**PENGEMBANGAN KAKAS**

**VISUALISASI EKSEKUSI KODE UNTUK GRAF**

**SEBAGAI MEDIA BELAJAR PEMROGRAMAN**

**PROPOSAL TESIS**

**Karya tulis sebagai salah satu syarat**

**untuk memperoleh gelar Magister dari**

**Institut Teknologi Bandung**

**Oleh**

**HABIBIE ED DIEN**

**NIM: 23515043**

**(Program Studi Magister Informatika)**

****

**INSTITUT TEKNOLOGI BANDUNG**

**Desember 2017**

**PENGEMBANGAN KAKAS**

**VISUALISASI EKSEKUSI KODE UNTUK GRAF**

**SEBAGAI MEDIA BELAJAR PEMROGRAMAN**

Oleh

**Habibie Ed Dien**

**NIM: 23515043**

**(Program Studi Magister Informatika)**

Institut Teknologi Bandung

Menyetujui

Tim Pembimbing

Tanggal \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

|  |  |
| --- | --- |
| Pembimbing Pertama  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (Dr. Ir. M.M. Inggriani) | Pembimbing Kedua  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (Yudistira Dwi W. Asnar, S.T., Ph.D.) |

# DAFTAR ISI

HALAMAN PENGESAHAN i

[DAFTAR ISI ii](#_Toc501911355)

[DAFTAR GAMBAR DAN ILUSTRASI iii](#_Toc501911356)

[DAFTAR TABEL iv](#_Toc501911357)

[Bab I Pendahuluan 1](#_Toc501911358)

[I.1 Latar Belakang 1](#_Toc501911359)

[I.2 Rumusan Masalah 3](#_Toc501911360)

[I.3 Tujuan Penelitian 3](#_Toc501911361)

[I.4 Batasan Implementasi 3](#_Toc501911362)

[I.5 Metodologi Penelitian 4](#_Toc501911363)

[I.6 Sistematika Penulisan 4](#_Toc501911364)

[I.7 Rencana Waktu Penyelesaian Tesis 5](#_Toc501911365)

[Bab II Tinjauan Pustaka dan Eksplorasi 6](#_Toc501911366)

[II.1 Terminologi Visualisasi Perangkat Lunak 6](#_Toc501911367)

[II.1.1 Alasan Menggunakan Visualisasi 7](#_Toc501911368)

[II.1.2 Sistem dan Manfaat Visualisasi Program (VP) 7](#_Toc501911369)

[II.2 Metodologi dan Prinsip Visualisasi Data 8](#_Toc501911370)

[II.3 Teori Desain Interaksi 10](#_Toc501911371)

[II.4 Representasi Struktur Data Graf 12](#_Toc501911372)

[II.5 Hasil Eksplorasi Kakas Visualisasi Program 14](#_Toc501911373)

[II.6 Eksplorasi Kakas *Online Python Tutor* 15](#_Toc501911374)

[II.6.1 Arsitektur Kakas OPT 17](#_Toc501911375)

[II.6.2 *Capturing Execution Trace* 19](#_Toc501911376)

[II.6.3 *Execution Trace Format* 20](#_Toc501911377)

[II.6.4 Fitur Pustaka *Data-Driven Documents* (D3) 21](#_Toc501911378)

[II.7 Kesimpulan Awal Berdasarkan Studi Literatur dan Eksplorasi 22](#_Toc501911379)

[Bab III Prelimineri Masalah 23](#_Toc501911380)

[III.1 Analisis Masalah 23](#_Toc501911381)

[III.2 Hipotesis Solusi Penyelesaian Masalah 26](#_Toc501911382)

[III.3 Penunjang Pengembangan Kakas 27](#_Toc501911383)

[III.3.1 Kebutuhan Perangkat Lunak 27](#_Toc501911384)

[III.3.2 Kebutuhan Perangkat Keras 28](#_Toc501911385)

[DAFTAR PUSTAKA 29](#_Toc501911386)

# DAFTAR GAMBAR DAN ILUSTRASI

[Gambar II.1 Keterhubungan antar tujuh tahapan (Fry, 2008) 9](#_Toc501911291)

[Gambar II.2 Konseptual model komponen interaksi (Fry, 2008) 11](#_Toc501911292)

[Gambar II.3 Contoh data matriks sebagai representasi graf 13](#_Toc501911293)

[Gambar II.4 Contoh data *array* atau *edge list* 13](#_Toc501911294)

[Gambar II.5 Contoh data *adjacency list* dengan indeks *vertex* 14](#_Toc501911295)

[Gambar II.6 Tampilan Antarmuka OPT untuk Bahasa Pemrograman C 16](#_Toc501911296)

[Gambar II.7 Tampilan Antarmuka OPT untuk Kolaborasi Pemrograman 16](#_Toc501911297)

[Gambar II.8 Tampilan Simulasi dan Visualisasi Kode C Pointer 17](#_Toc501911298)

[Gambar II.9 Arsitektur OPT untuk visualisasi kode C dan C++ 18](#_Toc501911299)

[Gambar II.10 Ilustrasi Sederhana Proses Kakas OPT 19](#_Toc501911300)

[Gambar II.11 Ilustrasi Format *Trace* Eksekusi Program 20](#_Toc501911301)

[Gambar II.12 Peran Utama *D3JS* sebagai *Framework* Visualisasi 21](#_Toc501911302)

[Gambar III.1 Visual primitif berupa tabel matrik 24](#_Toc501911303)

[Gambar III.2 Visual primitif berupa *struct* dan *pointer* 24](#_Toc501911304)

[Gambar III.3 Contoh data *JSON pointer* pada *heap* 25](#_Toc501911305)

# DAFTAR TABEL

[Tabel I.1 Jadwal rencana kegiatan penelitian tesis 5](#_Toc501911387)

[Tabel II.1 Perkembangan kakas VP untuk bahasa pemrograman C/C++ 14](#_Toc501911388)

[Tabel III.1 Variabel *‘a’* berupa matriks berdimensi 7 x 7 25](#_Toc501911389)

[Tabel III.2 Isi alamat memori pada objek *heap* 26](#_Toc501911390)

[Tabel III.3 Properti visual graf untuk visualisasi 27](#_Toc501911391)

[Tabel III.4 Kakas dan *library* pendukung pengembangan OPT 28](#_Toc501911392)

# Bab I Pendahuluan

Pada bab ini dijelaskan mengenai latar belakang, rumusan masalah, tujuan penelitian, batasan implementasi, metodologi penelitian, sistematika penulisan laporan tesis dan rencana waktu penyelesaian tesis.

