Improving Understanding of Data Structures for the Blind with Tactile Media and a User-Centered Iterative Approach

April R. Crockett
Department of Computer Science
Tennessee Technological University
Cookeville, TN USA
acrockett@tntech.edu

Gerald C. Gannod
Department of Computer Science
Tennessee Technological University
Cookeville, TN USA
jgannod@tntech.edu

***Abstract*—This Innovative Practice Full Paper addresses the challenges of teaching data structures and algorithms to blind students using tactile media and a user-centered approach. Computer Science educators have have long used diagrams and other visualizations to assist in teaching data structures and algorithms. As enrollments in computer science rise, a more diverse student body has led to widespread access to computer science, including an increase in the number of students with sight impairments that are seeking computer science degrees. In the Department of Computer Science at Tennessee Technological University we have been actively engaging in the development of methodologies and approaches necessary to facilitate creation of tactile documents suitable for helping students with sight impairments to understand and effectively use common data structures. Using tools generally available to practitioners, we applied a user-centered iterative process to develop standards necessary to create visual idioms that capture various data structures and algorithms as diagrams expressed using tactile documents. With feedback from our visually impaired students, we created several diagrams that represent various data structures using common drawing tools such as Microsoft Visio with Braille font. These documents were then printed using swell paper and a tactile printer. Students using these diagrams have reported gaining an increased understanding of concepts that were previously static abstractions only read about in their textbooks. Many lessons have been learned along the way that range from the general (i.e., how to properly space words in order to not affect ambiguity in the layout of diagrams) to the specific (i.e., how to demonstrate movement of data in a data structure using the tactile medium). In this paper, we report on the approaches used to create tactile diagrams representing various data structures, the ways in which we interacted with students in order to gain a better understanding of how to represent visual idioms in tactile form, challenges faced in creating the documents, and lessons learned. In addition, we discuss the work in the context of the state of the art, and suggest future investigations.**

I. Introduction

Data structures and algorithms are an essential part of knowledge in Computer Science curricula. Understanding data organization in data structures as well as dynamic processes such as the flow of data and working of various algorithms require visualizations and animations. There are extensive studies that have been done on algorithm visualization effectiveness with encouraging results [1]–[6]. Visualizations bring dynamic processes and algorithms to life by graphically representing their disparate states and animating transmutations between those states. They illustrate data structures in a natural, abstract way instead of just focusing on the details of implementation. Images and diagrams are required in data structures and algorithms courses to facilitate student comprehension and to pique interest in learning these core computer science concepts. Because of this, data structures is the most challenging course for a visually impaired student to complete in a computer science curriculum. Overcoming this challenge is important in order to increase the participation and performance of blind or visually impaired students in Computer Science.

In the Department of Computer Science at Tennessee Technological University we have been actively engaging in the development of methodologies and approaches necessary to facilitate creation of tactile documents suitable for helping students with sight impairments in order to increase understanding and effective use of common data structures. *Tactile* documents consist of documents that allow readers to interpret page contents through the sense of touch. Using tools generally available to practitioners, we applied a user-centered iterative process to develop standards necessary to create visual idioms that capture various data structures and algorithms as diagrams expressed using tactile documents. With feedback from our visually impaired students, we created several diagrams that represent flow charts, arrays, lists, trees, heaps, treaps, queues, stacks, hash tables, and graphs using common drawing tools such as Microsoft Visio with a Braille font. In addition, we used "animation" techniques to illustrate algorithms by showing progression of operations with several diagrams. These documents were then printed using Swell paper and a tactile printer. Students using these diagrams have reported gaining an increased understanding of concepts that were previously static abstractions only read about in their textbooks. Many lessons have been learned along the way that range from the general (i.e., how to properly space words in order to not affect ambiguity in the layout of diagrams) to the specific (i.e., how to demonstrate movement of data in a data structure using the tactile medium). In this paper, we present the approaches used to create tactile diagrams representing various data structures, the ways in which we interacted with students in order to gain a better understanding of how to represent visual idioms in tactile form, challenges faced in creating the documents, and lessons learned. The remainder of this paper is organized as follows. Section II discusses background information in the area of providing accessibility support to the blind and present related work. Our experiences and approaches for addressing the needs of our students is presented in Section III, including our techniques for interacting with students, resources used, and example documents. Section IV discusses evaluation metrics that we employ for our tactile documents using criteria provide by BANA [7]. Finally, Section V draws conclusions and suggestions for future investigations.

II. Background

In this section we describe background information and related work on accessibility for blind and visually impaired students.

A. Accessibility Approaches

Tactile Graphics are images that use raised surfaces so that visually impaired people can feel them with their fingers and interpret non-textual information such as maps, paintings, graphs, and diagrams. Tactile graphics are a subset of accessible images which are images translated as verbal description, sound, or textual (haptic) feedback. *Screen readers* are a form of assistive technology that converts text or other similar media into either audio or Braille, for instance. Screen readers are essential to blind or visually impaired students. Our students use Freedom Scientific's "Job Access With Speech" (JAWS) screen reader. The JAWS Scripting Language allows users to use programs without standard Windows controls and programs that were not designed for accessibility. *Sonification* is another assistive technique that uses non-verbal audio cues for the blind. Another significant assistive technology that is typically used is a *tactile printer* to print Braille onto *Swell paper*. In the work described in this paper we are primarily focused on the creation of tactile documents using tactile printers within the context of a data structures course.

