# OPT+Graph: Web-Based Program Visualization for Understand Code Graph Execution in CS Education

Habibie Ed Dien
School of Electrical Engineering and Informatics
Bandung Institute of Technology
Bandung, Indonesia
habibieeddien@students.itb.ac.id

Yudistira Dwi Wardhana Asnar
School of Electrical Engineering and Informatics
Bandung Institute of Technology
Bandung, Indonesia
yudis@informatika.org

***Abstract*— This paper presents a web-based program visualization tool for C/C++ as an approach to understand code graph execution. This tool is based on pythontutor.com (OPT). We examine three fundamental questions in program visualization – how to define understanding, how to build an effective visualization tool, and how to detect graph in the C/C++ code execution. The main contribution of this paper is a visualization code graph execution. Method of performance measurement that used is evaluation of visualization. The technique used a survey through an online questionnaire with four stages, the first stage is filling biodata of respondents, the second stage is completing the pretest, the third stage is a simulation using OPT and OPT+Graph. The last stage is the respondent completed the post-test. The subjects of this survey are undergraduate and postgraduate informatics students at Bandung Institute of Technology. This research's results are: 1) the visualization approach can be an effective and efficient tool for understand code graph execution in C/C++; 2) usability is one of the important aspects of visualization; and 3) OPT+Graph is free and open source software, available at codeviz.tk/codeviz.**

***Keywords—web-based tool, program visualization, understand, code graph execution***

## I. INTRODUCTION

The process of learning programming cannot be separated from doing program code execution. For some students, learning programming is not easy. In addition to understanding the algorithm, constructing program code is an important part for the implementation of the algorithm that has been designed.

Sometimes the teachers when explain the process of program code execution in classroom still using a whiteboard or Powerpoint slides. This requires extra preparation such as drawing, flowcharts, parts of program code related to the material to be explained, especially if the programming material goes to more complex level, such as data structure. So, to explain the process of program code execution more better, it required a particular learning media, so that students can understand the process actually happened in the computer program.

A study that has been conducted by Piteira and Costa [17] at the Setubal Polytechnic Intitute, Portugal, found that the concept of data structure programming has a high level of difficulty for most students. This is because the concept of data structure programming is a consept of abstract data which is less understood by students if written in program code form. The study also paid special atttention to the concept of data structure programming such as graph, pointer, parameter, and abstract data types (ADT).

Visualization as a programming learning media is not a new thing. The researchers have developed visualization tools to help students in learning algorithm and programming [4, 9, 10, 22, 23]. Because from the visual sense, humans more tend capture more information obtained than through other sense [28]. Visualization can support efficient and effective interaction for a variety of cognitive task such as analyzing, summarizing, and taking conclusions on the information obtained.

Now internet and web technologies provide easy access to share information [1]. Web application continue to evolve from simply delivering static contextual information, so that impact to sophisticated presentation of visualization information dynamycally. Technologies such as *Java3D*, *VRML*, *X3D*, and *SVG* have powerful rendering capabilities, but it difficult to interact with raw data sources [13].

The development of visualization tools program code execution or with the term Program Visualization (PV) web-based for graph is still very rare [22, 23]. Sorva learned comprehensively the development about fourty PV tools [22]. Most of the tools use *Java Applet* to be able to operate on the web. While the *Java Applet* still need to be installed and configured on browser, so this didn't leave ease of access to use.

Guo have developed PV tool named *Online Python Tutor* (OPT) [10]. This web-based tool has embeddable features easy to use. That features are used to attach visualization on other web. This tool using *library D3.js* [2] as the main technology to support visualization. However, OPT has not been developed for visualization graph data structure.

In this research, it has been developed tool based on the source code of OPT. In addition has supported web technologies, this tool is free and open source [10]. The results of this tool's development is expected to facilitate the students to understanding about program code execution that has graph data representation.

Based on background that has been described above, these are the formulation of problem in this reseach: (1) What is understanding graph execution of program code? (2) Is with graph visualization can effectively to understand graph execution of program code? (3) How is the techniques to detecting graph on the graph execution of program code so can be visualized?.

