**OPT+Graph: Web-Based Program Visualization**

**for Understand Code Graph Execution in CS Education**

23515043 – Habibie Ed Dien

Pembimbing: Yudistira Asnar

Institut Teknologi Bandung

**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

1. **Introduction**

Proses belajar pemrograman tidak lepas dari melakukan eksekusi kode program. Belajar pemrograman bagi sebagian peserta didik tidak mudah. Selain pemahaman tentang algoritma, kegiatan mengonstruksi kode program adalah bagian penting untuk implementasi dari algoritma yang telah dirancang.

Bagi pengajar untuk menjelaskan proses eksekusi kode program di kelas terkadang menggunakan papan tulis atau slide PowerPoint. Hal ini membutuhkan persiapan ekstra seperti gambar, alur diagram, atau bagian-bagian kode program terkait materi yang akan dijelaskan. Terutama jika materi pemrograman itu masuk ke tingkat yang lebih rumit, seperti struktur data. Untuk dapat menjelaskan proses eksekusi kode program tersebut diperlukan media belajar khusus, sehingga peserta didik dapat memahami proses yang sebenarnya terjadi di dalam program komputer.

Sebuah penelitian yang telah dilakukan (Piteira dan Costa, 2013) di Institut Politeknik Setubal, Portugal, menemukan bahwa konsep pemrograman struktur data memiliki tingkat kesulitan yang tinggi bagi sebagian besar peserta didik. Hal itu disebabkan konsep pemrograman untuk struktur data merupakan konsep data abstrak, yang kurang dipahami oleh peserta didik jika ditulis dalam bentuk kode program. Penelitian tersebut juga memberikan perhatian khusus terhadap konsep pemrograman struktur data seperti graf, pointer, parameter, dan abstract data types (ADT).

Visualisasi sebagai media belajar pemrograman bukan suatu hal yang baru. Para peneliti telah banyak mengembangkan kakas visualisasi untuk membantu mempelajari algoritma dan pemrograman (Cetin dan Andrews-Larson, 2016; Sorva, 2012; Sorva dkk., 2013; Gračanin dkk., 2005; Guo, 2013). Karena melalui indera visual, manusia lebih cenderung menangkap lebih banyak informasi yang diterima dibandingkan melalui indera lainnya (Ware, 2004). Visualisasi dapat mendukung interaksi yang efisien dan efektif untuk beragam pekerjaan kognitif seperti menganalisis, meringkas, dan menarik kesimpulan atas informasi yang diperoleh.

Saat ini teknologi internet dan web memberikan kemudahan akses untuk berbagi informasi (Bonk, 2009). Aplikasi web terus berkembang dari sekadar penyampaian informasi kontekstual statis, sehingga berdampak terhadap kemutakhiran penyajian visualisasi informasi secara dinamis. Teknologi seperti *Java3D*, *VRML*, *X3D*, dan *SVG* memiliki kemampuan *rendering* yang *powerful*, tetapi sulit untuk berinteraksi dengan sumber data mentah (Holmberg dkk., 2006).

Perkembangan kakas visualisasi eksekusi kode program atau dengan istilah visualisasi program (VP) berbasis web untuk graf masih sangat langka (Sorva, 2012; Sorva dkk., 2013). Sorva (2012) menelaah secara komprehensif perkembangan 40 kakas VP. Sebagian besar kakas menggunakan *Java Applet* untuk dapat beroperasi di web. Sedangkan *Java Applet* masih perlu di-*install* dan konfigurasi pada *browser*, sehingga ini tidak memberikan kemudahan akses untuk menggunakannya.

Philip Guo (2010) telah mengembangkan kakas VP bernama *Online Python Tutor* (OPT). Kakas berbasis web ini memiliki fitur *embeddable* yang mudah digunakan. Fitur tersebut digunakan untuk melampirkan visualisasi di halaman web lain. Kakas ini menggunakan *library* *D3.js* (Bostock dkk., 2011) sebagai teknologi utama untuk mendukung visualisasi. Namun, OPT belum dikembangkan untuk visualisasi struktur data graf.

Pada penelitian ini, dilakukan pengembangan kakas berdasar pada kode sumber dari OPT. Selain telah mendukung teknologi web, kakas ini bersifat bebas (*free*) dan bersumber terbuka (*open source*) (Guo, 2013). Hasil pengembangan kakas ini diharapkan dapat memudahkan peserta didik untuk memahami eksekusi kode program yang terdapat algoritma graf.