B. Related Work

With rising enrollments in Computer Science and a much needed increase in diverse populations into STEM fields, there is some literature and work aimed toward helping blind students succeed in Computer Science. There is extensive literature on the benefits of learning data structures and other computer science fundamentals using visualizations and animations [1]–[6]. Zemke presented a very interesting case study in 2018 of successfully teaching engineering graphics to a blind student using 3D printed parts, a braille sketch pad, and computer assistive technologies [8]. However, there is very little work focused on teaching the visual aspects and animations specifically of data structures and algorithms to blind and visually impaired students.

1) Learning Data Structures Through Visualizations: Alzubaidi et al. present research showing that visual aids and multimedia contribute positively to the understanding of data structures in computer science [9]. Shaffer et al. has also presented work on improving computer science undergraduate instruction in algorithms and data structures by providing students with data structure visualizations and animations into the curriculum [5]. Each of these highlights the fact that a similar medium is necessary as a supplement to text used to illustrate data structures. As such, we have focused on attention on an alternate media for sight impaired students.

2) Programming: Various software has been created to assist blind students write programs. Franqueiro and Siegfried created a scripting language to help blind students program in Visual Basic, which is a highly-visual programming language [10]. In addition, Stefik et al. focused on designing education infrastructure for the blind in computer science in which they came up with an auditory programming environment called Sodbeans and a programming language called Hop that allowed students to program with an auditory debugger and to write code that connected to their screen reader [11]. Each of these highlights the fact that identifying ways to support blind software developers has been of interest in the community for some time.

3) Enhancing Understanding of Data Structures and Algorithms: Calder et al. have focused on demonstrating algorithm animations through software called PLUMB EXTRA which conveys an algorithm animation using audio cues and synthesized speech [12]. Califf et al. used a closed-circuit television (CCTV) to allow their visually-impaired students to magnify documents containing AVL tree rotations and graphs in order for them to see the animations [13]. This worked well for their students, who were not blind but had a vision impairment. Connelly discussed efforts to educate a blind student in Data Structures by using tactile objects such as dominoes [14]. Other researchers discussed that they developed learning activities that used tactile manipulative objects to teach computing concepts [11]. These methods work well in very small class sizes, but will not work in classes of over seventy students like ours at Tennessee Technological University.

4) Tactile and Haptic Methods: There are a variety of methods that have been researched to produce alternate-media versions of images. Hasper et al. describe the 3D IMAGINE project where by they have developed methods to create and evaluate 3D tactile images to aid in teaching STEM courses to visually impaired students [15]. Their studies demonstrate the usefulness of creating tactile STEM models for blind and sighted students and included visualizations for astronomy and biology.

Balik et al. created specialized, accessible software that could help blind (and sighted) students view and create node-link diagram (graphs) [16]. Fitzpatrick et al. [17] developed an approach that will allow blind students to understand statistical graphs such as discrete and continuous data graphs by using the statistical software environment R to compute semantics for these diagrams and make them web accessible [16]. Ohene-Djan et al. presented a set of grammar tools that can be used by blind people to draw [18]. These tools enabled users to present graphics by mapping the cognitive visualization of blind people into spatial information on a computer screen and promoted an interactive drawing environment to build objects, associations, and layout information. There is also work being done [19] to use haptic and auditory interaction tools to engage Kindergarten through 12th grade students with visual impairments in robot programming. This work was focused on making computer science exciting for visually impaired students and allowing these students to become an active participant in robotics-based computing activities. Our work at Tennessee Technological University is similarly oriented towards touch and haptic methods. Our approach focuses on use of tactile printing through the development of a user-centered methodology for developing diagrams.

III. Approach

In this section we describe the context for our work including the *students*, the *out of the box* supports available in commonly used components of instruction, and the ways in which we develop *accommodations* for visual media, including the use of user-centered feedback and iteration. We also provide a number of examples that illustrate both the challenges faced as well as lessons learned.

A. Students

The Department of Computer Science at Tennessee Technological University offers Bachelors, Masters, and Ph.D. degrees and consists of approximately 600 total students. At present, we have a small population of students with visual impairments, including students that are blind. In this paper we describe our work on how we developed support for visualizations normally meant for sighted students. In particular, we make reference in this paper to some of our blind students that have recently entered our program. The students entered our Bachelors program in Fall 2018 and Fall 2019, respectively, and are high achieving individuals, with each taking part in advanced studies through the honors program or through undergraduate research. An important quality of these students is their level of engagement with instructors, making the ability to develop accommodations both easier and enjoyable.

B. "Out of the box" Accessibility in CS

Research suggests that course materials should be provided in several different formats (e.g., visual, auditory, and text-based) to enable students of different learning styles to absorb the concepts [20], [21]. This can be especially challenging with visual information and concepts required by the topic of data structures and demonstrating dynamic changes in algorithms. Below are a number of common resources that students regularly access along with accessibility accommodations provided in each "out of the box".