## II. Related Work

Software visualization is one of the active field in research and development of the system. There are so many software visualization system appear to be used with a particular purpose and continue growing every year [22]. Gračanin et al. define the software visualization as a field to investigate with approaches and certain techniques that aims to represent graphics algorithm staticly or dynamicly, program (code), and data processed [9]. Software visualization has the main purpose to analyzed program and development; to improve the understanding of invisible concept and how software works. The main challenge is seeking an effective step in mapping various aspects of software to be represented graphically using visual metaphor. In other words, software visualization not focuses on program construction process, but more to program analysis and software development process.

The term of "Software Visualization" defined as the use of an art of typography, graphic design, animation, and cinematography through modern interaction between human-computer and computer graphic technology as a means of understanding the effectiveness of computer software use [4]. Software visualization is divided into two, namely Algorithm Visualization (AV) and Program Visualization (PV) [4, 22,23]. AV is related to abstraction of algorithms, concepts and work steps of a software, while PV is related to the works of program code execution and data structure process.

Based on the timeline of PV development [23], there are only four PV tools that have been developed for graph, namely: Swan [21], *VisMod* [14], jGRASP [5], *Jype* [11], and one PV tool web-based (HTML, CSS, and JS) that is *Online Phyton Tutor* or OPT [10]. Swan is PV tool for data structure and C/C++ code execution. This tool can visualized graph, tree, list, and array. Methods using the annotation in the program code called SAIL (*Swan Annotation Interface Library*). The main purpose of Swan tool is to create the annotation library that easy to be used in visualization. VisMod can visualized program code execution with *Modula-2* programming languange. This tool support linier and tree data structure visualization. Methods by reading pointer variable data and reference. In addition, this tools can check syntax errors and make sure the use of all variable that has been declared before. jGRASP is a visualization tool of program code execution that support Java language. This tool can visualized binary tree and linked list. Visualization based on pointer data type represented as node and reference as edge. Jype used Matrix Framework [15] to visualized data structure such as the array and tree automatically when detected in Phyton program code. Jype is developed with Java which can operate on web as Java Applet. To activated it, all the componens in application must be downloaded, then the application can operate on the computer.

Online Phyton Tutor or OPT [10] is PV tool that is free and open source. This tool has many features that support a wide range of programming language, such as Phyton, Java, C, C++, Ruby, Javascript, Typescript and still continue to be developed[1]. When this research done, OPT has grown on version 5 released on July 27, 2016. Active research done on the version 5 by using Typescript programming languange as a basis of OPT tool development.

Based on the literature and exploration that has been done, most of PV tool made for dekstop computer or Java Virtual Machine (JVM). Java Applet application can operate on the web, but Java configuration on browser must be match with what is needed. This is certainly different with web application such as HTML, CSS, and Javascript that can be directly used in the browser. User do not need to configure Java to access it.

The exploration results obtained that there is no single PV tools web-based with C or C++ languange that can display graph visualization. So this becomes an opportunity for the next research in its development. Especially the use of Typescript as a basic of new tool development and has a future prospect in the latest web technology. OPT is chosen to be the base of tool's development, because in addition to supporting the web base, this tool also free and open code sourced. This tool is also still in the process of development and research by its developers. So, there are so many opportunities and gaps for further research in order to be used well. Remembering that web technology and mobile device applications continue to grow rapidly until the last decade.

## III. Design of Visualization

A graph is described by node and edge. Node and edge have attributes respectively. Node's attributes consists of shape, color, size, and label. While edge's attributes consists of the thickness of line, label, color, and arrow (for directed graph). The picture of basic graph visual with node and edge can be seen on Figure 1 following.

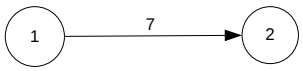
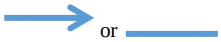
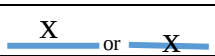
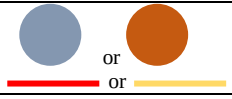



Fig. 1. Directed graph visual with two nodes and edges

Graph visualization have some properties that are need. Table 1 shows graph visual property that can be represented. From that list, it is not all visual property used, it should be able to customize with data processing from program code execution.