Berdasarkan latar belakang yang telah diuraikan tersebut, diperoleh rumusan masalah sebagai berikut:

1. Apakah yang dimaksud dengan memahami eksekusi graf kode program ?
2. Apakah dengan visualisasi graf dapat efektif untuk memahami eksekusi graf kode program ?
3. Bagaimana teknik untuk mendeteksi graf pada eksekusi graf kode program sehingga dapat divisualisasi ?
4. **Related Work**

Visualisasi perangkat lunak (Software Visualization) adalah salah satu bidang aktif dalam riset dan pengembangan sistem. Berbagai sistem visualisasi perangkat lunak bermunculan untuk digunakan dengan tujuan tertentu dan terus berkembang setiap tahunnya (Sorva, 2012). Gračanin dkk., (2005) mendefinisikan visualisasi perangkat lunak (Software Visualization) sebagai suatu bidang untuk menginvestigasi dengan pendekatan dan teknik tertentu yang bertujuan dalam merepresentasikan grafis algoritma secara statis atau dinamis, program (code), dan data yang diproses. Visualisasi perangkat lunak memiliki tujuan utama untuk menganalisis program dan pengembangan; untuk meningkatkan pemahaman terhadap konsep yang tak nampak dan cara kerja perangkat lunak. Tantangan utamanya adalah mencari langkah efektif dalam pemetaan berbagai aspek perangkat lunak untuk direpresentasikan secara grafis menggunakan metafora visual. Dengan kata lain, visualisasi perangkat lunak tidak fokus terhadap proses konstruksi program, akan tetapi lebih kepada analisis program dan proses pengembangan perangkat lunak.

Istilah “visualisasi perangkat lunak” didefinisikan sebagai penggunaan sebuah seni tipografi, desain grafis, animasi, dan sinematografi melalui interaksi modern antar manusia-komputer dan teknologi grafis komputer sebagai sarana untuk memahami dan keefektivan penggunaan perangkat lunak komputer (Cetin dan Andrews-Larson, 2016). Visualisasi perangkat lunak dibagi menjadi dua, yaitu visualisasi algoritma (VA) dan visualisasi program (VP) (Cetin dan Andrews-Larson, 2016; Sorva, 2012; Sorva dkk., 2013). VA berkaitan dengan abstraksi terhadap algoritma, konsep dan langkah kerja suatu perangkat lunak, sedangkan VP berkaitan dengan cara kerja eksekusi kode program dan proses struktur data.

Berdasarkan timeline pengembangan VP (Sorva dkk, 2013) hanya ada empat kakas VP yang telah dikembangkan untuk graf, yaitu: Swan (Shaffer dkk., 1996), *VisMod* (Jiménez-Peris dkk., 1999), jGRASP (Cross II dkk., 2007), *Jype* (Helminen dan Malmi, 2010), dan satu kakas VP yang berbasis web (HTML, CSS, dan JS) yaitu *Online Python Tutor* atau disingkat OPT (Guo, 2013). Swan adalah kakas VP untuk struktur data dan eksekusi kode C/C++. Kakas ini dapat memvisualisasikan graf, pohon, *list*, dan *array*. Metodenya dengan menggunakan anotasi pada kode program yang disebut SAIL (*Swan Annotation Interface Library*). Tujuan utama kakas Swan adalah untuk membuat *library* anotasi yang mudah digunakan dalam visualisasi. *VisMod* dapat memvisualisasikan eksekusi kode program dengan bahasa pemrograman *Modula-2*. Kakas ini mendukung visualisasi struktur data linier dan pohon. Metodenya dengan membaca data variabel *pointer* dan referensinya. Selain itu, kakas ini dapat mengecek kesalahan sintaks dan memastikan penggunaan semua variabel yang telah dideklarasikan sebelumnya. jGRASP adalah kakas visualisasi eksekusi kode program yang mendukung bahasa *Java*. Kakas ini dapat memvisualisasi *binary tree* dan *linked list*. Visualisasi berdasarkan pada tipe data *pointer* yang direpresentasikan sebagai *node* dan *reference* sebagai *edge*. *Jype* menggunakan *Matrix Framework* (Korhonen dkk., 2004) untuk memvisualisasikan struktur data seperti *array* dan pohon secara otomatis ketika terdeteksi di dalam kode program *Python*. *Jype* dikembangkan dengan *Java* yang dapat beroperasi di web sebagai *Java Applet*. Untuk mengaktifkannya, seluruh komponen dalam aplikasi harus diunduh, kemudian aplikasi dapat beroperasi di komputer.

*Online Python Tutor* atau disingkat OPT (Guo, 2013) adalah kakas VP yang bersifat bebas dan bersumber terbuka. Kakas ini telah memiliki banyak fitur yang mendukung berbagai macam bahasa pemrograman, seperti *Python, Java, C, C++, Ruby, JavaScript, TypeScript* dan masih terus dikembangkan[[1]](#footnote-1). Ketika penelitian ini dilakukan, OPT telah berkembang pada versi 5 yang dirilis pada tanggal 27 Juli 2016. Penelitian aktif dilakukan pada versi 5 dengan menggunakan bahasa pemrograman *TypeScript* sebagai dasar pembangunan kakas OPT.