1) Textbooks: We prefer to use digital, online textbooks provided by Zybooks [22] in a number of courses, including our Data Structures course. The textbooks provide students with a mixture of formats including textual information, simulations and animations of data structures, sample code and pseudo-code, and practice exercises that can be assigned as homework. The practice exercises include multiple-choice questions, fill-in-the-blank questions, definition-matching questions, and programming practice where students can write, compile, and run code directly within the Zybooks environment. Visually impaired students are able to have their screen reader read the Zybooks text and sample code, allowing them to read and answer the practice exercises just as their sighted counterparts. However, the animations and visualizations cannot be translated by the screen reader, thus motivating the work described in this paper. As such, while the online textbook is still a better and more interactive option than a PDF version of an e-book, it lacks some of the active learning supports afforded to sighted students. ZyBooks is available in an "accessibility mode" which increases color contrast, makes coding windows easier to use with screen readers, and replaces definition match activities with an accessible and equivalent alternative. Traditional textbooks are usually provided electronically in PDF format, which is typically not translated properly by screen readers, especially when they contain equations.

2) Equations: Studying algorithm efficiency requires equations to be conveyed to students. Our students have indicated that the best way to write equations for their screen reader to translate is by providing the equation in $\TeX$ and $\LaTeX$. Another accessible way to provide equations for students is by using MathType, which is software created by Design Science. We use a MathType plugin in the Microsoft Word application. Using MathType, students can toggle equations in Microsoft Word to other math markup languages such as $\TeX$, $\LaTeX$, and MathML. Using the built-in Microsoft Word equation editor is not read properly by the screen reader (JAWS) that is used by our students. If we have a PDF document that contains an equation that we want to make accessible to our students, we use EquatIO by Texthelp Ltd. EquatIO allows educators to take a screen capture of an equation in any format and it will translate that equation to $\LaTeX$, MathType, or many other formats that you may need. After translating, the educator can edit the equation and can also listen to how the equation will be read by screen readers to make sure it is correct.

3) Integrated Development Environments: While we do not require that students use specific *integrated development environments* (IDEs), text editors or compilers, we do make various recommendations including that they use Microsoft's Visual Studio Code (VSCode) to write and compile code in C++. This allows students to explore and find the compiler or IDE that works best on their devices. We also allow students to email us their solutions to programming assignments before the due dates to ensure that their program will compile on our computer as well as theirs.

Our blind students have reported that IDEs (and software in general) that require minimal mouse clicking always work better with screen readers. For coding they mostly used VSCode, because it has a code-completion which is accessible for screen readers. VSCode also allows the students to create tasks, which will allow them to press a key combination to allow the task to run such as `ctrl+shift+b` to build a project.

4) Lectures: We teach fundamental data structures and algorithms by first presenting an example through a diagram or animation. Then, we present less abstract examples of the data structure. Then, we demonstrate more concrete examples of the data structure or algorithm with pseudo code and C++ code.

We use a combination of methods to convey information in the classroom. Speech, presentation slides, demonstrations with stylus and OneNote, code examples, and sharing online resources. The U.S. Individuals with Disabilities Education Act (IDEA) [23] mandates that equal education opportunities be provided for students in the most integrated setting possible. For visually impaired students, speech is no problem. Microsoft PowerPoint presentation slides are delivered to the students before class (which can be read by a screen reader) and they are presented with speech and appropriate tactile diagrams for visual information. Tactile diagrams are also provided for visualizations and animations of algorithms and data structures that the instructor demonstrates on the computer screen with a stylus. Websites that use CSS/HTML are great for screen readers so they present no problem for students with visual impairments.

We are careful to minimize exclusive non-verbal communication such as head shakes, nodding, and pointing. Also, it is important to verbalize exactness instead of saying "this" or "that". We try to be mindful to verbalize and describe everything in the classroom. Sometimes, we have students demonstrate algorithms or do live-coding in front of the class. These students usually do not verbalize their actions to be inclusive of blind students and so we verbalize their actions for them.

C. Developing Accommodations for Visual Media

As we have discussed previously, much of the study of data structures uses the visual genre to convey information about important concepts, which creates impediments for students with visual impairments [12]. Our process for developing accommodations for visual media includes:

- Design
- Feedback
- Iteration
- Construction and Printing

1) Design: The first task of developing visual media for blind students is to design the diagram. We have created diagrams of all data structure visualizations as well as all algorithm animations discussed in our Data Structures and Algorithms course. The first line of each diagram has a label or title indicating the purpose of the diagram and a diagram number. The purpose of the diagram communicates the genre of the data structure, thus providing context to the reader. The diagram numbering is based on the order that the diagram is presented within the particular chapter that is being covered at the time. The diagrams often contain shapes in addition to the Braille font labels and descriptions. Some of the tactile diagrams we use allow us to show how data is organized in a data structure. For example, a singly-linked list is shown below in Figure 1(a), and Figure 1(b) shows the same document in Braille. The singly-linked list in Figure 1(a) shows three nodes, each with a corresponding integer value and a pointer link to subsequent nodes in the list (or to "NULL"). In the case of these diagrams, the labels on the Braille version (i.e., Figure 1(b)) communicate important features of the diagrams including the *head* and the *tail* references of the list, and the labels in the boxes indicating values of each list item. In addition, since the tactile printers draw lines, readers can easily find the connections between data structure elements. Proximity of the head and the tail communicates which items in the list are the head and the tail, respectively. We have used diagrams to represent flow charts, arrays, lists, trees, heaps, treaps, queues, stacks, hash tables, and graphs in a similar way.