## I.1 Latar Belakang

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

Bagi pengajar untuk menjelaskan proses eksekusi kode di kelas terkadang menggunakan papan tulis atau *slide PowerPoint*. Hal ini membutuhkan persiapan ekstra terhadap materi yang perlu disiapkan terkait gambar, alur diagram, atau bagian-bagian kode program yang perlu dijelaskan. Terutama jika materi pemrograman itu memasuki ke tingkat yang lebih rumit, seperti struktur data. Diperlukan media belajar khusus untuk dapat menjelaskan proses eksekusi kode itu. Sehingga pelajar 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 pelajar. Hal itu disebabkan konsep pemrograman untuk struktur data merupakan konsep data abstrak yang tidak tampak jelas dalam kode. Penelitian tersebut juga memberikan perhatian khusus terhadap konsep pemrograman seperti struktur data 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 interaksi 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 atau dengan istilah visualisasi program (VP) untuk graf masih sangat langka (Sorva, 2012; Sorva dkk., 2013). Pada bab 11 disertasi Sorva (2012) menelaah secara komprehensif perkembangan 40 kakas VP. Hasilnya tidak terdapat kakas VP yang mendukung visualisasi graf.

Pada penelitian yang telah dilakukan oleh Philip Guo (2010), kakas visualisasi eksekusi kode telah dibangun 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 dengan tujuan tertentu. Kakas ini menggunakan *D3JS* (Bostock dkk., 2011) sebagai teknologi utama dalam melakukan visualisasi. Kakas ini dapat memvisualisasikan berupa data primitif dan tumpukan data di dalam sebuah tabel. Kelemahannya kakas belum mendukung untuk visualisasi struktur data graf.

Pada penelitian tesis ini, pengembangan kakas dilakukan berdasar 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 pelajar untuk mempelajari eksekusi kode yang terdapat graf didalamnya.

## I.2 Rumusan Masalah

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

1. Apa strategi pengembangan kakas untuk mendeteksi graf di dalam kode sehingga dapat divisualisasikan ?
2. Desain interaksi apa yang sesuai untuk visualisasi graf sehingga dapat mendukung sebagai media belajar pemrograman ?

## I.3 Tujuan Penelitian

Tujuan dari penelitian tesis ini adalah untuk menghasilkan modifikasi kakas OPT yang dapat melakukan visualisasi graf dari eksekusi kode.

## I.4 Batasan Implementasi

Batasan implementasi pada tesis ini adalah :

1. Implementasi kakas VP berdasar kode sumber dari OPT dengan memanfaatkan pustaka *D3JS*. Karena dari hasil penelitian Sorva (2012), hanya kakas OPT ini yang telah mendukung teknologi web dan menggunakan pustaka *D3JS*. Pustaka ini sudah banyak mendukung *browser* modern dan mudah untuk melakukan visualisasi data.
2. Bahasa pemrograman yang dapat divisualisasikan adalah C dan C++ dengan maksimal 1000 langkah visual *render*. Karena hal ini tergantung kemampuan *processor* dan *RAM* pada komputer *server* untuk menghasilkan data *trace* eksekusi *JSON*.
3. Visualisasi graf dapat dilakukan dengan maksimal 100 simpul.
4. Kakas tidak menerima *standard input* (*stdin*) dan tidak dapat menerima deklarasi *header file* dari pengguna, artinya hanya *standard library C/C++* yang sudah tersedia di *compiler server*.

## I.5 Metodologi Penelitian

Uraian metode penelitian dan rencana pengembangan serta pengujian dijelaskan pada bagian ini yang akan dilaksanakan melalui beberapa tahapan sebagai berikut.

1. Tinjauan Pustaka dan Eksplorasi

Tinjauan pustaka menghasilkan kesimpulan awal yang dapat dijadikan landasan dalam perumusan masalah dan pengembangan kakas. Kemudian dilakukan eksplorasi terhadap kakas VP yang telah dikembangkan.

1. Analisis Masalah dan Perumusan Model Visualisasi

Tahap ini dilakukan kajian terhadap masalah yang akan diteliti kemudian dirumuskan hipotesis untuk disusun solusi yang mungkin dapat diterapkan. Selanjutnya, dirumuskan suatu model untuk mendeteksi graf pada kode.

1. Pengembangan Kakas

Pengembangan kakas dilakukan dengan beberapa tahap, yaitu (a) eksplorasi OPT dilakukan dengan meninjau proses kerja sistem; (b) menganalisis dan merancang diagram alur sistem; (c) merumuskan hipotesis solusi dan tahapan visualisasi data; dan (d) implementasi dan perbaikan kakas.

1. Pengujian dan Evaluasi Kakas

Hasil pengembangan kakas perlu diuji dengan memasukkan berbagai jenis kode dan tipe data, kemudian menganalisis hasil dari visualisasi tersebut. Selain itu, kakas perlu dievaluasi secara empiris dengan bantuan responden untuk menilai efektivitas kakas dalam representasi visual graf.

## I.6 Sistematika Penulisan

Penulisan laporan hasil penelitian tesis ini akan dibagi menjadi enam bab, yaitu:

1. Bab I Pendahuluan

Bab ini berisi latar belakang, rumusan masalah, tujuan, metodologi penelitian, dan sistematika penulisan laporan.

1. Bab II Tinjauan Pustaka dan Eksplorasi

Bab ini berisi uraian tentang terminologi visualisasi perangkat lunak, perkembangan kakas visualisasi dari hasil penelitian sebelumnya yang berkaitan dengan masalah yang dikaji, sehingga memberikan gambaran perkembangan terhadap masalah yang akan diteliti. Selain itu, eksplorasi perkembangan kakas VP dilakukan untuk menganalisis masalah dan merumuskan hipotesis atau solusi yang mungkin.

1. Bab III Analisis Masalah dan Model Visualisasi

Bab ini berisi analisis terhadap masalah yang akan diteliti, kemudian dirumuskan hipotesis untuk disusun solusi yang mungkin dapat diterapkan. Untuk mendukung hipotesis tersebut, dibangun pula rancangan fitur agar dapat diintegrasikan dengan sistem kakas yang akan dikembangkan.

1. Bab IV Pengembangan Kakas

Bab ini menjelaskan proses pengembangan dan implementasi kakas VP.

1. Bab V Pengujian dan Evaluasi Kakas

Bab ini menguraikan proses dan hasil pengujian serta evaluasi dari kakas yang dikembangkan. Pengujian dilakukan untuk membuktikan bahwa kakas berhasil dikembangkan lebih baik dibandingkan sebelumnya. Evaluasi kakas dilakukan untuk menilai efektivitas dan kualitas representasi visualisasi.

1. Bab VI Penutup

Bab ini berisi kesimpulan dan saran untuk pengembangan kakas lebih lanjut.

## I.7 Rencana Waktu Penyelesaian Tesis

Penelitian tesis ini akan dilaksanakan sekurang-kurangnya dalam waktu enam bulan dan paling lama dua belas bulan, dengan rincian kegiatan yang tercantum pada Tabel I.1 berikut ini.