Berdasarkan tinjauan pustaka dan eksplorasi yang telah dilakukan, sebagian besar kakas VP dibuat untuk komputer *desktop* atau *Java Virtual Machine* (JVM). Aplikasi *Java Applet* dapat beroperasi di web, namun konfigurasi *Java* pada *browser* harus sesuai dengan yang dibutuhkan. Ini tentu berbeda dengan aplikasi web *HTML*, *CSS*, dan *JavaScript* yang dapat langsung digunakan di *browser*. Pengguna tidak perlu melakukan konfigurasi *Java* untuk mengaksesnya.

Hasil eksplorasi diperoleh bahwa tidak ada satu pun kakas VP berbasis web dengan bahasa C atau C++ yang dapat menampilkan visualisasi graf. Sehingga ini menjadi peluang untuk penelitian lebih lanjut dalam pengembangannya. Terutama penggunaan *Typescript* sebagai dasar pengembangan kakas yang masih baru dan memiliki prospek ke depannya dalam teknologi web terkini. OPT dipilih menjadi dasar pengembangan kakas, karena selain telah mendukung basis web, kakas ini juga bersifat bebas dan bersumber kode terbuka. Kakas ini juga masih dalam proses pengembangan dan penelitian oleh pengembangnya[[2]](#footnote-2). Jadi, masih terdapat banyak peluang dan celah untuk diteliti lebih lanjut agar dapat dimanfaatkan dengan baik. Mengingat teknologi web dan aplikasi piranti bergerak terus berkembang pesat hingga dekade tahun terakhir ini.

1. **Design of Visualization**

Sebuah graf digambarkan dengan *node* dan *edge*. *Node* dan *edge* memiliki atribut masing-masing. Atribut *node* terdiri dari bentuk, warna, ukuran, dan label. Sedangkan atribut *edge* terdiri dari ketebalan garis, label, warna, dan anak panah (untuk graf berarah). Gambar dasar visual graf dengan *node* dan *edge* dapat dilihat pada Gambar I berikut ini.



Gambar I. Visual graf berarah dengan dua *node* dan *edge*

Visualisasi graf memiliki beberapa properti yang dibutuhkan. Pada Tabel I menunjukkan properti visual graf yang dapat direpresentasikan. Dari daftar itu tidak semua properti visual digunakan, harus dapat menyesuaikan data yang sedang diproses dari eksekusi kode program.

Tabel I. Daftar properti visual graf

| **No** | **Nama Properti** | **Visualisasi** |
| --- | --- | --- |
| 1 | Label untuk *node* | x |
| 2 | Bentuk *node* |  |
| 3 | Bentuk *edge* | atau |
| 4 | Label *edge* | x  x  atau |
| 5 | Warna *node* dan *edge* | atau  atau |
| 6 | Ketebalan *edge* | atau |

Untuk menganalisis teknik pendeteksian graf kode program, perlu diketahui terlebih dahulu cara representasi data graf dalam kode program sehingga sumber data untuk visualisasi dapat ditentukan. Representasi data graf dalam kode program dapat dilakukan dengan tiga cara, yaitu sebagai berikut (Sedgewick dan Wayne, 2011): a) Representasi *adjacency matrix* atau berupa data matriks; b) Representasi array of edges, yaitu menggunakan edge class dengan dua variabel bertipe int; dan c) Representasi array of adjacency lists, ini menggunakan indeks node yang menyimpan list dari simpul yang saling berdekatan atau bertetangga.

1. **Results**

Kakas yang dikembangkan fokus pada visualisasi graf eksekusi kode program. Tujuan pengembangan kakas adalah menambahkan fitur visualisasi graf pada kakas OPT, sehingga dapat digunakan untuk memahami eksekusi graf kode program. Gambar II menunjukkan skema umum hasil pengembangan kakas. *Input* berupa kode program C/C++, jika terdapat representasi graf di dalamnya, maka *output* dapat menyesuaikan untuk menampilkan visual graf.



Gambar II. Skema umum hasil pengembangan kakas



Gambar III. Skema umum modul visualisasi graf (*GraphVisualizer*)

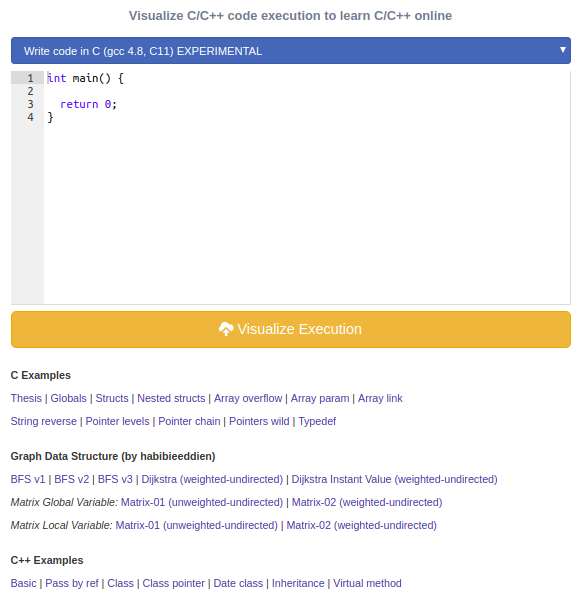
Pada Gambar III menunjukkan skema umum khusus untuk kelas visualisasi graf. Sebenarnya yang diproses dari *input* kelas ini adalah berupa data JSON, bukan kode program. Data JSON yang terdapat representasi graf, akan diekstrak dan *output* bisa menyesuaikan dengan menampilkan visualisasi graf atau tidak.

