An Auto-Generation Approach to Create Visualization Teaching Materials for Data Structures and Algorithms in MS-PPT Format

Sen Zhang, *Member, IACSIT*

***Abstract*—Computer science education community has long been searching for effective and feasible visualization solutions to facilitate teaching and learning of data structures and algorithms (DSAs). This paper presents a novel and practical approach that involves first creating generator programs to exploit the rich presentation features, the standardized format and powerful auto API of Microsoft PowerPoint (MS-PPT). These generators can then be used to automatically and systematically create DSAs teaching slides consisting of nontrivial, neat and consistent graphical illustrations. The approach has been prototyped and experimented at a small scale. The preliminary results of using the approach are promising. The advantages and challenges of the approach are discussed and the current status of the project building upon the approach is reported.**

***Index Terms*—Data structures, algorithms, visualization, auto generation, computer application, education reform and innovation, curricula and courseware design.**

I. Introduction

One of the core knowledge areas of computer science (CS) education of any quality CS curriculum is the study of fundamental and classical data structures and algorithms (DSAs) [1]-[4]. While most data structures are static placeholders thus relatively straightforward to teach, the algorithms that manipulate them are usually dynamic, combinatorial and abstract, thus can be challenging to teach. As the knowledge body of CS ever expands, students tend to have less and less time to spend on any single subject, and teaching of nontrivial DSAs has inevitably become increasingly more challenging than before. An intuitive way to improve quality, efficiency and effectiveness of teaching of nontrivial DSAs is to observe data structures and algorithms in action by using proper visual aids to precisely illustrate how the algorithms dynamically work on data structures step by step on nontrivial yet representative datasets. However, producing a meaningful and non-trivial algorithm illustration that can show the consecutive changes made by the algorithm to the underlying data structures and/or the data held by them requires repetitively, proportionally, accurately, systematically and consistently positioning, re-positioning, removing and sometimes scaling, translating, and rotating proper visual/graphic elements (shapes or other graphical notations). These often are extremely tedious to do by hands on paper or board. Therefore it has been in great need for DSA instructors to have access to specialized software solutions that can automatically produce high quality digital visualizations to facilitate their teaching and enhance their students' learning of DSAs.

In fact, the idea of using visualization software in DSA education has been a recurring topic [5]-[8] to the CS education research community. During the past decades, many DSA visualization toolboxes have been proposed and experimented, much research have been done to build the theoretical foundations for DSA visualization approaches, and a great deal of effort has been made to assess the effectiveness of using the DSA visualization systems. Despite it is generally believed that many existing DSA visualization toolboxes, if properly used, can be valuable for DSA teaching and learning community, no strong evidence has shown any of them has achieved widespread adoption as their developers might have hoped [5]-[9]. While reasons for the phenomena are complicated and remain to be further investigated, we felt that one possible discouraging factor could be due to the sharp learning curve for using diverse toolboxes (standalone, web-based, Java-based, flash-based, app-based etc.)

In this work, we focus on a practical aspect of the DSA visualization: how to partially standardize DSA visualization. Specifically, we started out from using the Microsoft PowerPoint (MS-PPT) as a promising platform for building non-trivial DSA visualization. It is known that PPT is light-weighted, standardized, stable, accessible, and easy to adopt, especially when the content is primarily text based. In DSA visualization where the mere appearance of text or visual objects is no longer the major concern; instead stepwise illustration usually involves accurate calculations and/or scaled drawings presented in a repetitive and/or recursive manner. As a result, high-quality DSA PPT visualization can still be extremely tedious, labor intensive, error-prone and prohibitively expensive to produce by hand editing; there has still been a surprising paucity of meaningful, high quality, PPT-based, reusable non-trivial algorithm visualization materials for CS instructors to adopt.

Motivated by the above observations, we have explored a MS-PPT based auto-generation approach in which we first code separate generators for different DSAs [7], [8]. These generators can then be re-used to automatically and effectively generate multiple DSA visualization presentation materials in the PPT format, which subsequently can be adopted by many instructors to facilitate their teaching. The preliminary results of a project that has pilot implemented the

Manuscript received April 7, 2014; revised July 7, 2014.

Sen Zhang is with the Department of Mathematics, Computer Science and Statistics, SUNY College at Oneonta, Oneonta, NY, U.S.A. 13820 (e-mail: zhangs@Oneonta.edu).

idea show that the approach appears to be promising.