Tabel I.1 Jadwal rencana kegiatan penelitian tesis

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **No.** | **Kegiatan** | **Bulan ke-** | | | | | |
| **1** | **2** | **3** | **4** | **5** | **6** |
| 1 | Studi literatur dan eksplorasi |  |  |  |  |  |  |
| 2 | Analisis masalah dan perumusan model visualisasi |  |  |  |  |  |  |
| 3 | Pengembangan kakas |  |  |  |  |  |  |
| 4 | Pengujian kakas |  |  |  |  |  |  |
| 5 | Evaluasi kakas dan penulisan laporan |  |  |  |  |  |  |

# Bab II Tinjauan Pustaka dan Eksplorasi

Pada bab ini dijelaskan mengenai terminologi visualisasi, proses konstruksi visualisasi data, perkembangan teknologi web dan pustaka untuk visualisasi, graf dan model visualnya, perkembangan kakas VP, hasil eksplorasi kakas OPT, dan kesimpulan awal berdasarkan studi literatur dan eksplorasi.

## II.1 Terminologi Visualisasi Perangkat Lunak

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 terlalu banyak fokus terhadap proses konstruksi program, akan tetapi lebih kepada analisis program dan proses pengembangan perangkat lunak.

Istilah “visualisasi perangkat lunak” telah lama berkembang dan didefinisikan sebagai sebuah seni tipografi, desain grafis, animasi, dan sinematografi melalui interaksi modern antar manusia-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 datanya.

### II.1.1 Alasan Menggunakan Visualisasi

Ada sebuah kalimat kutipan dari seorang matematikawan Amerika, yaitu sebagai berikut:

"*The purpose of computing is insight, not numbers*", R. Hamming (1962).

Merujuk dari kutipan tersebut, tujuan dari komputasi adalah untuk memperoleh pengetahuan atau informasi yang dibutuhkan oleh manusia, bukan sekumpulan angka yang belum memiliki makna tertentu. Berkaitan dengan visualisasi, memiliki makna untuk mengungkapkan sesuatu yang belum terlihat dari suatu data, dengan demikian, membuatnya lebih mudah dimengerti.

Beberapa alasan menggunakan visualisasi adalah sebagai berikut (Diehl, 2007):

* Membutuhkan skala ukuran fisik yang cukup besar dalam membuat presentasi pemahaman suatu data, jika harus membuat diagram atau gambar satu demi satu;
* Perubahan data dalam rentang waktu tertentu membutuhkan ruang untuk dapat menyampaikan pemahaman;
* Dibutuhkan fitur data tertentu yang penting untuk disajikan;
* Dibutuhkan tafsiran atau interpretasi dari suatu data;
* Subjek tidak nampak jelas jika tanpa visualisasi.

### II.1.2 Sistem dan Manfaat Visualisasi Program (VP)

Visualisasi program (VP) terdiri dari pembuatan animasi eksekusi program.

Animasi ini membuat hasil visual yang sangat mengesankan. Hal ini tidak terbatas pada algoritma atau aktivitas dari eksekusi kode. Beberapa aktivitas yang mungkin terjadi saat kode dikompilasi adalah seperti sistem *run-time*, data, dan bahkan perangkat keras yang mendasarinya. Sebagian besar sistem VP terbatas pada satu bahasa pemrograman, yang biasanya sama dengan bahasa yang digunakan untuk menulis sistem visualisasi tersebut. Namun ini bukan hal yang mutlak untuk diimplementasikan.

Kakas VP memiliki beberapa manfaat, yaitu (Sorva dkk., 2013):

* *Debugging* atau proses mencari kesalahan dalam kode program;
* Mengevaluasi dan memperbaiki performa suatu kode program;
* Mengevaluasi dan mengurangi pemanfaatan sumber daya;
* Mengevaluasi algoritma dalam konteks kode program lengkap dan data riil;
* Memahami perilaku kode program;
* Sebagai media belajar mengajar.

## II.2 Metodologi dan Prinsip Visualisasi Data

Kompleksitas data tidak akan bermakna menjadi sebuah informasi yang bermanfaat jika memahaminya dengan cara yang kurang tepat. Proses visualisasi dilakukan dengan mengombinasikan berbagai bidang disiplin ilmu, seperti statistik, *data mining*, desain grafis, dan visualisasi informasi. Desain grafis tidak menangani jutaan data, akan tetapi *data mining* yang memiliki kapasitas itu. Visualisasi perangkat lunak membuat interaksi dan berbagai jenis representasi data abstrak, namun metode yang digunakan untuk visual dinilai kurang aestetik karena tidak adanya keterlibatan dari desain grafis. Oleh karena itu, ketiga bidang disiplin ilmu tersebut harus saling melengkapi menjadi sebuah proses yang dapat mengomunikasikan data lebih efektif untuk direpresentasikan (Fry, 2008).

Untuk dapat melakukan visualisasi data, dibutuhkan beberapa tahapan yang merupakan kombinasi dari ketiga bidang disiplin ilmu tersebut. Meskipun ketiga bidang itu memiliki prinsipnya sendiri, tetap harus disesuaikan setiap bagian menjadi kesatuan proses. Prosesnya dimulai dengan memahami seperangkat data mentah dan mengajukan sebuah pertanyaan yang berkaitan dengan data tersebut. Secara rinci ada tujuh tahapan proses konstruksi visualisasi data (Fry, 2008):

1. *Acquire* (sumber data); cara perolehan sebuah data, apakah dari sebuah berkas di dalam *harddisk* komputer atau bersumber dari jaringan internet.
2. *Parse* (pengelompokan); mengurai struktur data menjadi makna tertentu dan mengategorikannya.
3. *Filter* (menyaring); menghapus data yang tidak penting sehingga tersisa data yang menjadi fitur.
4. *Mine* (menggali informasi); terapkan metode dari ilmu statistik atau *data mining* sebagai cara untuk memahami data sesuai konteksnya.
5. *Represent* (merepresentasikan); pilih visual model dasar seperti diagram batang, diagram garis, atau graf.
6. *Refine* (perbaikan visual); lakukan perbaikan terhadap representasi visual dasar untuk membuat lebih jelas dan menarik.
7. *Interact* (interaksi); tambahkan interaksi untuk memanipulasi data atau sebuah kontrol terhadap fitur data yang disajikan.

Tahapan tersebut tidak harus diikuti semua. Pada visualisasi tertentu hanya empat dari tujuh tahapan yang bisa diterapkan, namun pada visualisasi data lain mungkin semua dari tahapan tersebut harus diterapkan. Pada Gambar II.1 menunjukkan tahapan secara berurutan dan keputusan akhir berdasarkan tahapan yang telah dilakukan sebelumnya. Setiap tahap pada proses ini saling terhubung karena saling memengaruhi satu dengan yang lain.



1. Keterhubungan antar tujuh tahapan (Fry, 2008)

Keterhubungan antar setiap tahapan dalam proses tersebut menggambarkan sangat pentingnya kerja sama antar individu atau kerja tim dalam kesatuan proyek. Biasanya seorang *programer* menangani bagian teknis, seperti cara perolehan dan pengolahan data, dan *desainer* fokus terhadap pemilihan warna dan tampilan antarmuka. Ketika proses pengambilan data, biasanya ditentukan kapan data itu harus tersedia, setiap waktu atau sekali dalam sebulan. Hal ini juga menentukan desain grafis yang cocok untuk masalah ketersediaan data berikutnya.