Representasi visual graf dapat melalui proses seperti pada Gambar IV. Pada proses “Pencocokan Model” telah dilakukan pada langkah sebelumnya (pada langkah 4 *Mine* atau menggali informasi). Proses selanjutnya adalah visualisasi menggunakan *library* *D3.js*.



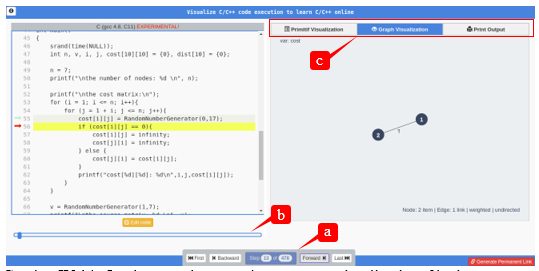
Gambar IV. Diagram alir *level-0* proses visualisasi data

Pada Gambar V menunjukkan implementasi antarmuka pengguna untuk input kode program C/C++ di browser Chrome. Pengguna dapat memilih contoh kode program di bagian bawah tombol ‘Visualize Execution’. Pengguna juga dapat mengetik sendiri kode program di code editor yang telah disediakan. Hasil visualisasi akan tampil seperti pada Gambar VI. Pengguna dapat eksplorasi eksekusi kode program dengan navigasi kontrol (Gambar VI.(a)) yang tersedia, atau dapat menggunakan slider (Gambar VI.(b)) untuk melihat langsung pada langkah yang diinginkan.

