if each node has an even number of edges a tour crossing every bridge exactly once can be found. With his article[**euler1741solutio**] Leonard Euler invented a new field in mathematics, the so called "Geometry of Position", now known as graph theory.

1.2. Mathematical Graph Theory Basic Definitions

The definitions and information in this section are taken from the books [**dorfler2011graphentheorie**] and [20].

1.2.1. Graph

A **graph** G is an ordered pair (V, E), where:

- V is a set of **vertices** (also called nodes).
- E is a set of **edges**, which are unordered pairs of vertices.

Unordered pairs are written in curly brackets. The number of nodes is called **order** of the graph. The number of edges is called **size** of the graph. Two nodes are **adjacent** if they are connected by an edge. In other words, if the two nodes are neighbors.

Mathematically, $G = (V, E)$, where $V = \{v_1, v_2, \ldots, v_n\}$ and $E = \{\{v_i, v_j\} \mid v_i, v_j \in V\}$.

The edges of a graph represent the relationship between the nodes. Nodes can be labeled by letters or numbers to distinguish them from each other. An example of a graph G is given by the sets:

- $V = \{1, 2, 3, 4\}$
- $E = \{\{1, 2\}, \{1, 3\}, \{2, 3\}, \{3, 4\}\}$

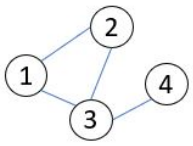

Figure 1.2.: Example of an undirected graph

1.2.2. Directed Graph

A **directed graph** or **digraph** G is an ordered pair (V, E), where:

- V is a set of vertices.
- E is a set of ordered pairs of vertices, called **directed edges**.

Mathematically, $G = (V, E)$, where $V = \{v_1, v_2, \ldots, v_n\}$ and $E = \{(v_i, v_j) \mid v_i, v_j \in V\}$.

Consider the following directed graph G with:

- $V = \{1, 2, 3, 4, 5, 6\}$
- $E = \{(1,2), (1,3), (2,5), (3,4), (3,6), (4,1)\}$

Ordered pairs are written in round brackets and are also called tuples. The graphical representation of the digraph is shown in figure **??**. It has 6 nodes and 6 edges.

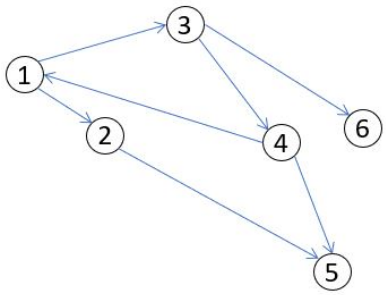

Figure 1.3.: Example of a directed graph

1.2.3. Weighted Graph

A **weighted graph** is a graph $G = (V, E)$ where each edge $\{v_i, v_j\}$ has an associated weight w_{ij}. Weighted graphs can be directed or undirected.
A street map can be visualized as a weighted graph, with weights representing the distance in kilometers. Another example is a railway network in which the weights between stations indicate the duration between them.

Mathematically, $G = (V, E, w)$, where $w : E \to \mathbb{R}$.

The representation of an edge is expanded from a pair to triplets to indicate the weight in the 3rd position.

Consider a weighted graph G with:

- $V = \{a, b, c, d\}$
- $E = \{\{a, b, 5\}, \{a, c, 3\}, \{b, c, 1\}, \{b, d, 4\}\}$

Here, nodes are labeled with letters. The edges have weights 5, 3, 1 and 4 respectively.

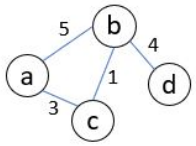

Figure 1.4.: Example of an undirected weighted graph

1.2.4. Walk

A **walk** in a graph is a finite sequence of nodes: $v_1, v_2, v_3, \ldots, v_k$ in which v_1 is connected by an edge to v_2, v_2 is connected by an edge to v_3 and so on til v_k. If the first and last node are the same it is a **closed walk**, otherwise it's **open**.