Ada tiga prinsip utama untuk membuat visualisasi data (Fry, 2008), yaitu :

1. Sadari bahwa setiap proyek visualisasi memiliki spesifikasi kebutuhan yang unik, karena tidak setiap data dapat direpresentasikan ke bentuk visual yang sama;
2. Hindari visual data yang terlalu rumit atau kurangi rincian data yang dapat mengganggu fitur data tersebut, sehingga pengguna tidak dapat melihat apa yang menjadi fokus dalam visualisasi tersebut; dan
3. Pahami siapa konsumen atau pengguna dari visualisasi tersebut. Visualisasi yang ditampilkan menggunakan komputer meja dibandingkan dengan gawai, tentu berbeda dalam hal interaksi visual dan tampilan kerumitan datanya.

## II.3 Teori Desain Interaksi

Desain interaksi secara khusus memiliki makna untuk menciptakan pengalaman pengguna (*user experience* / UX) dalam meningkatkan fungsi suatu produk sehingga dapat saling berkomunikasi dan berinteraksi dengan manusia. Winograd (Preece, 2002) mendeskripsikan bahwa desain interaksi adalah ruang untuk berkomunikasi dan berinteraksi dengan manusia yang bertujuan untuk mendukung aktivitas manusia. Komunikasi dan interaksi merupakan pencerminan dari aksi dan reaksi terhadap sesuatu. Media interaktif dikatakan efektif dan aestetik ketika dapat memberikan reaksi terhadap setiap aksi yang dilakukan oleh pengguna sehingga tercipta komunikasi.

Untuk dapat memahami konsep interaksi, dapat melihat pada Gambar II.2. Pada gambar tersebut, ada tiga esensi utama untuk bisa dikatakan sebagai sebuah interaksi pada desain produk (baik itu perangkat lunak atau keras), yaitu *designer*, pengguna, dan sistem. Secara tidak langsung, ketiga komponen ini saling berkaitan membentuk konseptual model (Fry, 2008) sebagai berikut:

* Model desain—model yang dibuat oleh *designer* bagaimana sistem itu seharusnya bekerja;
* Sistem—bagaimana sistem itu sebenarnya bekerja;
* Model pengguna—bagaimana pengguna dapat memahami sistem itu bekerja.



1. ­­Konseptual model komponen interaksi (Fry, 2008)

Idealnya dalam dunia nyata, ketiga komponen tersebut saling memetakan dengan komponen lainnya. Pengguna harus dapat melakukan aktivitas atau tugasnya sesuai yang didesain untuk dapat berinteraksi dengan sistem. Jika pengguna tidak dapat memahami model desain, maka tidak akan dapat memahami kerja sistem sehingga sistem akan berjalan dengan tidak efektif dan muncul kesalahan sistem (*error*).

Aspek kognisi (Fry, 2008) adalah salah satu kemampuan yang dimiliki oleh pengguna seperti berpikir, memperhatikan, mempelajari, mengingat, persepsi, membuat keputusan, merencanakan, membaca, berbicara, dan mendengar. Aspek ini sangat penting dan memiliki relevansi terhadap model desain dan proses interaksi. Sebuah tampilan antarmuka dapat dikatakan baik jika pengguna dapat mempersepsikan, mempelajari, dan mengingatnya untuk menyelesaikan tugas atau aktivitasnya. Kognisi memiliki tiga perspektif atau pendekatan (Fry, 2008), diantaranya:

* Mental model—didefinisikan sebagai pemahaman dan pengalaman dasar yang dimiliki pengguna terhadap kebiasaan menggunakan suatu produk;
* Memroses informasi—pendekatan ini memberikan informasi yang mudah dipersepsikan bagi pengguna untuk menggunakan produk;
* Kognisi eksternal—merupakan proses kognisi untuk berinteraksi dengan berbagai bentuk representasi atau media eksternal dalam membantu pengguna untuk mendukung aktivitasnya.

Secara esensi proses pengembangan desain interaksi memiliki empat tahap utama (Preece, 2002), yaitu:

1. Mengidentifikasi kebutuhan dan menetapkan persyaratan khusus;
2. Mengembangkan alternatif desain yang sesuai dengan persyaratan tersebut;
3. Membangun purwarupa versi desain interaktif sehingga dapat saling berkomunikasi dan dinilai kelayakannya;
4. Mengevaluasi apa yang sedang dibangun selama proses pengembangan desain berlangsung.

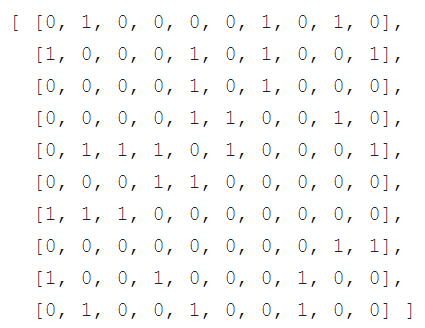
Keempat tahap tersebut saling berkaitan dan harus selalu diulang untuk memastikan desain yang dibangun telah memenuhi syarat dan kebutuhan. Sebagai contoh, kegunaan (*usability*) suatu produk yang telah dibangun dapat diukur dengan memperhatikan setiap perbaikan yang telah dilakukan atau dengan melihat persyaratan tertentu yang belum terpenuhi.

Pengguna mengharapkan sebuah sistem dan produk yang mudah untuk dipelajari dan digunakan dengan efektif, efisien, aman, dan memuaskan. Tampilan yang menarik dan atraktif juga merupakan esensi dari beberapa produk. Namun, semua itu tidak dapat secara langsung terwujud, akan tetapi diperlukan evaluasi secara terus-menerus dan berulang melalui pengujian dan survei kepada pengguna. Terdapat tiga prinsip yang direkomendasikan dalam hal ini (Fry, 2008), yaitu (1) fokus terhadap pengguna dan aktivitasnya terhadap produk tersebut; (2) amati, ukur, dan analisis interaksi pengguna dengan produk atau sistem; dan (3) lakukan desain secara berulang dengan versi berbeda. Evaluasi ini dibutuhkan untuk memastikan pengguna dapat menggunakan produk atau sistem dengan nyaman.