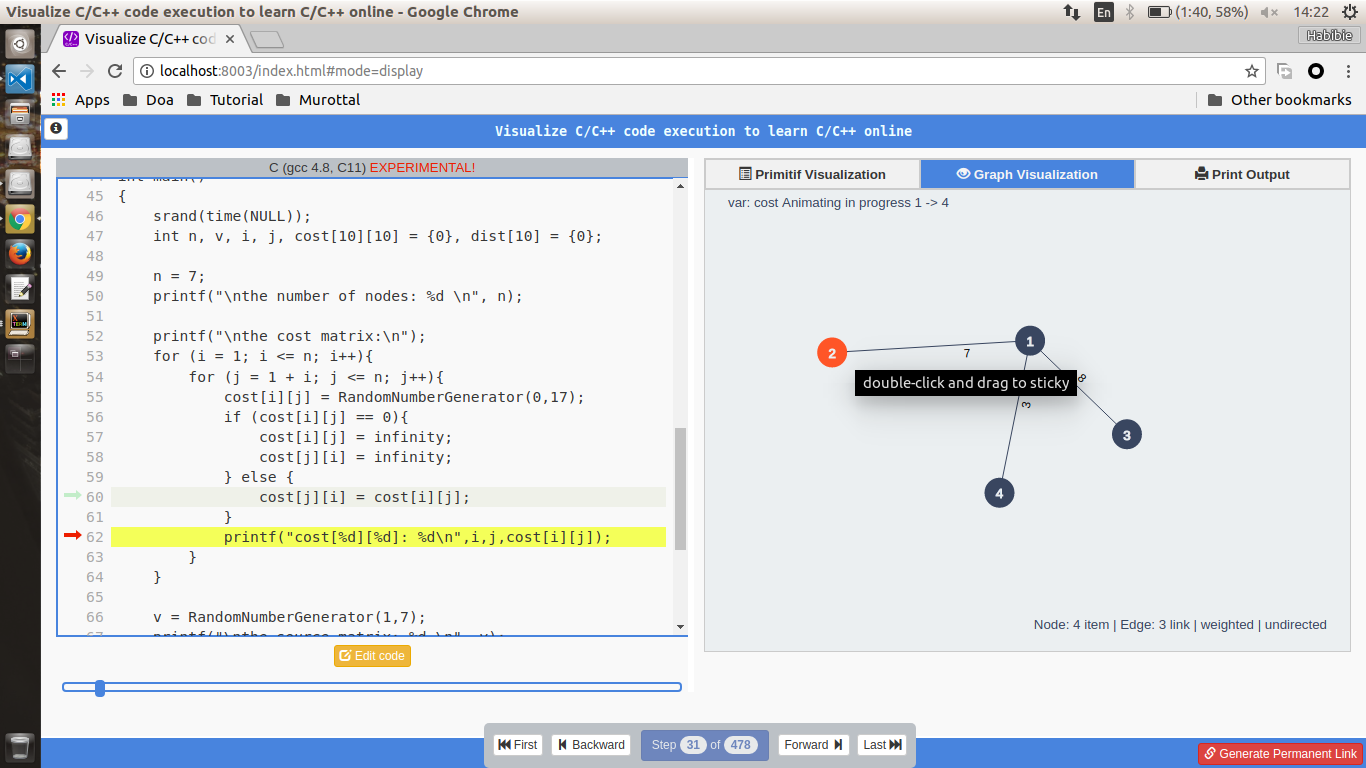


Gambar V. Implementasi antarmuka pengguna untuk input kode program

Gambar VI (c) merupakan panel yang terdiri dari tiga visualisasi, yaitu “Primitif Visualization” adalah untuk menampilkan data variabel kode program secara sederhana berupa stack dan heap. Panel “Graph Visualization” adalah bagian utama visualisasi graf yang dikembangkan dalam penelitian ini. Panel “Print Output” adalah untuk menampilkan standar output yang berasal dari perintah printf di dalam kode program.



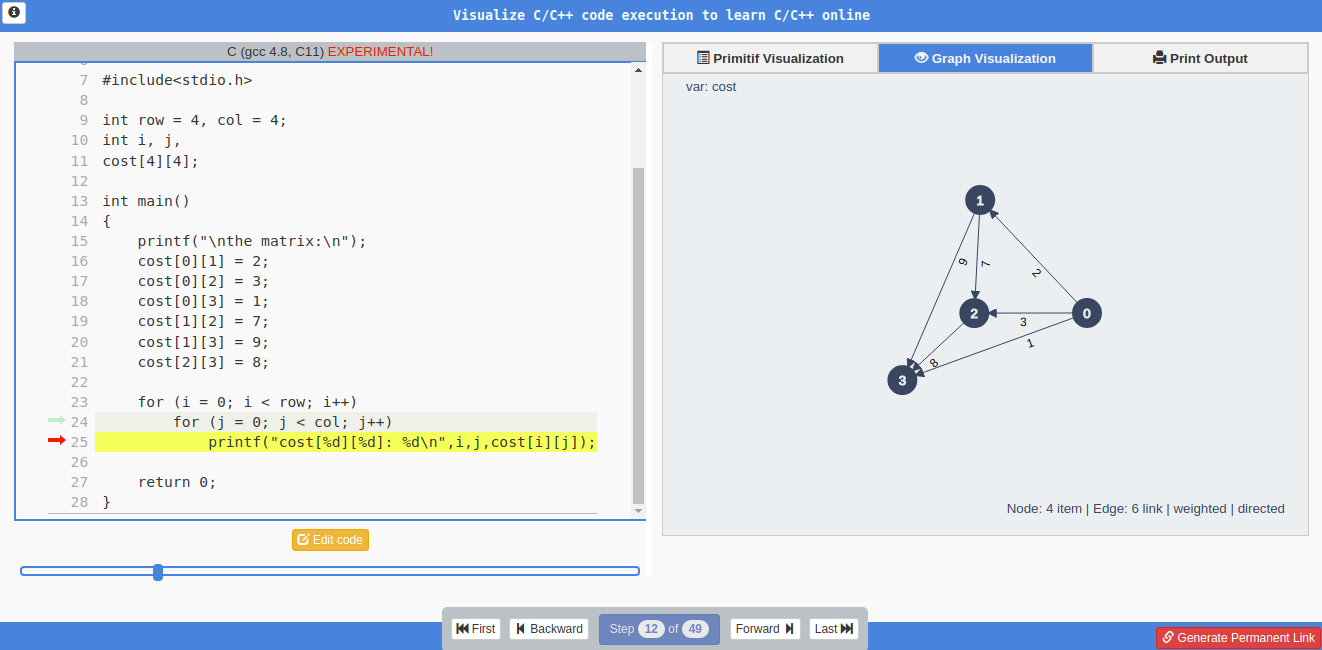
Gambar VI. Implementasi antarmuka pengguna visualisasi graf kode program; (a) Navigasi kontrol; (b) Slider; (c) Panel visualisasi