A **trail** is a walk with no repeated edges, though nodes may be repeated. Closed trails are also known as **circuits**. If all nodes in the sequence are distinct, a walk is called **path** (a walk with no repeated nodes and no repeated edges).

In figure **??** an example of a trail is the node sequence: b, a, c, b, d and an example of a path is the node sequence: a, c, b, d.

A graph is called **connected** if a walk from any node to any other node via edges exists.

1.2.5. Cycle

A **cycle** is a path that starts and ends at the same node with no repeated edges or nodes (except the starting and ending node). In figure **??** the directed graph contains a cycle of size 3 given by the following edge sequence: $(1,3),(3,4),(4,1)$.

1.3. Common Graph Characteristics and Terms

1.3.1. Degree of a Vertex

The **degree** of a node v in an undirected graph, denoted $\deg(v)$, is the number of edges attached to v. The degree of node 3 in figure **??** is $\deg(3) = 3$. In a directed graph, the degree is distinguished between:

- In-degree $deg^-(v)$: number of edges entering v
- Out-degree $deg^+(v)$: number of edges leaving v

In figure **??** the $deg^-(3) = 1$ and $deg^+(3) = 2$.

1.3.2. Complete Graph

A **complete graph** K_n is an undirected graph with n nodes where every pair of distinct nodes is connected by a unique edge. Figure **??** show the complete graph K_4.

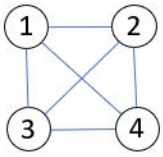

Figure 1.5.: Example complete graph with 4 nodes

1.3.3. Subgraph

A graph (V_1, E_1) is a **subgraph** of (V_2, E_2) if it is a graph where $(V_1 \subseteq V_2)$ and $(E_1 \subset E_2)$ The graph of figure **??** is a subgraph of K_4, shown in **??**.

1.3.4. Planar Graph

A connected graph is said to be **planar** if it can be drawn without any edges crossing. When a planar graph is drawn, it divides the plane into regions. The complete graph K_4 can be drawn without crossing edges, as shown in figure **??** . It divides the plane into 4 regions, numbered with roman numerals.

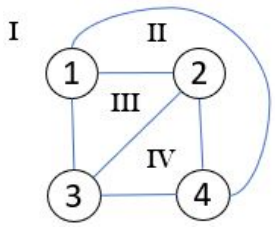

Figure 1.6.: Example complete planar graph with 4 nodes, divided into 4 regions

Definition 1 (Euler characteristic)**.** *For any connected planar graph the* ***Euler characteristic****:* $V - E + R = 2$*, respectively V number of nodes, E number of edges, R number of regions.*

1.4. Graph Representations

1.4.1. Visual Representation

The most intuitive representation is the visual one. Nodes are visualized by circles or dots and edges are lines between two nodes.

1.4.2. Adjacency List

An adjacency list is a collection of unordered lists used to represent a graph. Each list describes the set of neighbors of a vertex in the graph.

The adjacency list representation of the graph of figure **??** is:

- 1: 2, 3
- 2: 1, 3
- 3: 1, 2, 4
- 4: 3

1.4.3. Adjacency Matrix

The **adjacency matrix** A of a graph G with n vertices is an $n \times n$ matrix where the entry a_{ij} is 1 if there exists an edge from vertex i to vertex j, otherwise it is set to 0.

For a simple graph:

$$a_{ij} = \begin{cases} 1 & \text{if there is an edge between } v_i \text{ and } v_j, \\ 0 & \text{otherwise.} \end{cases}$$

In a weighted graph the entry a_{ij} is set to the weight of the edge from vertex i to vertex j. The adjacency matrix of the undirected graph in figure **??** is given below:

$$A = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Both adjacency lists and adjacency matrices are useful representations of graphs, most commonly used in programming languages. Adjacency lists are generally more space-efficient for graphs with less edges, while adjacency matrices allow for faster lookup times to determine if an edge exists between two nodes.

Understanding the basic definitions and properties of graphs is essential for exploring more advanced topics. Also, teachers should be aware of the mathematical foundation behind graph theory.