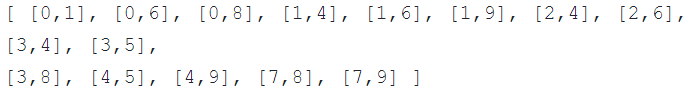
## II.4 Representasi Struktur Data Graf

Graf merupakan representasi abstrak (*abstract data type*) yang hanya dapat dibaca oleh manusia. Komputer tidak dapat memahami representasi tersebut, sehingga dalam kode program, graf direpresentasikan dengan caranya tersendiri. Ada beberapa cara untuk merepresentasikan graf, masing-masing memiliki kelebihan dan kekurangan. Representasi graf dalam kode program dapat dilakukan dengan tiga cara, yaitu sebagai berikut (Sedgewick dan Wayne, 2011):

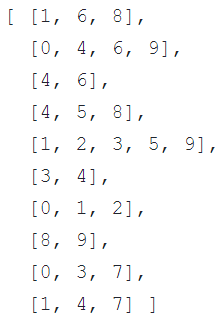
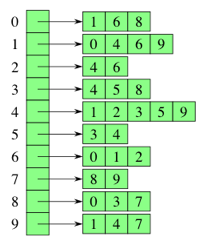
1. Representasi *adjacency matrix* atau berupa data matriks yang simetris dengan 2-dimensi *array* (lihat Gambar II.3), yaitu baris dan kolom sebagai *node*. Baris dan kolom yang ada *edge*, diberi nilai 1 atau *true*, jika kosong maka diberi nilai 0 atau *false*. Jika matriks tidak simetris, maka direpresentasikan sebagai graf berarah begitu sebaliknya. Akan tetapi representasi ini tidak efektif karena akan membutuhkan ruang sebesar ;



1. Contoh data matriks sebagai representasi graf
2. Representasi *array of edges*, yaitu menggunakan *edge class* dengan dua variabel bertipe *int* (lihat Gambar II.4). Namun *class* tersebut dalam implementasinya akan melibatkan semua data *edge* yang ada dalam graf. Representasi ini juga tidak efektif, karena proses pencarian akan melibatkan semua data *edge*;



1. Contoh data *array* atau *edge list*
2. Representasi *array of adjacency lists*, ini menggunakan indeks *vertex* yang menyimpan *list* dari simpul yang saling berdekatan atau bertetangga. Representasi ini (lihat Gambar II.5) cukup efektif dibanding kedua jenis representasi di atas. Adanya indeks *vertex* memudahkan penelusuran *edge*, karena untuk mengakses *vertex* yang berdekatan cukup dari indeks *array* tersebut.



1. Contoh data *adjacency list* dengan indeks *vertex*

Secara umum representasi data graf menggunakan *adjacency list* lebih disarankan, karena hal ini lebih efektif untuk graf yang kepadatannya rendah, yaitu nilai kedekatan antar *node* berbeda jauh. Untuk *adjacency matrix* lebih disarankan ketika graf yang dibentuk memiliki kepadatan yang tinggi.

## II.5 Hasil Eksplorasi Kakas Visualisasi Program

Tinjauan pustaka terhadap penelitian mengenai pengembangan kakas VP dilakukan untuk memperoleh gambaran mengenai konsep VP dalam representasi visual serta mempelajari arsitektur kakas yang telah dikembangkan. Kemudian dilakukan eksplorasi yang bertujuan untuk menghindari duplikasi riset, sehingga dapat diperoleh sebuah ide baru yang belum pernah dikembangkan sebelumnya.

Pada Tabel II.1 menunjukkan perkembangan kakas VP khusus untuk bahasa pemrograman C dan C++ yang masih aktif pada dekade tahun terakhir ini. Sorva (2012) menjelaskan secara komprehensif perkembangan 40 kakas VP pada Bab 11 disertasinya. Dari semua perkembangan kakas yang masih aktif itu, dipilih hanya kakas VP yang mendukung bahasa pemrograman C dan C++.

1. Perkembangan kakas VP untuk bahasa pemrograman C/C++

| **No.** | **Tahun** | **Nama** | **Dukungan Bahasa** | **Visual Graf** | **Berbasis Web** | **Status** |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | 1996 | GRASP / jGRASP | Ada, Java, C, C++, Objective-C, VHDL | Binary Tree | - | Aktif |
| 2 | 2000 | The Teaching Machine | C++, Java | **-** | **√** | Aktif |
| 3 | 2002 | PlanAni | Pascal, Java, C, Python | - | - | Aktif ? |
| 4 | 2003 | Jeliot 2000 / Jeliot 3 | Java, C, Python | - | - | Aktif |
| 5 | 2004 | OGRE | C++ | - | - | Non-aktif? |
| 6 | 2005 | VIP | C++ | - | - | Aktif |
| 7 | 2005 | ViLLE | Java, C++, Python, PHP, JavaScript, pseudocode | - | - | Aktif |
| 8 | 2010 | Online Python Tutor | Python, Java, C, C++, Ruby, JavaScript | **-** | **√** | Aktif |
| 9 | 2011 | CSmart | C | - | - | Aktif |

Hasil eksplorasi perkembangan kakas VP sejak tahun 1996 hingga terakhir tahun 2011 terlihat pada Tabel II.1. Terdapat tujuh kakas yang masih aktif dan dua kakas yang statusnya tidak jelas. Hanya dua kakas yang mendukung teknologi web. Kakas *The Teaching Machine* (*www.theteachingmachine.org*) memang masih aktif dalam pengembangan. Kekurangan kakas ini adalah membutuhkan konfigurasi lebih lanjut terhadap *plug-in Java* di *browser* pengguna. Hal ini sangat berbeda dengan kakas OPT yang dapat langsung dikunjungi (*pythontutor.com*) dan digunakan secara langsung dari *browser* pengguna tanpa perlu memasang *plug-in* tambahan.

## II.6 Eksplorasi Kakas *Online Python Tutor*

*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 tesis ini dibuat, 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.



1. Tampilan Antarmuka OPT untuk Bahasa Pemrograman C

Pada Gambar II.6 ditunjukkan tampilan antarmuka kakas OPT untuk bahasa pemrograman C. Fungsi nomor 1 adalah untuk berbagi antarmuka bagi pengguna lain yang ingin belajar pemrograman secara berkolaborasi. Ketika tombol tersebut diklik, maka tampilan berubah seperti pada Gambar II.7. Fungsi ini menggunakan kode pustaka dari *TogetherJS*[[2]](#footnote-2) yang dikembangkan oleh *Mozilla Labs* secara bebas dan terbuka. Pengguna lain dapat ikut bergabung dengan mengunjungi pranala yang dibagikan oleh pengguna pertama yang mengaktifkan fitur ini.



1. Tampilan Antarmuka OPT untuk Kolaborasi Pemrograman

Fungsi nomor 2 pada Gambar II.6 adalah untuk memasukkan kode program. Pengguna dapat mulai belajar bahasa pemrograman dengan mengetikkan kode di *form editor* tersebut. Selanjutnya dapat menekan tombol pada nomor 3 untuk melihat simulasi dan visualisasi dari kode program yang telah dibuatnya, seperti pada Gambar II.8 berikut ini.



1. Tampilan Simulasi dan Visualisasi Kode C Pointer

Pada beberapa subbab berikut dijelaskan susunan arsitektur dasar pembangunan kakas OPT yang mendukung proses VP khusus dalam bahasa pemrograman C dan C++.