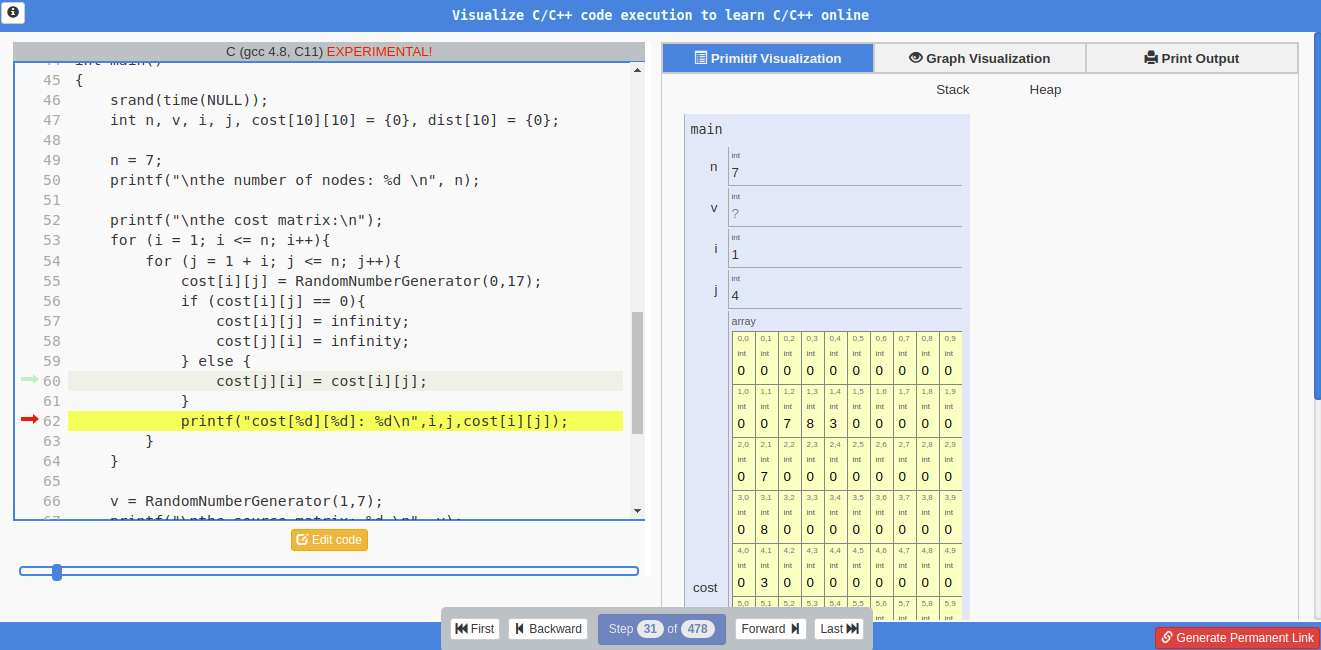
Gambar VII. Implementasi visualisasi graf berbobot tak-berarah

Pada Gambar VII menunjukkan implementasi kakas untuk visualisasi graf berbobot dan tak-berarah dengan representasi data matriks. Node dapat menerima aksi drag dan drop dari pengguna untuk diposisikan sesuai yang diinginkan oleh pengguna.

Pada Gambar VIII menunjukkan implementasi kakas untuk visualisasi graf berbobot dan berarah dengan representasi data matriks. Edge akan otomatis menampilkan anak panah jika terdeteksi data variabel yang berisi graf berarah. Label angka untuk bobot tiap edge juga otomatis tampil ketika terdeteksi variabel bobot antar node.



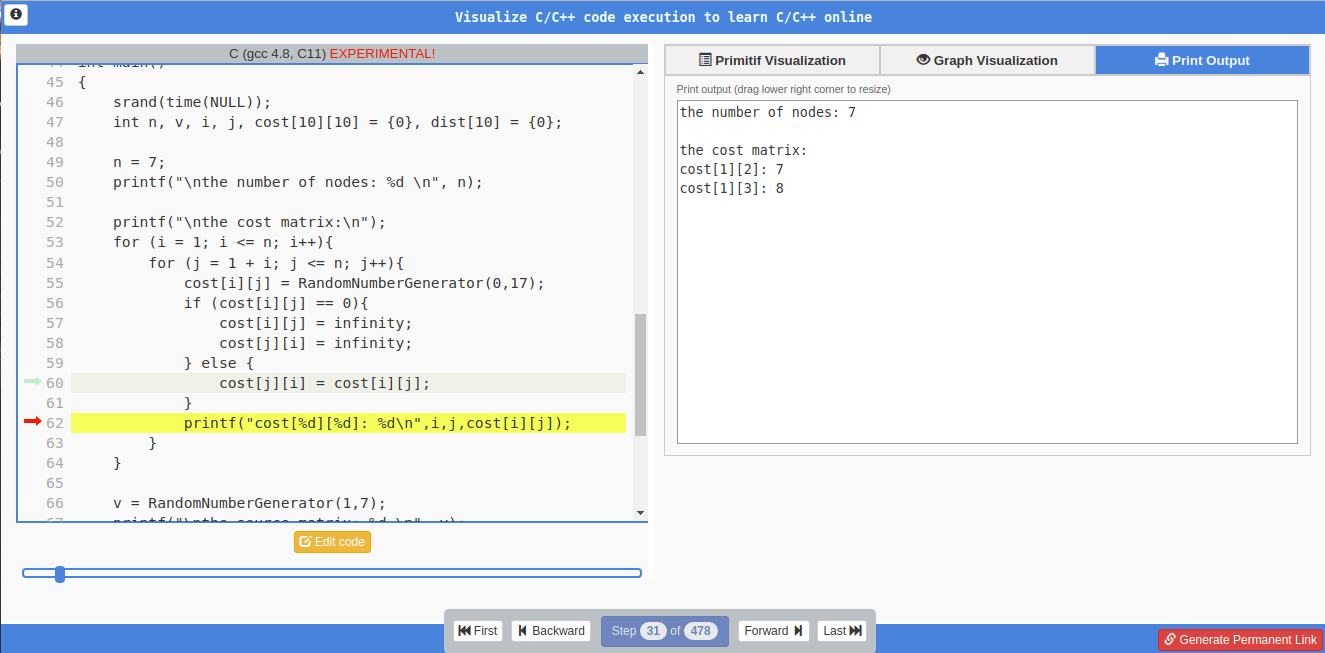
Gambar VIII. Implementasi visualisasi graf berbobot berarah