In the field of graph theory many exciting problems exists:

- Enumeration
- Subgraphs
- Planar Graphs
- Graph Coloring
- Route Problems
- Seven bridges of Königsberg
- Decomposition Problems
- Traveling Salesman
- Cycle Detection

The focus on the presented learning sequence is based on route problems.

1.5. Shortest Path Problem

An interesting problem for students in secondary school is to find the shortest path on a map. An algorithm to find the shortest path from a starting node to a destination node in a graph is described in the paper of Dijkstra[15].

The algorithm is shown below.

Algorithm 1 Dijkstra shortest path algorithm

Require: $G = (V, E), s \in V$

Ensure: $P = V$

1: **for** each $v \in V$ **do**
2: $dist(v) \leftarrow \infty$
3: $prev(v) \leftarrow null$
4: **end for**
5: $dist(s) = 0$
6: **repeat**
7: $c \leftarrow arg\min_{v \in V \setminus P} dist(v)$
8: $P = P \cap c$
9: **for** each node $u \in V \setminus P$ adjacent to c **do**
10: **if** $dist(u) > d(c) + w(c, u)$ **then**
11: $dist(u) = d(c) + w(c, u)$
12: $prev(u) = c$
13: **end if**
14: **end for**
15: **until** $P = \emptyset$
16: **return** $dist, prev$

Explanation of the key steps:

- Initialization (line 1 to 5): Set distance of all nodes to infinity, except starting node gets distance zero.
- Main Loop (line 6 to 15): While there are unvisited nodes, select the node with smallest known distance as current node.
- Check neighbors (line 9 to 14): For each neighbor of current node, calculate new distance. If it's less than current known distance, update distance and previous node is set to current node.

Bibliography

[1] Anton Reiter and Christian Berger. “20 Jahre Schulinformatik in Österreich und IKT-Einsatz im Unterricht”. In: *CDA Verlag* (2005) (cit. on p. 1).

[2] Cynthia Luna Scott. “The futures of learning 2: What kind of learning for the 21st century”. In: (2015) (cit. on p. 1).

[3] Barbara Sabitzer, Peter K Antonitsch, and Stefan Pasterk. “Informatics concepts for primary education: preparing children for computational thinking”. In: *Proceedings of the 9th Workshop in Primary and Secondary Computing Education*. 2014, pp. 108–111 (cit. on p. 1).

[4] Seymour Papert. *Children, computers, and powerful ideas*. Harvester, 1980 (cit. on p. 2).

[5] Jeannette M Wing. “Computational thinking”. In: *Communications of the ACM* 49.3 (2006), pp. 33–35 (cit. on p. 2).

[6] Valerie J Shute, Chen Sun, and Jodi Asbell-Clarke. “Demystifying computational thinking”. In: *Educational research review* 22 (2017), pp. 142–158 (cit. on p. 2).

[7] BBC Bitesize. *Introduction to computational thinking*. 2018. URL: `https://www.bbc.co.uk/bitesize/guides/zp92mp3/revision/1` (visited on 02/02/2022) (cit. on p. 2).

[8] OpenStax. *Computational Thinking*. 2024. URL: `https://openstax.org/books/introduction-computer-science/pages/2-1-computational-thinking` (visited on 02/02/2025) (cit. on p. 2).

[9] Stefania Bocconi et al. “Reviewing computational thinking in compulsory education: State of play and practices from computing education”. In: (2022) (cit. on pp. 2, 19).

[10] Franziska Mayrleitner. “COMPUTATIONAL THINKING ALS LERNSTRATEGIE IM MATHEMATIKUNTERRICHT”. M.S. Thesis. Johannes Kepler University, 2020 (cit. on p. 2).

[11] Wissenschaft und Forschung Bundesministerium Bildung. *Digitale Grundbildung: Pflichtgegenstand im Schuljahr 2022/23 in der Sekundarstufe I – Lehrplan verordnet!* 2023. URL: https://www.bmbwf.gv.at/Themen/schule/zrp/dibi/dgb.html (visited on 03/10/2023) (cit. on p. 3).