Another important aspect of learning about data structures is the fact that the data contained in them are dynamic, with the internal relationships between different elements changing over time. We demonstrate changes in a data structure by creating multiple versions of the structure in several different diagrams. Annotations on the diagrams allow us to communicate different state changes in the data structures, with the top line in the diagram providing context (i.e., the data type being represented). For example, we can demonstrate a left-right rotation of an AVL tree by providing three diagrams in one document. Refer to Figure 2(a), which shows the left-right rotation of an AVL tree and then Figure 2(b), which shows the same document in Braille. Each stage of the respective diagrams represents changes to the data structure after a balancing operation is performed. Specifically, the first stage shows the AVL tree before rotation, the second stage in the middle shows the AVL tree after the first rotation, and the last stage in the diagram shows the state of the tree after the second rotation. The Braille version of the diagram in Figure 2(b) depicts exactly the same information using the standards we describe later governing guidelines such as proximity and content of the labels, structure implied by the lines, and context (i.e., the data structure type) to convey internal structure.

2) Feedback and Iteration: An important aspect of creating diagrams for tactile readers is the employment of user feedback. That is, we are very cognizant of being *user-centered* in our creation of diagrams in a tactile media. The instructor of the data structures course acquired regular feedback from the students in various ways. One way was via email communication. However, the most helpful feedback was with quick conversations directly after each class period with each student where the instructor would ask questions about the understanding of the content in the diagrams used during lecture. Occasionally, further discussion was required