### II.6.1 Arsitektur Kakas OPT

Penjelasan arsitektur ini berguna untuk memberikan gambaran umum tentang cara kerja atau alur sistem didalamnya, sehingga dapat menentukan metode yang optimal. Selain itu, dapat dijadikan dasar dalam mengembangkan sistem dan menunjang modularisasi pada jangka waktu yang panjang.

Pada Gambar II.9 ditampilkan diagram alur komunikasi antara *browser* milik pengguna dengan *server*. Pertukaran informasi kode program dilakukan dengan fitur permintaan *AJAX* (*Asynchronous JavaScript And XML*).



1. Arsitektur OPT untuk visualisasi kode C dan C++

Untuk lebih jelas tahapan proses pada Gambar II.9 di atas, berikut langkah-langkah yang terjadi ketika pengguna melakukan “Eksekusi Visualisasi” pada kode programnya.

1. Pada antarmuka *browser*, sebenarnya kode tersebut dikirim sebagai tipe data *string* ke *server* dengan metode *AJAX request*.
2. Kemudian *server* mengeksekusi kode tersebut dengan *NodeJS* yang dikombinasikan dengan *server* *Python* untuk menghasilkan *trace* v*algrind*.
3. *Server Python* mengubah format *trace* eksekusi sebagai *JSON* (*JavaScript Object Notation*) sebelum dikembalikan ke *server* *NodeJS*.
4. *JSONP* digunakan untuk memperoleh data dengan permintaan *AJAX* yang berbeda domain atau alamat *IP* (*Internet Protocol*).
5. Respon *AJAX* diterima oleh *visualizer* kemudian dibaca sesuai format yang ditentukan. Maka diperoleh visualisasi kode program yang tampil di *browser*.
6. Ketika pengguna menekan tombol “*Forward*” atau “*Back*” (lihat Gambar II.8) proses visualisasi akan membaca poin indeks yang berada pada *trace* eksekusi sesuai dengan banyak langkah yang telah terbentuk.

Kakas OPT dapat disederhanakan menjadi beberapa inti komponen dengan representasi abstrak seperti terlihat pada Gambar II.10 berikut ini.



1. Ilustrasi Sederhana Proses Kakas OPT

*OPT backend* berfungsi untuk merespon terhadap terbentuknya *trace* eksekusi program. Kemudian *trace* eksekusi diubah menjadi standar format tertentu yang dipaketkan berbentuk *JSON*. *OPT frontend* (antarmuka *browser*) menerjemahkan *JSON* menjadi visualisasi yang dikombinasikan dengan pustaka *D3JS*. Maka visualisasi dapat tampil kepada pengguna seperti pada Gambar II.8.

### II.6.2 *Capturing Execution Trace*

Ketika kode program dikirim ke *OPT backend*, kode C/C++ dieksekusi dengan *valgrind framework*. Kemudian dari *valgrind* terbentuk *trace* yang sudah dimodifikasi oleh pengembangnya. Versi *valgrind* yang digunakan adalah 3.11.0 dengan modifikasi beberapa baris kode agar dapat menghasilkan *trace* eksekusi program. Beberapa berkas yang dimodifikasi adalah sebagai berikut:

|  |  |
| --- | --- |
| 1 | valgrind-3.11.0\memcheck\**mc\_main.c**  (Baris ke 54, 55, 3789, 5693, 5715-5718, 7394-7406, 7646-7647, 7761-7763) |
| 2 | valgrind-3.11.0\include\**pub\_tool\_debuginfo.h**  (Baris ke 36-37, 124-132, 191, 198, 214, 285-408) |
| 3 | valgrind-3.11.0\coregrind\m\_debuginfo\**debuginfo.c**  (Baris ke 74-1458, 5043-5159, 5472, 5594-5719, 5846) |
| 4 | valgrind-3.11.0\memcheck\**mc\_translate.c**  (Baris ke 6261-6481, 6665, 6708-6716, 6795-6811) |
| 5 | valgrind-3.11.0\coregrind\m\_debuginfo\**tytypes.c**  (Baris ke 47-51, 328-864) |
| 6 | valgrind-3.11.0\coregrind\m\_debuginfo\**priv\_tytypes.h**  (Baris ke 160-165) |
| 7 | valgrind-3.11.0\**config.h** (Baris ke 41) |
| 8 | valgrind-3.11.0\memcheck\**mc\_include.h** (Baris 142-156) |

Berkas-berkas *valgrind* tersebut dianalisis perbedaannya dengan menggunakan kakas *WinMerge* (*winmerge.org*). Tekniknya dengan membandingkan *valgrind* versi asli yang diunduh dari *www.valgrind.org*.

Untuk dapat menghasilkan *trace* eksekusi program, kode program dikirim ke *server* yang diolah oleh *ExpressJS*. Kemudian dengan program *Python* dihubungkan dengan *Pipe* ke *terminal* untuk di-*compile* dalam *valgrind debugger*. Dari sini menghasilkan *trace* khusus dari *valgrind* disebut *vgtrace*. Selanjutnya dikonversi ke bentuk *JSON* atau *JSONP* (*JSON with Padding*) oleh *ExpressJS* dan dikembalikan sebagai respon permintaan *AJAX*. Ilustrasi lengkap dapat dilihat pada Gambar II.9.

### II.6.3 *Execution Trace Format*

*OPT back-end* menyimpan informasi *trace* eksekusi dalam standar format tertentu yang kemudian dapat digunakan untuk berinteraksi dengan *visualizer* di *OPT front-end.* Sebagai ilustrasi format *trace* eksekusi program yang dimaksud dapat dilihat pada Gambar II.11 berikut ini.



1. Ilustrasi Format *Trace* Eksekusi Program

Bagan pada Gambar II.11 di atas mengilustrasikan posisi eksekusi kode pada saat masih kosong (*empty state*). *Trace* eksekusi program disimpan dalam bentuk *JSON* yang didefinisikan dalam dua bagian besar, yaitu:

1. “***code***” berbentuk string, yang berisi kode program dari pengguna.
2. “***trace***” berbentuk objek-objek larik, yang setiap objek merepresentasikan posisi eksekusi kode sebagai berikut:

|  |  |  |
| --- | --- | --- |
| 1 | ordered\_globals: [], | // sebuah larik yang merepresen-tasikan urutan visual untuk atribut global |
| 2 | stdout: “”, | // total standar output dari kode program pada posisi eksekusi kode saat ini |
| 3 | func\_name: “<module>”, | // fungsi saat ini yang sedang diekseskusi |
| 4 | stack\_to\_render: [], | // merupakan *list* dari objek-objek, setiap objek direpresentasikan sebagai sebuah *stack frame* |
| 5 | globals: {}, | // sebuah *dictionary global* ‘*stack frame*’ |
| 6 | heap: {}, | // sebuah *dictionary* objek-objek *heap* |
| 7 | line: 1, | // indikasi baris kode yang sedang dieksekusi |
| 8 | event: “step\_line” | // sebuah parameter *event* yang dapat berisi: *user\_call*, *user\_return*, *user\_exception* atau *user\_line* |

### II.6.4 Fitur Pustaka *Data-Driven Documents* (D3)

Teknologi visualisasi utama yang digunakan oleh OPT adalah *Data-Driven Documents* (D3JS) (Bostock dkk., 2011) yang merupakan salah satu pustaka terpopuler untuk implementasi visualisasi berbasis web. Format *trace* eksekusi program dibaca oleh *D3JS* untuk mendeklarasikan pemetaan data dan atribut yang diperlukan oleh elemen *Document Object Model* (DOM) pada laman *Hyper Text Markup Language* (HTML). Peran *D3JS* dapat dilihat pada Gambar II.12 berikut ini.