[12] Abdul Majeed and Ibtisam Rauf. "Graph theory: A comprehensive survey about graph theory applications in computer science and social networks". In: *Inventions* 5.1 (2020), p. 10 (cit. on p. 4).

[13] Lawrence Page et al. *The PageRank citation ranking: Bringing order to the web*. Tech. rep. Stanford infolab, 1999 (cit. on p. 4).

[14] Renzo Angles and Claudio Gutierrez. "Survey of graph database models". In: *ACM computing surveys* 40.1 (2008), pp. 1–39 (cit. on p. 4).

[15] Edsger W Dijkstra. "A note on two problems in connexion with graphs". In: *Numerische Mathematik 1*. 1959, pp. 287–290 (cit. on p. 4).

[16] Daniela Ferrarello and Maria Flavia Mammana. "Graph theory in primary, middle, and high school". In: *Teaching and learning discrete mathematics worldwide*. Springer, 2017, pp. 183–200 (cit. on p. 4).

[17] Wissenschaft und Forschung Bundesministerium Bildung. *Lehrpläne der Allgemeinbildenden höheren Schulen*. 2023. URL: https://www.ris.bka.gv.at/GeltendeFassung.wxe?Abfrage=Bundesnormen&Gesetzesnummer=10008568 (visited on 03/10/2023) (cit. on p. 4).

[18] Wissenschaft und Forschung Johannes Kepler University. *UV Graphentheorie und Anwendungen*. 2019. URL: https://www.kusss.jku.at/kusss/lvaregistrationlist.action?coursegroupid=21417&abhart=all&courseclassid=48456 (visited on 02/02/2025) (cit. on p. 5).

[19] Robert Sedgewick. *Algorithms in Java, Part 5: Graph Algorithms: Graph Algorithms*. Addison-Wesley Professional, 2003 (cit. on p. 6).

[20] Jörg R Mühlbacher, Günter Pilz, and Marcel Widi. *Mathematik explorativ*. Trauner Verlag, 2006 (cit. on p. 6).

[21] Norman Biggs. *Discrete mathematics*. Oxford University Press, 2002 (cit. on p. 6).

[22] John Niman. "Graph theory in the elementary school". In: *Educational studies in mathematics* (1975), pp. 351–373 (cit. on p. 6).

[23] Phyllis Chinn. “Discovery method teaching: a case study using graph theory”. In: (1981) (cit. on p. 7).

[24] Daniel Lessner. “Graph Theory in High School Education”. In: *Department of Software and Computer Science Education, Faculty of Mathematics and Physics, Charles University, Prague, Czech Republic* (2011) (cit. on p. 7).

[25] J Paul Gibson. “Teaching graph algorithms to children of all ages”. In: *Proceedings of the 17th ACM annual conference on Innovation and technology in computer science education*. 2012, pp. 34–39 (cit. on p. 7).

[26] Rocio Blanco. “Graph theory for primary school students with high skills in mathematics”. In: *Mathematics* 9.13 (2021), p. 1567 (cit. on p. 7).

[27] Joel David Hamkins. *Graph Theory for Kids*. 2018. URL: `http://jdh.hamkins.org/wp-content/uploads/Graph-theory-for-Kids.pdf` (visited on 02/02/2022) (cit. on pp. 7, 14).

[28] IH Dafik et al. “Integrating a graph theory in a school math curriculum of Indonesia under realistic mathematics education”. In: *Int. J. Sci. Technol. Res* 9.1 (2020), pp. 2437–2454 (cit. on p. 7).

[29] Theresa Müssigang. “Algebraische Graphentheorie und Routenplanung im Schulunterricht”. MA thesis. Univesity of Innsbruck, 2019 (cit. on p. 8).

[30] Thomas Wassong. “Graphentheoretische Konzepte in der Gymnasialen Oberstufe - Ein Unterrichtsentwurf unter Beruecksichtigung der Neuen Medien”. MA thesis. Univesity of Göttingen, 2007 (cit. on p. 8).