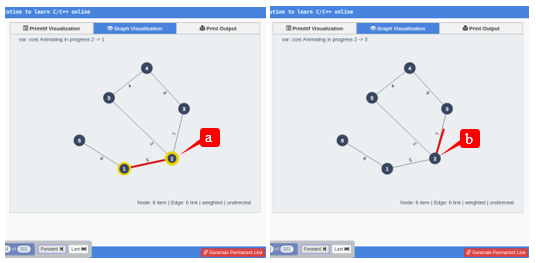
Gambar IX. Implementasi panel “Primitif Visualization”

Pada Gambar IX dan Gambar X merupakan implementasi untuk panel “Primitif Visualization” dan “Print Output” yang sebenarnya sudah tersedia pada kakas OPT sebelumnya. Hanya menempatkan pada panel untuk memudahkan pengguna dalam mengaksesnya sesuai yang telah dijelaskan pada subbab III.1.2 tentang rancangan antarmuka pengguna.

Pada Gambar XI menunjukkan implementasi fitur animasi untuk pencarian rute antar node. Huruf (a) pada Gambar XI merupakan animasi pencarian rute dari node 2 ke node 1. Huruf (b) pada Gambar XI merupakan kelanjutan animasi pencarian rute dari node 2 ke node 3.



Gambar X. Implementasi panel “Print Output”



Gambar XI. Implementasi fitur animasi pencarian rute antar node

1. **Future Work**

Berdasarkan hasil penelitian yang telah dilakukan, diketahui bahwa visualisasi memiliki keefektifan yang cukup baik bagi pengguna dalam proses memahami eksekusi graf kode program. Namun terkadang masih terkendala dengan lambatnya proses kompilasi kode program untuk bisa menjadi visualisasi. Kendala tersebut dapat mengurangi interaktifitas kakas. Harapan penelitian selanjutnya dapat difokuskan untuk melakukan kompilasi kode program dengan menggunakan sumber daya masing-masing komputer pengguna di *browser*. Sehingga proses kompilasi tidak membebani kerja komputer *server*.

1. **Conclusion**

Beberapa kesimpulan yang telah diperoleh selama penelitian ini adalah sebagai berikut:

1. Pendekatan visualisasi dapat menjadi sarana yang efektif dan efisien untuk memahami eksekusi graf kode program;
2. *Usability* adalah salah satu aspek penting dalam melakukan visualisasi, karena hal tersebut berkaitan dengan kemudahan penggunaan kakas. Kakas harus dapat membantu aktivitas manusia, bukan sebaliknya menjadi lebih rumit dan sulit untuk digunakan.
3. *TypeScript* (TS) adalah bahasa pemrograman *JavaScript* (JS) berbasis objek yang cukup menjanjikan untuk mengembangkan aplikasi atau kakas berbasis web berskala besarh yang interaktif. Kompleksitas program berbasis JS sangat didukung oleh TS.