TABLE I. LIST OF GRAPH VISUAL PROPERTY

| Number | Property's Name | Visualization |
|---|---|---|
| 1 | Label for node |   |
| 2 | Node's shape |  |
| 3 | Edge's shape |   |
| 4 | Edge's label |   |
| 5 | Node and Edge's color |   |
| 6 | Edge's thickness |   |

To analyze the detection technique of program code graph, it neccessary to know in advance how the representation of graph data in the program code so that the

[1] https://github.com/pgbovine/OnlinePythonTutor

source for visualization can be determined. Graph data representation in the program code can be done in three ways, as follows [20]: a) Adjacency matrix representation or matrix data; b) Array of edges representation, that is using edge class with two variables of int type; and c) Array of adjacency lists representation, this uses a node index that store list from adjacent nodes.

## IV. RESULTS

The developed tool focused on graph visualization of program code execution. The purpose of tool's development is to add graph visualization feature on OPT tool, so it can be used to understand graph execution of program code. Figure 2 shows the general scheme of tool's development results. The input is in the form of C/C++, if there is a graph representation, so the output can adjust to display graph visual.

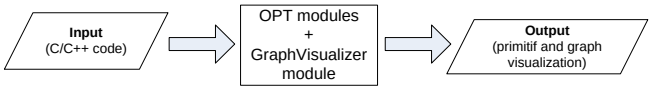



Fig. 2. The general scheme of tool development results

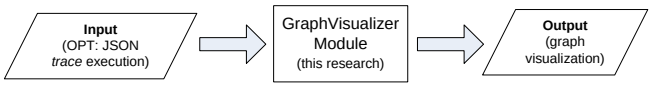



Fig. 3. The general sheme of graph visualization modul (*GraphVisualizer*)

Figure 3 shows about specific and general scheme for graph visualization class. Actually the input of class processed is JSON data, not a program code. JSON data that has graph representation, will be extracted and the output can adjust to display graph visualization or not.

Graph visual representation can through the process as in Figure 4. A "model matching" process is find which match of graph data representation in code. The next process is visualization using library D3.js.

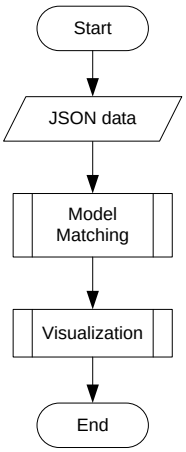



Fig. 4. Level-0 flowcharts of data visualization process

Figure 5 shows user interface to input C/C++ program code in Chrome. User can choose the example of program code in the bottom of button "Visualize Execution". User also can type program code in editor code that has been provide. Visualization results will appear as in Figure 6. User can explore program code execution with control navigation (Figure 6.(a)) that available, or can use slider (Figure 6.(b)) to look directly on the step desired.

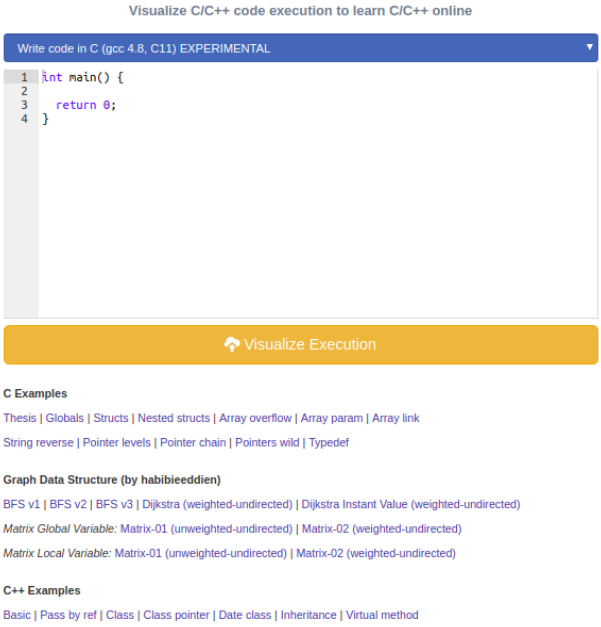



Fig. 5. User interface's implementation to input program code

Figure 6 (c) is a panel consists of three visualizations, namely "Primitif Visualization" is to display variable data of program code simply in a form stack and heap. The "Graph Visualization" panel is the main part of graph visualization developed in this research. The "Print Output" panel is to display output standard that come from printf command in program code.