The paper is organized as follows. In the next section, we describe our approach. In Section III, the advantages of the approach will be future discussed. In Section IV, the preliminary results of the project based on the approach will be reported. Finally, we conclude our paper in Section V with discussion on future work.

II. A MS-PPT Based Auto-Generation Approach

A. Our Approach

In our approach, a dedicated generator needs to be coded for each DSA. In this step, we first correctly implement the underlying algorithm itself, typically in an object oriented way, with all the relevant methods of the algorithms properly decoupled and thoroughly debugged; then we inject slide generation snippets into the proper methods of the algorithm implementation. This way, as the generators run, the critical underlying data structures will be continuously manifested in the forms of tables, basic shapes, and diagrams with the changes being recorded by a sequence of consecutive slides to produce a "filmstrip" of the algorithm in animation. As a result, dozens of presentation slides can be created automatically, accurately, and efficiently, in minutes, instead of hours or days. This will dramatically simplify the otherwise tedious and labor-intensive DSA visualization PPT manual production process. Fig. 1 shows the screenshot of a snapshot of the PPT slides auto-generated by a pilot generator for the Kruskal's algorithm. From the figure it is easy to see that the auto-generated slides are neat and consistent. The slides are easy to be post processed and can be combined with other primarily text-based PPT presentations for the DSAs under investigation. Technically, this is the most critical component of the approach, because it highly relies on the expert knowledge on both the DSA, programming skills, and the familiarity with the PPT auto API (a hierarchy of classes of MS Office and MS PPT).

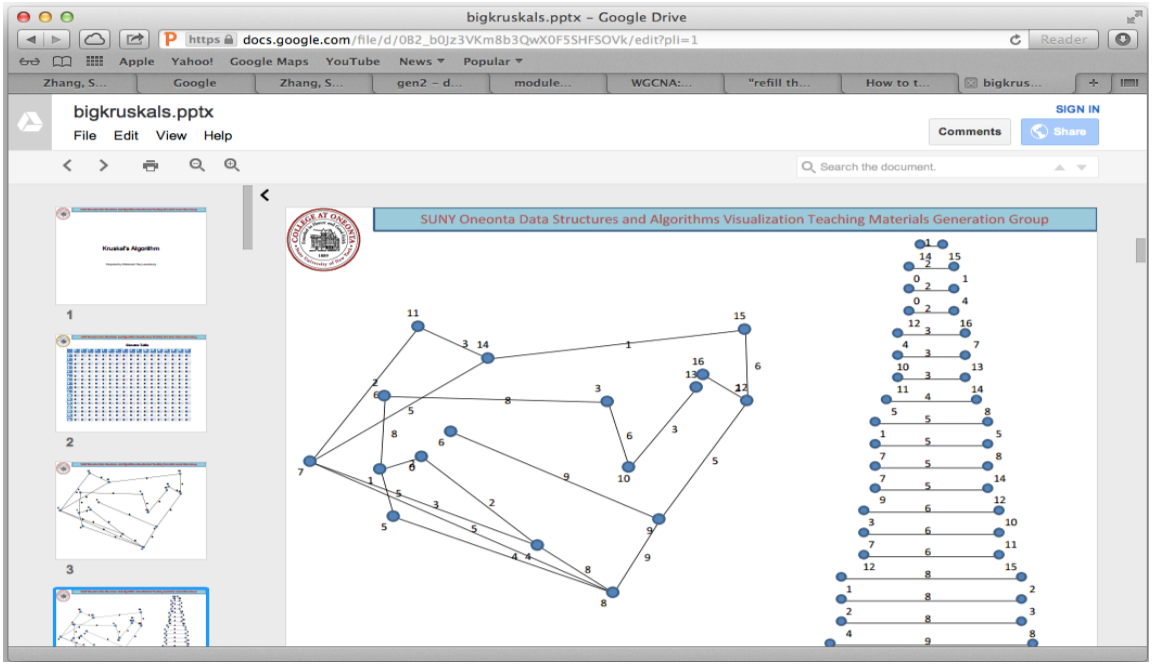


Fig. 1. A snapshot of auto-generated slides showing Kruskals' algorithm in MS PowerPoint viewer.