1. Peran Utama *D3JS* sebagai *Framework* Visualisasi

Pustaka *D3JS* sangat fleksibel terhadap abstraksi data tingkat tinggi. Pustaka ini juga mampu memvisualisasikan berbagai macam data, bahkan cocok untuk visualisasi dengan tingkat kerumitan yang tinggi.

## II.7 Kesimpulan Awal Berdasarkan Studi Literatur dan Eksplorasi

Studi literatur yang telah dilakukan memberikan pemahaman mendasar mengenai terminologi visualisasi dan kegunaannya, metodologi dan prinsip visualisasi data, teori desain interaksi, serta kakas VP yang telah berkembang hingga saat ini. Kakas VP dibuat sebagai alat peraga atau media untuk membantu proses belajar mengajar pemrograman dan ini tidak menjamin pelajar dapat langsung mahir memrogram.

Dari hasil eksplorasi kakas VP, diperoleh bahwa tidak ada satu pun kakas VP berbasis web dengan bahasa C atau C++ yang dapat menampilkan visual 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 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[[3]](#footnote-3). 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.

Untuk membangun sebuah kakas VP, ada beberapa yang perlu dipertimbangkan. Salah satu aspek tersulit dari VP adalah memilah data mana yang dapat direpresentasikan dengan baik dan benar. Secara umum, ini lebih sulit dari pada animasi algoritma tertentu. Aspek perilaku suatu kode program juga harus diperhatikan untuk visualisasi. Pilihan representasi visual untuk lingkungan program juga harus dipertimbangkan. Selain itu, dampak dari visualisasi terhadap perilaku program juga harus dipertimbangkan, karena bisa terjadi salah paham terhadap suatu proses eksekusi kode. Keterbatasan ruang layar komputer juga sering menjadi permasalahan yang perlu diperhatikan dalam desain interaksi visual.

# Bab III Prelimineri Masalah

Bab ini menganalisis masalah kakas VP yang akan dikembangkan, selanjutnya dilakukan penentuan ruang lingkup pengembangan kakas terhadap proses visualisasi graf dan peluang pemecahan masalah.

## III.1 Analisis Masalah

Langkah awal sebelum pengembangan adalah eksplorasi kakas OPT yang dilakukan dengan meninjau proses kerja sistem, alur kode program dan membaca baris-baris kode pengembangnya. Versi terkahir yang dikembangkan telah bermigrasi ke bahasa pemrograman *Typescript*. Pengembang berargumen agar dapat dikembangkan lebih lanjut dengan teknik modularisasi[[4]](#footnote-4). Ini bermanfaat ketika peneliti atau pengembang lain yang ingin melanjutkan dengan penambahan fitur dapat dilakukan dengan mudah.

Permasalahan utama pada penelitian ini adalah menemukan data graf pada kode untuk dapat dilakukan visualisasi sebagaimana persepsi visual graf itu beroperasi. Perhatikan pada Gambar III.1 terlihat bahwa untuk merepresentasikan graf (salah satu contoh kode graf *Breadth First Search*) masih dilakukan dengan metode primitif, yaitu berupa tabel data larik atau matriks. Selain itu, pada Gambar III.2 yang menggunakan *pointer* sebagai abstraksi untuk relasi antar simpul, tidak membuat menjadi lebih jelas relasinya karena saling tumpang tindih visual yang ditampilkan. Tentu ini membuat pengguna untuk menelusuri kode dengan kecocokan visualnya menjadi lebih rumit.

Perubahan visual antar setiap langkah pada baris kode kurang interaktif, karena setiap ada perubahan data dari langkah sebelum dan sesudahnya tidak ada perubahan warna atau penanda khusus sebagai bentuk atensi kepada pengguna. Adanya proses ini akan memudahkan proses kognisi pengguna dalam menangkap informasi yang tampil pada visual tersebut.



1. Visual primitif berupa tabel matrik



1. Visual primitif berupa *struct* dan *pointer*

Data *JSON execution trace* adalah sumber data yang digunakan untuk visualisasi. Salah satu contoh data *JSON* berupa matriks yang dihasilkan dari Gambar III.1 dapat dilihat pada Tabel III.1 berikut ini. Data ini adalah contoh variabel *‘a’* berupa matriks yang memiliki dimensi 7 x 7. Nilai berupa "C\_MULTIDIMENSIONAL\_ARRAY" adalah jenis data *n-array*; nilai baris kedua "0x601100" adalah posisi alamat memori yang digunakan; kemudian ukuran dimensi yang digunakan berada di posisi ketiga [7, 7]. Sisa baris berikutnya adalah isi data setiap elemen di dalam variabel matriks tersebut.

1. Variabel *‘a’* berupa matriks berdimensi 7 x 7

|  |
| --- |
| "a": [  "C\_MULTIDIMENSIONAL\_ARRAY",  "0x601100", [  7,  7  ],  [  "C\_DATA",  "0x601100",  "int",  0  ],  [  "C\_DATA",  "0x601104",  "int",  0  ],  [...]  ] |

Sedangkan untuk data *JSON* dari Gambar III.2 dapat dilihat pada Gambar III.3. Objek *heap* ini berisi alamat memori yang digunakan oleh *pointer* dan *struct*. Pencarian pola visualisasi akan dijelaskan lebih lanjut pada subbab III.2. Untuk memudahkan dalam proses pengolahan data *JSON,* maka digunakan bantuan kakas web yang dapat diakses melalui alamat http://www.json2table.com. Kakas ini membantu mengolah dan mencari pola visualisasi dalam data *JSON* dengan mengonversi ke bentuk tabel dan daftar poin-poin.



1. Contoh data *JSON pointer* pada *heap*

## III.2 Hipotesis Solusi Penyelesaian Masalah

Pada Tabel III.1 dan Gambar III.3 menunjukkan hasil data *JSON trace* yang akan dilakukan proses visualisasi data sesuai pada subbab II.2 tentang tujuh tahapan proses konstruksi visualisasi data. Dari data *JSON* dapat dibagi menjadi dua kategori utama untuk representasi graf dalam kode, yaitu (1) bersumber dari tabel informasi berupa matriks atau *n-*dimensi *array*, dan (2) berupa *struct* yang berisi *pointer*.