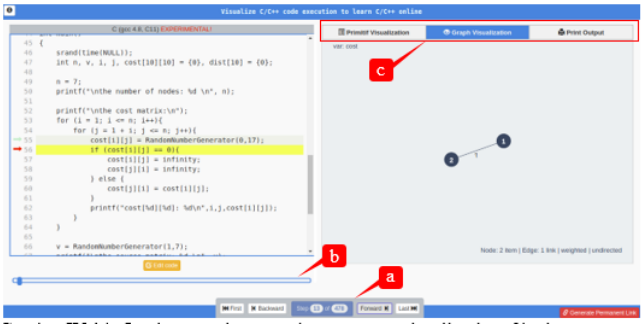



Fig. 6. Implementation of user interface graph visualization of program code; (a) Control navigation; (b) Slider; (c) Visualization panel

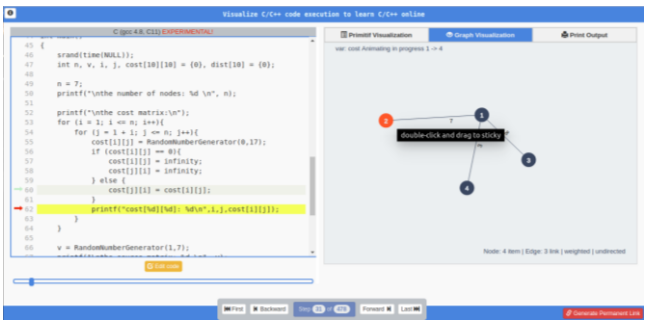


Fig. 7. Weighted and non direct of graph visualization implementation

Figure 7 shows tool implementation for weighted and non direct of graph visualization with matrix data

representation. Node can accept drag and drop action from user to be positioned as desired by user.

Figure 8 shows tool implementation for weighted and direct of graph visualization with matrix data representation. Edge will automatically display the arrows if a variable data is detected that contains directed graph. Numbers label for the weight of each edge also will automatically appear when detected by weight variables between nodes.

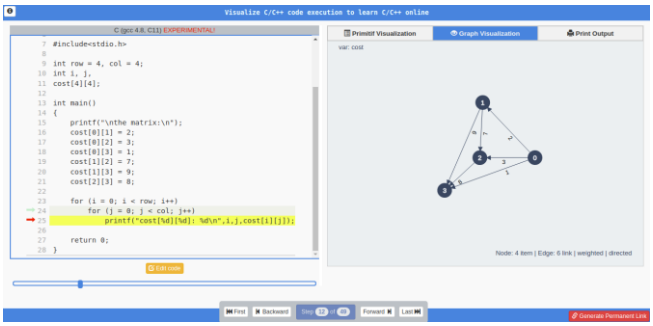


Fig. 8. Weighted-direct of graph visualization implementation

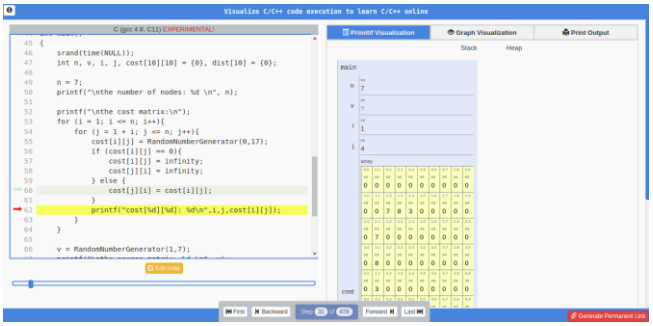


Fig. 9. "Primitif Viualization" panel implementation

Figure 9 and 10 is an implementation for "Primitif Visualization" and "Print Output" panel that actually available in previous OPT tool. Just place it on the panel to make easier for user to access it.

Figure 11 shows the implementation of animation feature to route search between nodes. The letter (a) on Figure 11 is route search animation from node 2 to node 1. The letter (b) on Figure 11 is the continuation of the route search animation from node 2 to node 3.