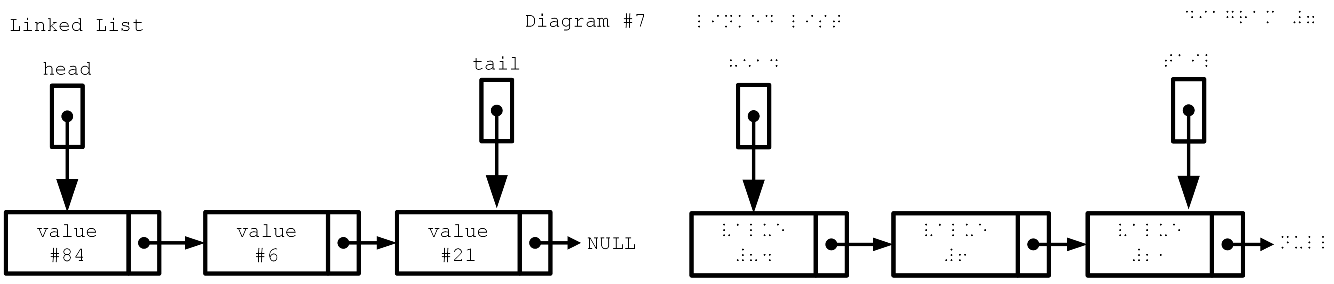


Fig. 1. Singly Linked List

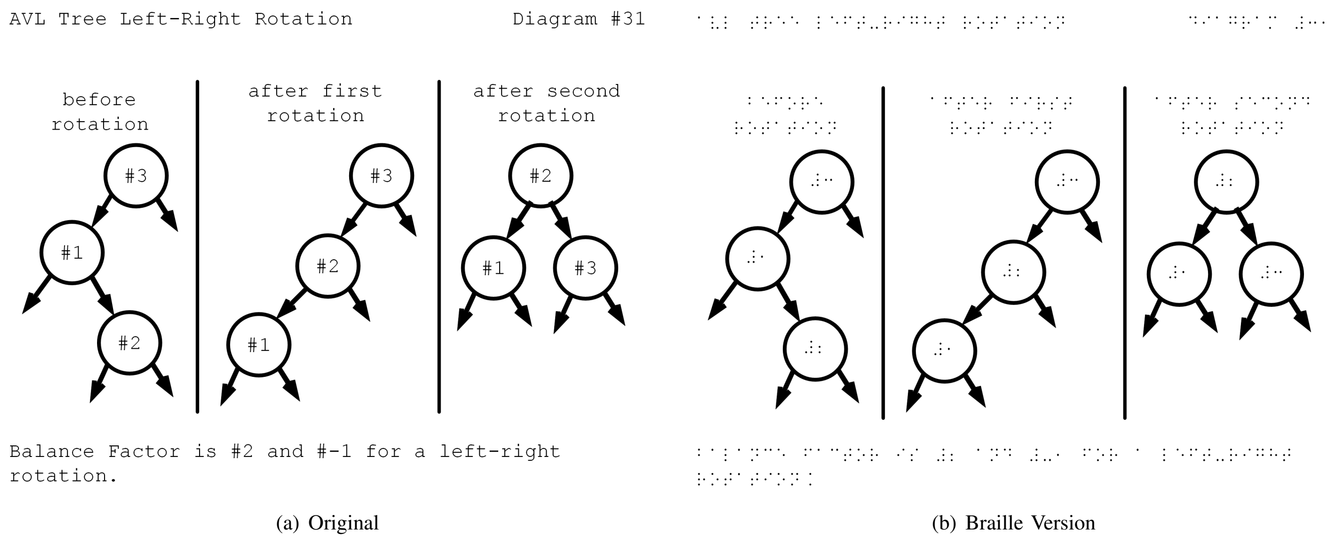
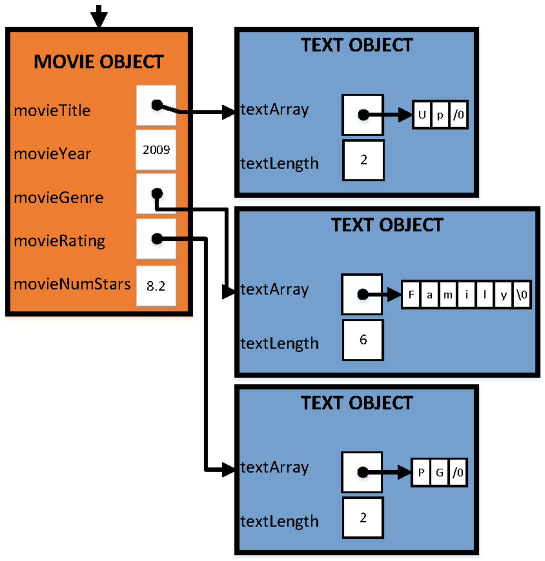


Fig. 2. AVL Tree

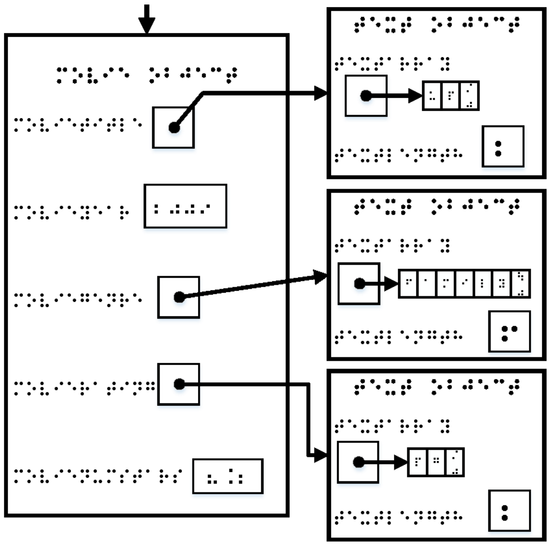
and therefore individual meetings were helpful to provide the students deeper understanding of the content and for the instructor to determine how the diagrams could be improved. One of the early diagrams we created simply did not lend itself in the tactile media format, which led to a valuable lesson learned on our part. Our first programming assignment in data structures introduces students to using classes and objects in C++. As typical with this assignment, we created a diagram that conveyed classes and their interrelationships. In Figure 3 we show a portion of this diagram where we demonstrated classes and relationships between the class data to increase understanding of the assignment. The diagram shows a *Movie* object with a number of attributes including a title, year, genre, rating, and stars. Each of the string attributes were then mapped to text objects as shown in Figure 3(a). While this diagram worked well for our sighted students, the corresponding *naive* translation in a tactile version for our blind students had many issues. There was too much information to be conveyed in a single document and the text was too small. In addition, some of the lines in the document were too close together to be felt individually. As such, in the case of Figure 3, we had to verbally describe the relationships since the diagram was ultimately unusable.

We found in the long term, that the most useful diagrams were developed in an iterative process using student feedback. For example, consider the diagrams shown in Figure 4. Using a more user-centered and iterative approach we were able to successively refine representations of binary tree structures from the original (shown in Figure 4(a)) to a more complete representation (shown in Figure 4(c)). Note that in the diagrams we represent here that the diagram we provided to the students were ultimately printed with the Braille font but for brevity sake are omitted from the paper.

A sub-section of the first diagram we created to introduce the tree data structure is in Figure 4(a). The diagram shows a subtree with a node labeled “18” in the root of the subtree with left and right children 15 and 6, respectively. The dots in the nodes lowest row of the tree depict NULL pointers. Each of the nodes are connected to each other with an arrow to provide directionality. With our student’s feedback we learned that there were several issues with this diagram that made it unreadable. First, all text should be the same size and be 24 point font. Second, we learned that numbers in Braille must have a ‘#’ in front of them to indicate that it is a number. The reason why is because the capital letter ‘A’ and the number ‘1’ both have a single dot for their Braille representation; therefore



(a) Original



(b) Braille Version

Fig. 3. Movie Library

using the ‘#’ makes it clear that we are using a number. We then gave the students the next iteration of this diagram in Figure 4(b). The students told us that they could now read the text and tell the difference between the numbers and letters, but there were still significant improvements that could be made. For instance, the diagram was too cluttered and there was too much being presented on one page. Also, some of the diagram was not printed due to the content being outside the printing range. They additionally said it would be helpful to put the diagram numbers and title of the diagram at the top of the page and to leave some space before the diagram begins. They also reported that it was sometimes difficult to distinguish right child from left child and that it would be helpful to indicate the NULL pointers in the tree. The last change was to increase the size of the arrow heads so that they are more distinguishable and better indicate the direction of the pointers. The diagram in Figure 4(c) represents the final version of this diagram after all the improvements. These improvements include all of those features mentioned above. The students reported that this diagram was very usable and exactly conveyed the structure of a binary tree.