B. Using the Generator

The generator is coded and debugged once by competent programmers, but can be run by many instructors. We also consider an IO module to our generators so that we can reuse the generator executables to create slides for different datasets. Terminal users do not need to learn how to modify and recompile the generators; instead only need to describe the input (data and operation sequence or the combination of the two) in intuitive XML or plain text file formats. Using binary search tree (BST) as an example, the input file typically describes a set of action-data pairs (insert, delete, update certain numbers). Optionally, a BST input file can also specify traversal methods, e.g. breadth first traversal, and depth first traversal and request statistics of the trees. Of course, the description needs to conform to the interface implemented by the generator so that the generator knows how to interpret the input data properly. These input files, when fed to the generators, will drive the generators to produce slides for illustrating how the algorithms work on the input data. The slides then can be presented in the MS PPT. This process is shown in Fig. 2.

Running a generator might be an extra burden to some instructors. However, most instructors won't even need to go through this generation process, if they are willing to adopt the slides created by other instructors who have already used our generators. Currently, instructors can send the dataset to the author who will be happy to generate slides for them. To further simply this process, the author has also been exploring the second level automation by building a cloud-based platform. Ideally, terminal instructor users only need to prepare their datasets and submit to the platform; then our server will dispatch the request to the right generator to create the slides on the cloud or the server. Once the slides are completely created, the instructors will be notified to download t for their instruction purpose. The technical details

and the status of this higher level automation will be reported in a separate paper.

C. Preparedness Pedagogy

Since our approach clearly separates offline generation and online presentation, it greatly supports preparedness pedagogy allowing instructors to use offline time effectively. In other words, the instructors have more control in preparing and rehearsing all the possible cases for the interested algorithm in advance so that they can efficiently use classroom time focusing on presenting the materials rather than generating the materials.

Using Binary Search Tree (BST) as an example, before an instructor teaches the topic, our approach allows she/he to plan the data set (e.g. numbers) and operation sequence (insertion, deletion, update, traversal etc.) to try to create all cases (e.g. leaf node, internal node having a single child, and the internal node having both children are three cases of BST deletion). Then she/he can use the datasets and the commands to run the BST generator to generate slides. After that, the instructor can review the slides to ensure that all the cases have been properly covered. If any important case is missing, the instructor can refine the input data and then regenerate the slides. This process may be iterated a couple of rounds until the instructor feels completely satisfied with the datasets, the algorithm cases and the overall visualization.

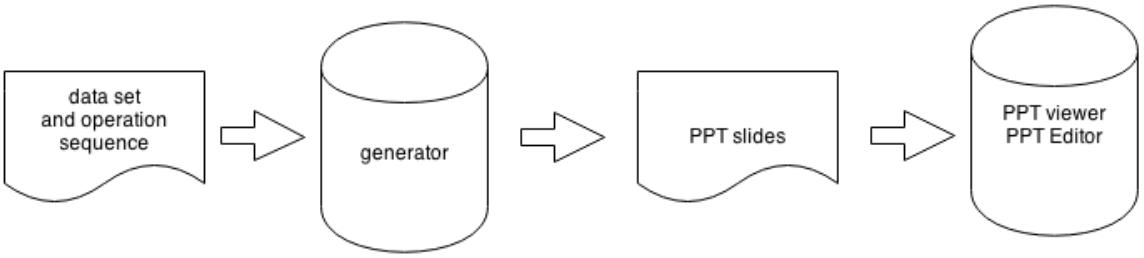


Fig. 2. Use of generator.

TABLE I: ADVANTAGES OF THE PROPOSED APPROACH COMPARED WITH MANY OTHER SOLUTIONS

Feature	The proposed Approach	Other Approaches
Presentation oriented	Yes	Usually Not
Standardized	Yes	Usually Not
Industry Level Support	Yes	Usually Not
Easy to Adopt	Yes	Usually Not
Customizable/Editable	Yes	Usually Not
Separable from generators/Light-weighted	Yes	Usually Not
Built-in Navigation	Yes, Built-in Support	Usually Not
Teaching Friendly	Yes	Usually Not
Printable	Yes	Usually Not
Learning Curve	Low, Almost Zero	Usually Sharp
Sustainability	High, Once Generated, Detached as Files	Usually Low

Then the instructor can rehearse her/his lecture using the presentation possibly with other materials and time her/his teaching pace of the lecture with using the rehearsed slides and other traditional materials. Optionally, the instructor may also want to annotate some and all slides by hand.

All the above activities can effectively be conducted using outside the classroom time to save classroom time on more critical issues of the DSAs under discussion such as on more theoretical aspects of the algorithm, implementation tricks, time and space complexities etc. If used properly, our approach can help the instructor to gain better control on the classroom, improve efficiency of their teaching, and avoid unnecessary glitches due to using random dataset created on the fly by the students. From the learning perspective, these slides can also be easily shared with the students for the reviewing purpose or self-paced learning. Due to the popularity of the MS PPT, these slides can be viewed on most computers and mobile devices.