**References**

1. Bonk, C.J. (2009): *The world is open: how Web technology is revolutionizing education*, 1st ed, San Francisco, Calif, Jossey-Bass.
2. Bostock, M., Ogievetsky, V. dan Heer, J. (2011): D3: Data-Driven Documents, *IEEE Trans. Vis. Comput. Graph.*, **17**, 2301–2309.
3. Bruening, D. dan Zhao, Q. (2011): Practical memory checking with Dr. Memory, *Proceedings of the 9th Annual IEEE/ACM International Symposium on Code Generation and Optimization*, IEEE Computer Society, 213–223.
4. Cetin, I. dan Andrews-Larson, C. (2016): Learning sorting algorithms through visualization construction, *Comput. Sci. Educ.*, **26**, 27–43.
5. Cross II, J.H., Hendrix, T.D., Jain, J. dan Barowski, L.A. (2007): Dynamic object viewers for data structures, *ACM SIGCSE Bulletin*, ACM, 4–8.
6. Diehl, S. (2007): *Software visualization: visualizing the structure, behaviour, and evolution of software ; with 5 tables*, Berlin, Springer.
7. Fry, B. (2008): *Visualizing Data*, First Edition, USA, O`Reilly Media, Inc.
8. Gestwicki, P. dan Jayaraman, B. (2005): Methodology and architecture of JIVE, *Proceedings of the 2005 ACM symposium on Software visualization*, ACM, 95–104.
9. Gračanin, D., Matković, K. dan Eltoweissy, M. (2005): Software visualization, Innov. Syst. Softw. Eng., 1, 221–230.
10. Guo, P.J. (2013): Online python tutor: embeddable web-based program visualization for cs education, Proceeding of the 44th ACM technical symposium on Computer science education, ACM, 579–584.
11. Helminen, J. dan Malmi, L. (2010): Jype-a program visualization and programming exercise tool for Python, *Proceedings of the 5th international symposium on Software visualization*, ACM, 153–162, diperoleh melalui situs internet: http://dl.acm.org/citation.cfm?id=1879234 (diakses 20 Juli 2016).
12. Hendrix, T.D., Cross II, J.H. dan Barowski, L.A. (2004): An extensible framework for providing dynamic data structure visualizations in a lightweight IDE, *ACM SIGCSE Bulletin*, ACM, 387–391.
13. Holmberg, N., Wünsche, B. dan Tempero, E. (2006): A framework for interactive web-based visualization, *Proceedings of the 7th Australasian User interface conference-Volume 50*, Australian Computer Society, Inc., 137–144, diperoleh melalui situs internet: http://dl.acm.org/citation.cfm?id=1151778 (diakses 15 September 2017).
14. Jiménez-Peris, R., Patiño-Martínez, M. dan Pacios-Martínez, J. (1999): VisMod: a beginner-friendly programming environment, ACM Press, 115–120, diperoleh melalui situs internet: http://portal.acm.org/citation.cfm? doid=298151.298218 (diakses 31 Desember 2017).
15. Korhonen, A., Malmi, L., Silvasti, P., Karavirta, V., Lönnberg, J., Nikander, J., Stålnacke, K. dan Tenhunen, P. (2004): *Matrix - a framework for interactive software visualization*, Research Report TKO-B 154/04, Department of Computer Science and Engineering, Helsinki University of Technology, 26–35.
16. Nielsen, J. (n.d.): Usability First - Methods - Heuristic Evaluation | Usability First, *Heuristic Evaluation*, diperoleh melalui situs internet: http://www.usabilityfirst.com/usability-methods/heuristic-evaluation (diakses 15 Februari 2018).
17. Piteira, M. dan Costa, C. (2013): Learning computer programming: study of difficulties in learning programming, *Proceedings of the 2013 International Conference on Information Systems and Design of Communication*, ACM, 75–80, diperoleh melalui situs internet: http://dl.acm.org/citation.cfm? id=2503871 (diakses 8 September 2016).
18. Preece, J. (2002): *Interaction Design: Beyond Human-Computer Interaction, First Edition*, First Edition, USA, John Wiley & Sons, Inc.
19. Preece, J., Sharp, H. dan Rogers, Y. (2015): *Interaction Design, beyond human-computer interaction Fourth Edition*, 4th edition, John Wiley & Sons, Ltd.
20. Sedgewick, R. dan Wayne, K. (2011): *Algorithms Fourth Edition*, Fourth Edition, USA, Pearson Education, Inc.
21. Shaffer, C.A., Heath, L.S. dan Yang, J. (1996): Using the Swan data structure visualization system for computer science education, *ACM SIGCSE Bull.*, **28**, 140–144.
22. Sorva, J. (2012): *Visual program simulation in introductory programming education*, Aalto University publication series Doctoral dissertations, Espoo, Aalto Univ. School of Science.
23. Sorva, J., Karavirta, V. dan Malmi, L. (2013): A review of generic program visualization systems for introductory programming education, ACM Trans. Comput. Educ. TOCE, 13, 15.
24. Sue, V.M. dan Ritter, L.A. (2007): *Conducting online surveys*, United States of America, Sage Publications.
25. Thompson, E., Luxton-Reilly, A., Whalley, J.L., Hu, M. dan Robbins, P. (2008): Bloom’s Taxonomy for CS Assessment, ACE ’08, Wollongong, NSW, Australia, Australian Computer Society, Inc., 155–161, diperoleh melalui situs internet: https://dl.acm.org/citation.cfm?id=1379265.
26. Urquiza-Fuentes, J. dan Velázquez-Iturbide, J.Á. (2009): A Survey of Successful Evaluations of Program Visualization and Algorithm Animation Systems, *ACM Trans. Comput. Educ.*, **9**, 1–21.
27. Velázquez-Iturbide, J.Á., Hernán-Losada, I. dan Paredes-Velasco, M. (2017): Evaluating the Effect of Program Visualization on Student Motivation, *IEEE TRANSACTIONS ON EDUCATION*.
28. Ware, C. (2004): *Information visualization: perception for design, 2nd edition*, 2nd edition, San Francisco, Kanada, Elsevier Inc.

1. https://github.com/pgbovine/OnlinePythonTutor [↑](#footnote-ref-1)
2. https://github.com/pgbovine/OnlinePythonTutor [↑](#footnote-ref-2)