3) Creation and Printing: We are unable to read braille font proficiently so we first create the diagrams using visual diagramming software such as Microsoft Visio, in 24 point New Courier font. We use New Courier because it is a mono-spaced font where letters and characters each occupy the same amount of horizontal space, which is the same for braille font. We save this version of the diagram so that we can use it to make future iterations. Then, we convert the font to Braille Kiama True Type font with no shadow dots because this is the recommended font to be used with the specialized paper needed for printing tactile documents. We downloaded the Braille Kiama True Type font from the Texas School for the Blind and Visually Impaired website [24]. The font size that prints best on Swell paper is bold, 20 to 24-point. We save this version of the diagram and then we ultimately create a PDF version. Our use of a tactile printer in the context of a computer science program and data structures course is one of the features of our approach that enables scaling to larger audiences. While the approaches of Connelly [14] and Stefik et al. [11] address similar problems, we believe that use of manipulatives will not scale as well as the printed option.

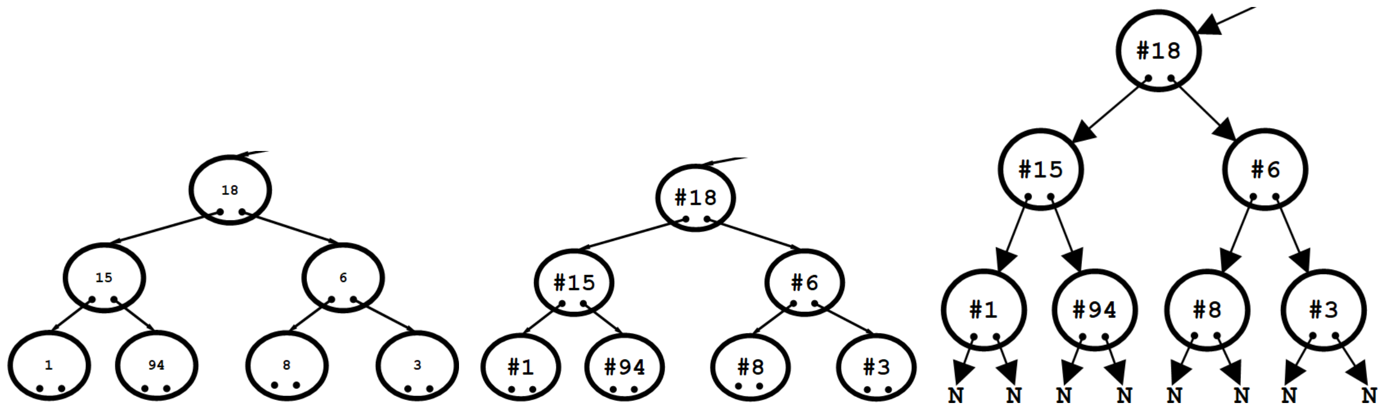
The diagrams are printed on Swell paper with a tactile printer called TactPlus. Swell paper is durable fine thermal foamed capsule paper that “swells” along specified black/dark lines when it is heated by the TactPlus printer, which prints Braille and graphics simultaneously. The printer can only print on 8.5 x 11 standard-sized Swell paper, so diagrams are limited to that size. A diagram takes approximately 3 to 5 minutes to print. One interesting feature of the TactPlus printer is that it provides sound guidance during the printing process to assist visually impaired users.

IV. Evaluation

A. Quality Criteria

The Braille Authority of North America and the Canadian Braille Authority has specified a complete set of “Guidelines and Standards for Tactile Graphics” [7] for assisting those that are interested in creating accessible tactile documents. In this section we identify the standards that we followed to create quality diagrams.

1) Content: The most important task of creating a tactile diagram is deciding on the purpose and the critical information that needs to be portrayed by the diagram. The drawing needs to be simplified to only the necessary components. There should be a heading or title at the top of the diagram. When using a number, you must type a number-sign ‘#’ before the number. There are some symbols that should be spelled out



(a) Original (b) Second Iteration (c) Third Iteration

Fig. 4. AVL Tree

instead of the symbol used. For example, write the word "and" instead of using an ampersand. If the diagram contains an equation, there are some guidelines for symbols that should be typed to represent the operands and functions. For example, fractions should begin with a question mark and end with a number sign (?1/3# is one-third).