1. Isi alamat memori pada objek *heap*

|  |
| --- |
| "0x5402210": [  "C\_ARRAY",  "0x5402210", [  "C\_STRUCT",  "0x5402210",  "node", [  "vertex", [  "C\_DATA",  "0x5402210",  "int",  2  ]  ],  [  "next", [  "C\_DATA",  "0x5402218",  "pointer",  "0x5402170"  ]  ]  ]  ] |

Untuk dapat melakukan visualisasi secara otomatis, diperlukan fitur tertentu dari data-data tersebut. Ada beberapa fitur yang bisa menjadi acuan, yaitu :

* "C\_MULTIDIMENSIONAL\_ARRAY" digunakan untuk menemukan variabel data matrik yang menyimpan nilai-nilai simpul dan relasinya serta bobot suatu graf (akan dibahas lebih lanjut pada Bab IV);
* "C\_STRUCT" digunakan untuk menemukan variable *struct* yang masih harus dilanjutkan dengan pengecekan fitur ketiga berikut;
* "pointer" di dalam "C\_DATA" yang mengindikasikan adanya relasi antar simpul dengan menggunakan *pointer* di dalam *struct*.

Fitur-fitur tersebut merupakan kunci utama, selain harus dicari pola visual lain seperti variabel pendukung lain yang sedang digunakan dalam kode program.

Visualisasi graf memiliki beberapa properti yang dibutuhkan untuk menampilkan hasil pengolahan data dari *JSON trace*. Pada Tabel III.3 menunjukkan berbagai jenis properti visual graf yang mungkin dapat direpresentasikan menjadi visualisasi. Dari daftar itu tidak semua properti visual digunakan, harus dapat menyesuaikan data yang sedang diproses dari eksekusi kode.

1. Properti visual graf untuk visualisasi

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Graf** | **Pohon** | **Nama Properti** | **Visualisasi** |
| 1 | √ | √ | Label Simpul (*Vertex*) | x  atau |
| 2 | √ | √ | Bentuk Simpul (*Vertex*) | atau atau |
| 3 | √ | √ | Bentuk Sisi (*Edge*) | atau atau |
| 4 | √ | X | Label Sisi | x  x  atau |
| 5 | √ | √ | Warna (*Vertex* dan *Edge*) | atau |
| 6 | √ | √ | Ketebalan Garis (*Vertex* dan *Edge*) | atau |

## III.3 Penunjang Pengembangan Kakas

Untuk menunjang pengembangan kakas diperlukan seperangkat kebutuhan yang harus dipenuhi, baik dari sisi perangkat lunak yang mutakhir maupun perangkat kerasnya. Subbab berikut akan merinci spesifikasi kebutuhan sistem.

### III.3.1 Kebutuhan Perangkat Lunak

*Typescript* merupakan bahasa pemrograman berbasis objek dari *JavaScript* yang cocok untuk pengembangan kakas OPT. Bahasa ini dibuat oleh Microsoft pada tahun 2012 untuk menunjang pengembangan aplikasi web *JavaScript* berskala besar dengan konsep pemrograman berorientasi objek.

Rincian kakas dan *library* pendukung untuk pengembangan kakas OPT yang telah terintegrasi dapat dilihat pada Tabel III.4 berikut ini.

1. Kakas dan *library* pendukung pengembangan OPT

| **No.** | **Nama** | **Versi** | **Keterangan** |
| --- | --- | --- | --- |
| 1 | NodeJS | 4.5.0 | *JavaScript runtime* |
| 2 | Npm | 2.15.9 | *Node package manager*, memanajemen *library* dari *node* |
| 3 | Webpack | 3.5.5 | *Bundle js, ts, css, image* |
| 4 | Typescript | 1.8.10 | Bahasa pemrograman utama |
| 5 | Typings | 2.1.1 | *Typescript definitions* |
| 6 | Ace Code Editor | 1.2.8 | *Text editor* pada aplikasi web untuk *input* kode program |
| 7 | D3JS | 2.0 | Pustaka *Data-Driven Documents* |
| 8 | jQuery | 3.0.0 | *Cross-platform JavaScript library* |
| 9 | jQuery.bbq | 1.3pre | *Back Button and Query Library* |
| 10 | jQueryUI | 1.11.4 | *JavaScript Library* untuk interaksi antarmuka pengguna |
| 11 | jQuery.qtip | 2.0.0 | *jQuery tooltip plugin* |
| 12 | RequireJS | 2.1.20 | *JavaScript module loader*. *Framework* untuk manajemen dependensi. |
| 13 | jQuery.jsPlumb | 1.3.10 | *Library* visual *connectivity* untukaplikasi web |
| 14 | jQuery.simplemodal | 1.4.4 | *Lightweight jQuery plugin* untuk antarmuka *modal* dialog |

### III.3.2 Kebutuhan Perangkat Keras

OPT dapat berjalan dengan baik pada komputer dengan spesifikasi yang direkomendasikan sebagai berikut:

* *Processor core i3*
* *RAM 2GB*
* Sistem operasi Ubuntu 16.04.

Kebutuhan perangkat itu untuk proses pengembangan kakas, namun untuk produksi versi publik yang dipasang pada komputer server, dibutuhkan spesifikasi RAM sekitar 4GB – 8GB. Hal ini karena sistem *sandbox docker engine* membutuhkan ruang memori lebih jika memroses sekitar 500 baris kode untuk dapat menghasilkan data *trace* eksekusi *JSON*.

# DAFTAR PUSTAKA

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

Bostock, M., Ogievetsky, V. dan Heer, J. (2011): D3 data-driven documents, *IEEE Trans. Vis. Comput. Graph.*, **17**, 2301–2309.

Cetin, I. dan Andrews-Larson, C. (2016): Learning sorting algorithms through visualization construction, *Comput. Sci. Educ.*, **26**, 27–43.

Diehl, S. (2007): *Software visualization: visualizing the structure, behaviour, and evolution of software ; with 5 tables*, Berlin, Springer.

Fry, B. (2008): *Visualizing Data*, First Edition, USA, O`Reilly Media, Inc.

Gračanin, D., Matković, K. dan Eltoweissy, M. (2005): Software visualization, Innov. Syst. Softw. Eng., 1, 221–230.

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.

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).

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).

Preece, J. (2002): *Interaction Design: Beyond Human-Computer Interaction, First Edition*, First Edition, USA, John Wiley & Sons, Inc.

Sedgewick, R. dan Wayne, K. (2011): *Algorithms Fourth Edition*, Fourth Edition, USA, Pearson Education, Inc.

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

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.

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://togetherjs.com/ [↑](#footnote-ref-2)
3. https://github.com/pgbovine/OnlinePythonTutor [↑](#footnote-ref-3)
4. https://github.com/pgbovine/OnlinePythonTutor [↑](#footnote-ref-4)