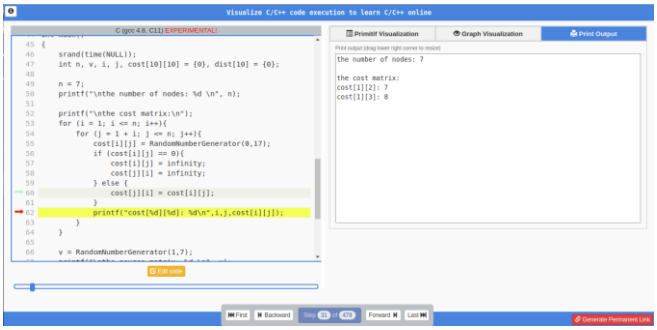


Fig. 10. "Print Output" panel implementation

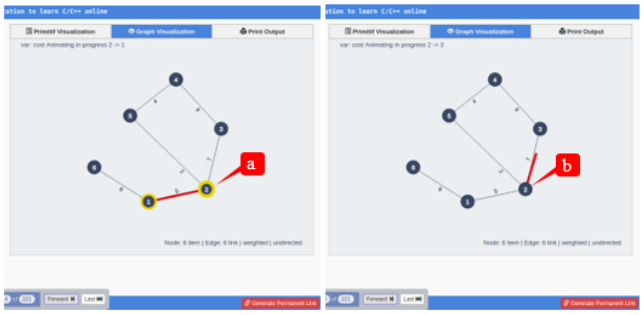



Fig. 11. Feature implementation of route search animation between nodes

## V. FUTURE WORK

Based on the results of research that has been done, it is known that visualization has a well enough effectiveness for user in understand graph execution of program code prosess. But sometimes still constrained by the slow process of compiling the program code to be a vizualization. These constraints can reduce tool interactivity. A hope for future research can focused to compile program code using the resources in each user's computers in the browser. So the compilation process not burden the work of server computers.

## VI. CONCLUSION

Some of the conclusion that has been obtained during this research are as follows:

1) Visualization approach can be an effective and efficient tool to understand graph execution of program code;
2) Usability is one of the important aspects of visualization, because it is related to tool's ease of use. The tools must be able to help human's activity, not otherwise become more complicated and difficult to be used.
3) Typescript (TS) is Javascript (JS) programming language based on object that promises enough to develop application or web-based tools which is interactive big scale. The complexity of JS-based program is strongly supported by TS.

## REFERENCES


[1] C.J. Bonk, The world is open: how Web technology is revolutionizing education, 1st ed, San Francisco, Calif, Jossey-Bass, 2009.

[2] M. Bostock, V. Ogievetsky, and J. Heer, D3: Data-Driven Documents, *IEEE Trans. Vis. Comput. Graph.*, vol. 17, pp. 2301–2309, 2011.

[3] D. Bruening and Q. Zhao, Practical memory checking with Dr. Memory. In *Proceedings of the 9th Annual IEEE/ACM International Symposium on Code Generation and Optimization*, IEEE Computer Society, pp. 213–223, 2011.

[4] I. Cetin and C. Andrews-Larson, Learning sorting algorithms through visualization construction, *Comput. Sci. Educ.*, vol. 26, pp. 27–43, 2016.

[5] J.H. Cross II, T.D. Hendrix, J. Jain, and L.A. Barowski, Dynamic object viewers for data structures, *ACM SIGCSE Bulletin*, ACM, pp. 4–8, 2007.

[6] S. Diehl, Software visualization: visualizing the structure, behaviour, and evolution of software ; with 5 tables, Berlin, Springer, 2007.

[7] B. Fry, *Visualizing Data*, First Edition, USA, O`Reilly Media, Inc, 2008.

[8] P. Gestwicki and B. Jayaraman, Methodology and architecture of JIVE, *Proceedings of the 2005 ACM symposium on Software visualization*, ACM, pp. 95–104, 2005.

[9] D. Gračanin, K. Matković, and M. Eltoweissy, Software visualization, Innov. Syst. Softw. Eng., vol. 1, pp. 221–230, 2005.

[10] P.J. Guo, Online python tutor: embeddable web-based program visualization for cs education, Proceeding of the 44th ACM technical symposium on Computer science education, ACM, pp. 579–584, 2013.