III. ADVANTAGES AND DISADVANTAGES OF THE APPROACH

Our PPT based solution has a variety of advantages. First, PPT is standardized. The slides can be seamlessly integrated into other PPT materials. Second, PPT poses nearly no learning curve and has achieved ubiquitous adoption in classroom practice. It is detachable and extremely self-sustainable, persistent. Files can be easily edited, annotated, printed, customized and extended. Third, PPT files are light-weighted. They are easy to download, save, copy, play and replay anywhere. Fourth, PPT slides can easily be converted to other formats such as PDF, HTML, or others via proper third-party plug-ins. Fifth, the whole project is highly scalable, and each generator is perfectly independent from others. Table I summarizes the advantages of the proposed approach by comparing it with many other previous approaches.

On the other side, the proposed approach also has certain disadvantages or limitations. First, we recognize that individual generators need to be coded for each unique algorithm, which is a very consuming process and coding intensive. This is especially for challenging and often needed algorithms. Second, the approach isn't a universal solution for every instructor. For example, for instructors who resist PPT, our approach won't help. Third, the approach can be further refined and enhanced. For example, some reviewers of the project suggested a feature to step through pseudo-code of the algorithm on the slides to synchronize with the visualization. Fourth, the method is mainly a practical solution for which the theoretical foundation may need to be further studied. There might be also some other

restrictions and weaknesses of the proposed approach that we haven't observed.

IV. Current Status of Piloting the Idea

We have pilot coded generators [10] for a dozen of frequently taught algorithms including heap-sort, binary search tree, AVL tree, Red-black tree, shortest path, longest common string, magic square, etc. We also shared the generated sample slides on the Internet with the community through a website temporarily hosted by the author's working website [11]. For a couple of algorithms, we even added external links to their corresponding Wikipedia pages in order to reach out to the instructors and students.

Fig. 1. Distribution of global visits of the auto-generated materials.

Fig. 3 shows the distribution of the web visits to the auto-generated materials produced by our pilot project. From the figure, we can tell that the materials are interesting to many others from all over the world. Table I and Table II show the breakdowns of these web hits by countries and U.S. states. It needs to be pointed out that since the project was developed in NY, the developing group has contributed to a significant number of web hits. Please also notice that Table II shows that the number of hits roughly is related to the number of higher education institutions (the number of college students). Also the states on the top of the list are relatively IT industry heavy. Table III probably is biased with respect to the countries where English is used as academic language.

TABLE II: Number of Web Visits of the Project Website from Difference States of U.S. (Top 7 out of 46 States Where the Internet Surfers Visiting the Repository)

Rank	State	Web hits	Num of Institutions
1	New York (NY)	738	633
2	California (CA)	339	1246
3	Virginia (VA)	115	222
4	Washington (WA)	57	164
5	New Jersey (NJ)	43	207
6	Massachusetts(MA)	41	261
7	Pennsylvania	33	544

We have also conducted a small-scale case study to evaluate the effectiveness of the approach. Four computer science instructors from three institutions at NY, IN and NJ have pilot used the materials to facilitate their teaching of data structures and algorithms (DSAs) [12]. The instructors found the new materials generated using our approach helpful in their teaching. Initial feedbacks from students are also consistently positive, indicating that undergraduate students find the auto-generated slides accessible, convenient, and useful in learning DSAs. More detailed information about the evaluation will be reported in a separate manuscript that is currently under draft.

TABLE III: Number of Web Visits of the Project Website from Difference Countries (Top 8 out of about 200 Countries Where the Internet Surfers Visited the Repository)

Rank	Country	Web hits
1	U.S.A.	1889
2	India (IN)	790
3	Germany (DE)	122
4	China (CN)	115
5	Canada (CA)	108
6	Brazil (BR)	103
7	United Kingdom(GB)	100
8	Poland (PL)	92

V. Conclusion

In this paper, we have presented a novel and practical solution to create accessible DSA visualization slides by taking advantage of powerful MS PPT API. PPT slides are editable, savable, light-weighted, standardized and instructor-friendly. The approach also provides instructors to adopt preparedness pedagogy. We anticipate that the proposed approach can benefit many CS instructors by facilitating their teaching of DSAs and the learning of even more students.