2) Size, Spacing, and Lines: Diagram components should not be cramped or cluttered. There should be appropriate space between shapes, lines, and labels so that the reader's fingertip can sense the separation of content. Braille should be between 20 to 24 point font in size, bold, and 1.5 spacing. Lines must be far enough apart to be able to distinguish them separately. There should be one blank line between the diagram heading and the graphic.

3) Labeling: Diagram components often need labels. There should be just enough space between the component and the label to distinguish them but still be able to tell that the label belongs to that component. Abbreviations may be made for labels, but make sure to follow the approved abbreviations in the guidelines document mentioned earlier.

B. Student Feedback and Changes

The first diagram we created for the Data Structures class was a flow chart to demonstrate algorithm steps. One of the students claimed that he had previously read in a textbook how a flowchart worked, but had never been able to visualize one in his mind until he read through the tactile flow chart diagram. This was the defining moment where we knew the diagrams were increasing understanding of the content. Both students offered advice throughout the semester on how some of the diagrams could be improved for usability, some of which are described below.

1) Content: With feedback from the students, were were able to simplify the content being displayed on the diagram to make it less cluttered. For example, in the first diagrams we created for the students we originally had several sentences of text explaining what the diagram was, which was unnecessary for understanding the content. Another content change was that students informed us of their difficulty of identifying a specific graphic that was being talked about in class and so we developed a numbering system for the diagrams.

2) Size, Spacing, and Lines: Some spacing issues arose when we created diagrams where the font was too bold, which caused the braille dots to over-swell and resulted in students not being able to distinguish the individual dots. There were also times where lines in a diagram were too close together, which would change the meaning of the diagram. Some of our graph diagrams had intersecting edges, which the students reported were difficult to follow. In some cases, we were able to move graph vertices around to where the graph edges were no longer intersecting. In others, we added an adjacency matrix to indicate edges.

3) Labeling: We provided some weighted, digraph diagrams where we had a weight label for each edge. Occasionally we provided a diagram in which students indicated that they were unclear of the matching between edges and weights. We found that there is a balance between making sure the label is not too close to the line so that it can still be read, but not too far away so that students could tell which edge the label belonged to. For dense graphs, we sometimes had to remove the edge weights and instead provide an adjacency matrix with the edge weights.

C. Student Performance

In addition to being able to discuss topics and ask relevant questions in class, the visually impaired students performed well on exams and programming assignments. They were able to demonstrate that they had gained a deep understanding of data structures through the tactile-diagrams, accessible course materials, and lectures. We appreciate that we could have gained more knowledge by comparing performance of students using the diagrams to the performance of students not using the diagrams by our blind student population. However, we do not have substantial population in order to do this comparison.

D. Limitations

Creating usable, accessible, tactile diagrams that accurately present data structures and algorithms take significant time and dedication. Diagrams take longer to print then a standard document. Therefore, instructors must prepare way in advance of the time when they expect to deliver the content covered by the diagrams. Also, the space limitation of a standard-size page means that diagrams are limited in size. A large graph or binary tree would have to be reduced, which may cause loss of meaning for a particular algorithm or animation. Instructors must also have access to a tactile printer with specialized Swell paper in order to print the documents.

V. Conclusions and Future Investigations

Our approach to addressing the accessibility of diagrams used in a data structures course is, by necessity, a user-centered, iterative process. We engage our sight impaired students in this endeavor in order to gain deeper awareness into what their needs are with respect to communication of concepts that are embedded in diagrams for which we may have some inherent bias. Our user-centered, tactile diagram creation efforts allowed us to create usable diagrams to increase understanding of data structures and algorithms for our blind students. Although time-consuming for the instructor, we were able to provide a visual and deeper understanding of fundamentals in data structures, which would have been impossible otherwise.

We often evaluate a student's understanding of a data structure or workings of an algorithm by having the student construct and demonstrate their own visualizations. The effectiveness of this practice in our classroom is based on the social constructivist learning theory [25]. Although we have figured out a method to create tactile diagrams for our blind students, we would like to investigate methods for which our blind students can create and present visualizations for assessment of their understanding and to increase their understanding. This would require researching methods which would automate the creation of the diagrams. In addition, we are interested in engaging sighted students to determine how they can better communicate similar information in order to prepare them to work in diverse teams that might include those with sight impairments.

References

[1] Slavomír Šimoňák. Algorithm visualizations as a way of increasing the quality in computer science education. In *SAMI 2016 - IEEE 14th International Symposium on Applied Machine Intelligence and Informatics - Proceedings*, pages 153–157. Institute of Electrical and Electronics Engineers Inc., mar 2016.

[2] J. Ángel Velázquez-Iturbide and Celeste Pizarro-Romero. A study on students' preferences in graphical design of algorithm visualizations. In *2016 International Symposium on Computers in Education, SIIE 2016: Learning Analytics Technologies*. Institute of Electrical and Electronics Engineers Inc., nov 2016.

[3] Slavomír Šimoňák. Using algorithm visualizations in computer science education. *Open Computer Science*, 4(3):183–190, 2014.

[4] Vassilios Lazaridis, Nikolaos Samaras, and Angelo Sifaleras. An empirical study on factors influencing the effectiveness of algorithm visualization. *Computer Applications in Engineering Education*, 21(3):410–420, 2013.