[11] J. Helminen, and L. Malmi, Jype-a program visualization and programming exercise tool for Python, *Proceedings of the 5th international symposium on Software visualization*, ACM, pp. 153–162, 2010.

http://dl.acm.org/citation.cfm?id=1879234

[12] T.D. Hendrix, J.H. Cross II, and L.A. Barowski, An extensible framework for providing dynamic data structure visualizations in a lightweight IDE, *ACM SIGCSE Bulletin*, ACM, pp. 387–391, 2004.

[13] N. Holmberg, B. Wünsche, and E. Tempero, A framework for interactive web-based visualization, *Proceedings of the 7th Australasian User interface conference-Volume 50*, Australian Computer Society, Inc., pp. 137–144, 2006.

http://dl.acm.org/citation.cfm?id=1151778

[14] R. Jiménez-Peris, M. Patiño-Martínez, and J. Pacios-Martínez, VisMod: a beginner-friendly programming environment, ACM Press, 115–120, 1999.

http://portal.acm.org/citation.cfm? doid=298151.298218

[15] A. Korhonen, L. Malmi, P. Silvasti, V. Karavirta, J. Lönnberg, J. Nikander, K. Stålnacke, and P. Tenhunen, *Matrix - a framework for interactive software visualization*, Research Report TKO-B 154/04, Department of Computer Science and Engineering, Helsinki University of Technology, pp. 26–35, 2004.

[16] J. (n.d.) Nielsen, Usability First - Methods - Heuristic Evaluation | Usability First, *Heuristic Evaluation*.

http://www.usabilityfirst.com/usability-methods/heuristic-evaluation

[17] M. Piteira and C. Costa, Learning computer programming: study of difficulties in learning programming, *Proceedings of the 2013 International Conference on Information Systems and Design of Communication*, ACM, pp. 75–80, 2013.

http://dl.acm.org/citation.cfm? id=2503871

[18] J. Preece, Interaction Design: Beyond Human-Computer Interaction, First Edition, First Edition, USA, John Wiley & Sons, Inc, 2002.

[19] J. Preece, H. Sharp, and Y. Rogers, *Interaction Design, beyond human-computer interaction Fourth Edition*, 4th edition, John Wiley & Sons, Ltd, 2015.

[20] R. Sedgewick and K. Wayne, *Algorithms Fourth Edition*, Fourth Edition, USA, Pearson Education, Inc, 2011.

[21] C.A. Shaffer, L.S. Heath, and J. Yang, Using the Swan data structure visualization system for computer science education, *ACM SIGCSE Bull.*, vol. 28, pp. 140–144, 1996.

[22] J. Sorva, *Visual program simulation in introductory programming education*, Aalto University publication series Doctoral dissertations, Espoo, Aalto Univ. School of Science, 2012.

[23] J. Sorva, V. Karavirta, and L. Malmi, A review of generic program visualization systems for introductory programming education, ACM Trans. Comput. Educ. TOCE, vol. 13, 15, 2013.

[24] V.M. Sue and L.A. Ritter, *Conducting online surveys*, United States of America, Sage Publications, 2007

[25] E. Thompson, A. Luxton-Reilly, J.L. Whalley, M. Hu, and P. Robbins, Bloom's Taxonomy for CS Assessment, ACE '08, Wollongong, NSW, Australia, Australian Computer Society, Inc., pp. 155–161, 2008.

https://dl.acm.org/citation.cfm?id=1379265

[26] J. Urquiza-Fuentes and J.Á.Velázquez-Iturbide, A Survey of Successful Evaluations of Program Visualization and Algorithm Animation Systems, *ACM Trans. Comput. Educ.*, vol. 9, pp. 1–21, 2009.

[27] J.Á. Velázquez-Iturbide, I. Hernán-Losada, and M. Paredes-Velasco, Evaluating the Effect of Program Visualization on Student Motivation, *IEEE TRANSACTIONS ON EDUCATION*, 2017.

[28] C. Ware, Information visualization: perception for design, 2nd edition, San Francisco, Kanada, Elsevier Inc, 2004.