[31] Dayna Brown Smithers. “Graph theory for the secondary school classroom”. MA thesis. East Tennessee State University, 2005 (cit. on p. 8).

[32] Brigitte Lutz-Westphal. “Wie komme ich optimal zum Ziel. Unterricht über kürzeste-Wege-Algorithmen für Graphen”. In: (2004) (cit. on p. 8).

[33] Melissa Windler. “Der Einfluss graphentheoretischer Konzepte im Mathematikunterricht der Grundschule auf psychologische Schülerinnen und Schülermerkmale”. PhD thesis. Univesity of Hildesheim, 2018 (cit. on p. 8).

[34] Jens Gallenbacher. *Abenteuer Informatik: IT zum Anfassen-von Routenplaner bis Online-Banking*. Springer, 2008 (cit. on p. 9).

[35] Sabitzer JKU COOL Lab. *Unterrichtspaket Kürzeste Wege der Dijkstra-Algorithmus*. 2019 (cit. on p. 9).

[36] Judith Lampl. "Ausgewählte Beispiele der Graphentheorie für den Schulunterricht". MA thesis. Johannes Kepler University, 2015 (cit. on p. 9).

[37] Ravensburger. *Mister X – Flucht durch Europa*. 2009. URL: https://de.wikipedia.org/wiki/Mister_X_%E2%80%93_Flucht_durch_Europa (visited on 01/10/2025) (cit. on p. 9).

[38] Janka Medová et al. "Undergraduate students' solutions of modeling problems in algorithmic graph theory". In: *Mathematics* 7.7 (2019), p. 572 (cit. on p. 9).

[39] Milena Corrales-Alvarez, Lina Marcela Ocampo, and Sergio Augusto Cardona Torres. "Instruments for Evaluating Computational Thinking: A Systematic Review". In: *TecnoLógicas* 27.59 (2024) (cit. on pp. 10, 20).

[40] Golnaz Arastoopour Irgens et al. "Modeling and measuring high school students' computational thinking practices in science". In: *Journal of Science Education and Technology* 29 (2020), pp. 137–161 (cit. on pp. 10, 20).

[41] Ashok Basawapatna et al. "Recognizing computational thinking patterns". In: *Proceedings of the 42nd ACM technical symposium on Computer science education*. 2011, pp. 245–250 (cit. on p. 10).

[42] Linda Seiter and Brendan Foreman. "Modeling the learning progressions of computational thinking of primary grade students". In: *Proceedings of the ninth annual international ACM conference on International computing education research*. 2013, pp. 59–66 (cit. on p. 11).

[43] Erich Gamma et al. *Design patterns: elements of reusable object-oriented software*. Pearson Deutschland GmbH, 1995 (cit. on p. 11).

[44] John Maloney et al. "Scratch: a sneak preview [education]". In: *Proceedings. Second International Conference on Creating, Connecting and Collaborating through Computing, 2004.* IEEE. 2004, pp. 104–109 (cit. on p. 11).

[45] Loice Victorine Atieno. "Are Computational Thinking Skills Measurable? An Analysis." In: *ICAI*. 2020, pp. 12–23 (cit. on p. 11).

[46] Timothy Ryan Duckett and Gale A Mentzer. "Measuring student computational thinking in engineering and mathematics: Development and validation of a non-programming assessment". In: *2020 ASEE Virtual Annual Conference Content Access*. 2020 (cit. on p. 11).

[47] Janka Medová, Gregor Milicic, and Matthias Ludwig. "Graph problems as a means for accessing the abstraction skills". In: *Twelfth Congress of the European Society for Research in Mathematics Education (CERME12)*. 07. 2022 (cit. on p. 11).

[48] Emre Coban and Özgen Korkmaz. "An alternative approach for measuring computational thinking: Performance-based platform". In: *Thinking Skills and Creativity* 42 (2021), p. 100929 (cit. on p. 11).