[5] Clifforda Shaffer, Matthew L. Cooper, Alexander Joel D. Alon, Monika Akbar, Michael Stewart, Sean Ponce, and Stephen H. Edwards. Algorithm Visualization. *ACM Transactions on Computing Education*, 10(3):1–22, aug 2010.

[6] Christopher D. Hundhausen, Robert Patterson, Jonathan Lee Brown, and Sean Farley. The effects of algorithm visualizations with storylines on retention: An experimental study. In *Proceedings - 2004 IEEE Symposium on Visual Languages and Human Centric Computing*, pages 226–228, 2004.

[7] Braille Authority of North America and Canadian Braille Authority. Guidelines and Standards for Tactile Graphics. 2011.

[8] Steven C. Zemke. Case study of a blind student learning engineering graphics. In *Proc. of the ASEE Annual Conference and Exposition*, 2019.

[9] Loay Alzubaidi and Ammar El Hassan. Data Structures Learning-A Visually Assisted Approach. In *Proc. of the International Conference on Computer Graphics and Virtual Reality (CGVR) The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp)*, page 37, 2013.

[10] Kenneth G Franqueiro and Robert M Siegfried. Designing a Scripting Language to Help the Blind Program Visually. In *Eighth International ACM SIGACCESS Conference on Computers and Accessibility, ASSETS 2006*, pages 241–242. Adelphi University, 2006.

[11] Andreas Stefik, Christopher Hundhausen, and Derrick Smith. On the Design of an Educational Infrastructure for the Blind and Visually Impaired in Computer Science. In *SIGCSE '11: Proceedings of the 42nd ACM technical symposium on Computer science education*, pages 571–576. ACM, 2011.

[12] Matt Calder, Robert F. Cohen, Jessica Lanzoni, Neal Landry, and Joelle Skaff. Teaching data structures to students who are blind. In *ITiCSE 2007: 12th Annual Conference on Innovation and Technology in Computer Science Education - Inclusive Education in Computer Science*, pages 87–90, New York, New York, USA, 2007. ACM Press.

[13] Mary Elaine Califf, Mary Goodwin, and Jake Brownell. Helping him see: Guiding a visually impaired student through the computer science curriculum. In *SIGCSE'08 - Proceedings of the 39th ACM Technical Symposium on Computer Science Education*, pages 444–448, New York, New York, USA, 2008. ACM Press.

[14] Richard Connelly. Lessons and Tools from Teaching a Blind Student. *Journal of Computing Sciences in Colleges*, 25(6):34–39, 2010.

[15] Eric Hasper, Rogier Windhorst, Terri Hedgpeth, Leanne Van Tuyl, Ashleigh Gonzales, Debra Baluch, Britta Martinez, Hongyu Yu, and Zoltan Farkas. Research and Teaching: Methods for Creating and Evaluating 3D Tactile Images to Teach STEM Courses to the Visually Impaired. *Journal of College Science Teaching*, 044(06):82–89, 2015.

[16] Suzanne Balik, Sean Mealin, Matthias Stallmann, Robert Rodman, Michelle Glatz, and Veronica Sigler. Including blind people in computing through access to graphs. In *ASSETS14 - Proceedings of the 16th International ACM SIGACCESS Conference on Computers and Accessibility*, pages 91–98. Association for Computing Machinery, Inc, oct 2014.

[17] Donal Fitzpatrick, A. Jonathan R. Godfrey, and Volker Sorge. Producing accessible statistics diagrams in R. In *Proceedings of the 14th Web for All Conference, W4A 2017*. Association for Computing Machinery, Inc, apr 2017.

[18] James Ohene-Djan and Sandra Fernando. Drawing for blind learners: Assistive technology for graphical design. In *Proceedings - IEEE 16th International Conference on Advanced Learning Technologies, ICALT 2016*, pages 436–440. Institute of Electrical and Electronics Engineers Inc., nov 2016.

[19] Ayanna M. Howard, Chung Hyuk Park, and Sekou Remy. Using Haptic and Auditory Interaction Tools to Engage Students with Visual Impairments in Robot Programming Activities. *IEEE Transactions on Learning Technologies*, 5(1):87–95, 2012.

[20] Christine Brown and Christine Brown. Learning Through Multimedia Construction: A Complex Strategy. *Journal of Educational Multimedia and Hypermedia*, 2007.

[21] BrainPOP research team. Understanding Multimedia Learning. https://aboutbrainpop.wpengine.com/wp-content/uploads/2017/11/Understanding-Multimedia-Learning-2014.pdf.

[22] Zybooks. Zybooks. https://www.zybooks.com/.

[23] IDEA. Individuals with Disabilities Education Act (IDEA). https://sites.ed.gov/idea/.

[24] TSBVI. Texas School for the Blind and Visually Impaired Braille Fonts. https://www.tsbvi.edu/download-braille-and-asl-specialty-fonts.

[25] Christopher D. Hundhausen. Integrating algorithm visualization technology into an undergraduate algorithms course: Ethnographic studies of a social constructivist approach. *Computers and Education*, 39(3):237–260, 2002.