Having been mentioned, the proposed approach is coding intensive and expert labor expensive. In the future, we plan to seek resources for developing new generators for more advanced algorithms and refining the existing generators. Furthermore, we will further automate the whole process to enable effortless adoption of the approach by implementing a cloud based platform. We will also try to pre-generate multiple slides using representative datasets to create slides to share with the DSA teaching and learning community and to expand the evaluation and assessment activities on the approach by working with a relatively larger group of DSA instructors from different institutions to collect more evidences of the usefulness of the approach and support the DSA teaching and learning community.

Acknowledgment

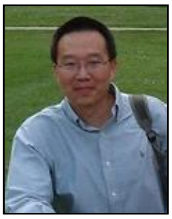
The project prototyping the proposed approach was partially supported by State University of New York (SUNY) Innovative Instructional Technology Grant (IITG) and SUNY College Oneonta. The author also thanks the TLTC and the grant office at SUNY College at Oneonta, as well as his colleagues and students for their constant support to the project. Finally, the author thanks the reviewers of the paper for their very constructive suggestions to improve the presentation of the paper.

References

[1] ACM/IEEE-CS Joint Task Force on Computing Curricula, *Computer Science Curricula 2013*, ACM Press and IEEE Computer Society Press, December 2013.

[2] T. Chen and T. Sobh, "A tool for data structure visualization and user-defined algorithm animation," in *Proc. 2001. 31st Annual Frontiers in Education Conference*, Reno, NV, vol. 1, pp. TID-2, 2001.

[3] C. Kehoe, J. Stasko, and A. Taylor, "Rethinking the evaluation of algorithm animations as learning aids: An observational study," *International Journal of Human-Computer Studies*, vol. 54, no. 2, pp. 265-284, Jan 2001.
[4] M. Munk, "How helpful are systems for algorithm visualization?" Technical Report, Technische Universität München, April 3, 2003.
[5] C. A. Shaffer *et al.*, "Algorithm visualization: The state of the field," *ACM Transactions on Computing Education*, vol. 10, no. 3, August 2010.
[6] M. H. Lee and G. Rößling, "Toward replicating handmade algorithm visualization behaviors in a digital environment: A pre-study," in *Proc. 16th Annual Joint Conference on Innovation and Technology in Computer Science Education*, pp. 198-202, New York, NY, USA: ACM, 2011.
[7] T. L. Naps, "JHA VE: Supporting algorithm visualization," *IEEE Computer Graphics and Applications*, vol. 25, no. 5, pp. 49-55, August 2005.
[8] J. T. Stasko, "A framework and system for algorithm animation," Technical Report, Brown University, 1989, Providence, RI, USA.
[9] S. Zhang and J. Ryder, "Building a generator-based cyber platform for automating production of PPT-Based algorithm visualization teaching materials," presented at the 42nd ACM Technical Symposium on Computer Science Education (SIGCSE), March 9-12, 2011, Dallas, Texas, USA.
[10] S. Zhang and J. Ryder, "Pedagogical practices and benefits of using auto- generation approach to facilitate teaching data structures and algorithms," *Journal of Computing Sciences in Colleges*, vol. 28, no. 6, pp. 203-204, June 2013.
[11] S. Zhang. A pilot website of the Auto DSA PPT project and a repository of PPT based Auto-generated DSAs teaching slides materials. [Online]. Available: http://employees.oneonta.edu/zhangs/powerpointplatfom.
[12] D. H. Wei and S. Zhang, "Using auto-generated materials to facilitate instructors' offline preparation and improve students' learning outcomes," *Journal of Computing Sciences in Colleges Archive*, vol. 29, issue 6, pp. 151-152, June 2014.



Sen Zhang received the B.S., M.S. and Ph.D. degrees all in computer science, respectively from Tian Jin University, China, in 1992, the South China University of Science and Technology, Guangzhou, China in 1995, and the New Jersey Institute of Technology (NJIT), NJ, U.S.A. in 2005.

He joined the Department of Mathematics, Computer Science and Statistics, College at Oneonta, State University of New York (SUNY), as an assistant professor in 2004, and became an associate professor in 2010. His current research interests include data mining, database design and algorithm design. He has abundant teaching experience and is also interested in computer science education and innovative instructional research. His work has received support from SUNY Innovative Instruction Technology Grants program and SUNY Oneonta faculty development grants.

Dr. Zhang is a member of IACSIT, ASE, CE21 Community and ACM CSTA.