[49] S Wetzel, G Milicic, and M Ludwig. "GIFTED STUDENTS' USE OF COMPUTATIONAL THINKING SKILLS APPROACHING A GRAPH PROBLEM: A CASE STUDY". In: *Edulearn20 Proceedings*. IATED. 2020, pp. 6936–6944 (cit. on p. 11).

[50] Dominik Jochinger. *Github repository teaching material*. Jan. 26, 2024. URL: `https://github.com/nikrats/graphmaterial/` (visited on 01/26/2025) (cit. on p. 13).

[51] OEBB. *Fernreisen in Österreich: Ein Überblick über die wichtigsten Fernreiselinien und praktische Informationen für Ihre Reise.* 2025. URL: `https://www.oebb.at/de/fahrplan/fernreisen-oesterreich` (visited on 02/02/2025) (cit. on p. 16).

[52] Cabero-Almenara. "Digital teaching competence according to the DigCompEdu framework. Comparative study in different Latin American universities". In: *Journal of New Approaches in Educational Research* 12.2 (2023), pp. 276–291 (cit. on p. 19).

[53] M. Hohenwarter et al. *GeoGebra 5.0.507.0*. URL: `http://www.geogebra.org` (visited on 10/10/2018) (cit. on p. 20).

[54] Kern Sabrina. "GeoGebra bei Mathematikhausaufgaben eine Untersuchung zum GeoGebra-Einsatz von Schülerinnen und Schülern der 8. Klasse AHS bei Mathematikhausaufgaben". M.S. Thesis. University Vienna, 2022. URL: `https://phaidra.univie.ac.at/detail/o:1433359` (cit. on p. 20).

[55] RM Falcon and R Rios. "The use of Geogebra in discrete mathematics". In: *GeoGebra International Journal of Romania* 4.1 (2015), pp. 39–50 (cit. on p. 20).

[56] Trevor Kenneth Williams and David E Brown. "Graph Theory in GeoGebra". In: *North American GeoGebra Journal* 7.1 (2018) (cit. on p. 20).

[57] Patrik Vovstinar. "GeoGebra applets for graph theory". In: *EDULEARN17 Proceedings*. IATED. 2017, pp. 10142–10148 (cit. on p. 20).

[58] Franz Rittenschober. *Graph theory: Experimenting and proofing theorems of graphs.* 2019. URL: `https://www.geogebra.org/m/nerTbP7Q` (visited on 01/10/2025) (cit. on p. 20).

[59] K Schulz, S Hobson, and J Zagami. "Bebras Australia computational thinking challenge tasks and solutions". In: *Digital Careers, Brisbane, Australia* (2015) (cit. on p. 20).

[60] Maria Zapata-Caceres, Estefania Martin-Barroso, and Marcos Roman-Gonzalez. "Computational thinking test for beginners: Design and content validation". In: *2020 IEEE global engineering education conference (EDUCON)*. IEEE. 2020, pp. 1905–1914 (cit. on p. 20).

[61] Barbara Sabitzer and Stefan Pasterk. *Cool informatics: A new approach to computer science and cross-curricular learning*. Tech. rep. 2014, pp. 149–160 (cit. on p. 37).

[62] Barbara Sabitzer, Stefan Pasterk, and Sabrina Elsenbaumer. *Informatics is COOL: Cooperative and Computer-Assisted Open Learning*. Tech. rep. New York, NY, USA, 2013, pp. 91–94 (cit. on p. 37).

[63] Gavin Bierman, Martin Abadi, and Mads Torgersen. *Understanding typescript*. Tech. rep. 2014, pp. 257–281 (cit. on p. 37).

[64] Nilesh Jain, Ashok Bhansali, and Deepak Mehta. "AngularJS: A modern MVC framework in JavaScript". In: *Journal of Global Research in Computer Science* 5.12 (2014), pp. 17–23 (cit. on p. 37).

[65] Mike Bostock. *D3.js - Data-Driven Documents*. 2012. URL: `http://d3js.org/` (cit. on p. 37).