**PENGEMBANGAN KAKAS VISUALISASI PROGRAM BERBASIS WEB UNTUK GRAF SEBAGAI**

**PEMBELAJARAN ALGORITMA PEMROGRAMAN**

**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**

**September 2016**

**PENGEMBANGAN KAKAS VISUALISASI PROGRAM BERBASIS WEB UNTUK GRAF SEBAGAI**

**PEMBELAJARAN ALGORITMA PEMROGRAMAN**

Oleh

**Habibie Ed Dien**

**NIM: 23515043**

**(Program Studi Magister Informatika)**

Institut Teknologi Bandung

Menyetujui

Pembimbing

Tanggal September 2016

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Yudistira Dwi Wardhana Asnar, S.T., Ph.D.)

# DAFTAR ISI

HALAMAN PENGESAHAN i

[DAFTAR ISI ii](#_Toc466288855)

[DAFTAR GAMBAR DAN ILUSTRASI iv](#_Toc466288856)

[DAFTAR TABEL v](#_Toc466288857)

[Bab I Pendahuluan 1](#_Toc466288858)

[I.1 Latar Belakang 1](#_Toc466288859)

[I.2 Rumusan Masalah 5](#_Toc466288860)

[I.3 Tujuan 5](#_Toc466288861)

[I.4 Batasan Masalah 5](#_Toc466288862)

[I.5 Metodologi Penelitian 6](#_Toc466288863)

[I.6 Sistematika Penulisan 7](#_Toc466288864)

[I.7 Rencana Waktu Penyelesain Tesis 8](#_Toc466288865)

[Bab II Tinjauan Pustaka 9](#_Toc466288866)

[II.1 Struktur Data 9](#_Toc466288867)

[II.1.1 Graf 9](#_Toc466288868)

[II.1.2 Pohon 12](#_Toc466288869)

[II.2 Revolusi Teknologi Web terhadap Pembelajaran 13](#_Toc466288870)

[II.3 Terminologi Visualisasi Perangkat Lunak 14](#_Toc466288871)

[II.4 Hasil Eksplorasi Kakas Visualisasi Program 16](#_Toc466288872)

[II.5 Pentingnya Belajar Eksekusi Kode dalam Pemrograman 17](#_Toc466288873)

[II.6 Matrix Framework 18](#_Toc466288874)

[II.6.1 Visualisasi 18](#_Toc466288875)

[II.6.2 Struktur 19](#_Toc466288876)

[II.6.3 Konstruksi Visual 20](#_Toc466288877)

[II.6.4 Animasi 21](#_Toc466288878)

[II.6.5 Simulasi 21](#_Toc466288879)

[II.6.6 Tampilan Antarmuka Pengguna 22](#_Toc466288880)

[II.7 Kesimpulan Awal Berdasarkan Studi Literatur 22](#_Toc466288881)

[Bab III Analisa Awal 23](#_Toc466288882)

[DAFTAR PUSTAKA 25](#_Toc466288883)

# DAFTAR GAMBAR DAN ILUSTRASI

Gambar I.1  [Dokumentasi penggunaan kakas visualisasi ... (Guo, 2013) **Error! Bookmark not defined.**](#_Toc457811209)

Gambar II.1 [Salah satu contoh kakas visualisasi ... (Sorva, 2012). **Error! Bookmark not defined.**](#_Toc457811211)

Gambar III.1 [Contoh visualisasi *Online Python Tutor* (Guo, 2013) 15](#_Toc457811213)

# DAFTAR TABEL

Tabel I. 1 [Rincian kegiatan penelitian tesis 8](#_Toc457811602)

# Bab I Pendahuluan

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

## I.1 Latar Belakang

Graf adalah salah satu pokok bahasan perkuliahan Algoritma Pemrograman dan Struktur Data dalam bidang informatika atau ilmu komputer (Valiente, 2002). Graf merupakan topik penting yang selalu ada dalam perkuliahan Struktur Data. Representasi objek visual dari graf dinyatakan sebagai noktah, bulatan, atau titik, sedangkan hubungan antar objek dinyatakan dengan garis. Graf bermanfaat untuk memodelkan suatu masalah dengan cara merepresentasikan objek-objek diskrit dan hubungan antar objek tersebut. Contoh manfaat graf dalam pemodelan dapat dilihat pada jaringan komunikasi, jaringan komputer, analisis algoritma dan peta. Graf dalam pemrograman digunakan untuk memodelkan sesuatu yang abstrak, seperti aliran kerja dalam proyek, perencanaan dan manajemen proyek, pencarian rute terpendek, perpindahan dalam permainan (*game*) serta dalam bidang kecerdasan buatan (*artificial intelligence*).

Pohon (*tree*) merupakan salah satu bentuk khusus dari struktur suatu graf (Valiente, 2002). Suatu graf terhubung yang setiap pasangan simpulnya hanya dapat dihubungkan oleh suatu lintasan tertentu, maka graf tersebut dinamakan pohon (*tree*). Pohon (*tree*) merupakan graf tak-berarah yang terhubung dan tidak memiliki sirkuit. Struktur pohon merupakan topik struktur data yang penting dalam informatika (Inggriani Liem, 2008). Pohon bermanfaat untuk mengorganisasi informasi berdasarkan suatu struktur logika dan memungkinkan cara akses yang khusus terhadap suatu elemen. Contoh manfaat pohon dalam memodelkan suatu masalah, seperti pohon keputusan, pohon keluarga, klasifikasi botani, pohon sintaks, dan pohon ekspresi aritmatika.

Pada abad 21 ini, memiliki kemampuan dalam pemrograman merupakan keahlian yang sangat bernilai (Helminen dan Malmi, 2010; Margulieux dkk., 2016). Oleh sebab itu banyak pelajar ingin memasuki informatika atau bidang komputer lainnya (Cetin, 2016; Piteira dan Costa, 2013). Namun, situasi tersebut tidak diimbangi dengan kualitas kemampuan pemrograman yang harus dimiliki. Untuk mempelajari pemrograman, pelajar harus memiliki pengetahuan yang baik tentang deklarasi dan prosedural program, daya ingat yang kuat, pemahaman, penyelesain masalah, abstraksi dan kemampuan berpikir logik (Piteira dan Costa, 2013).

Riset yang ditulis oleh Piteira dan Costa pada tahun 2013 berisi survei tentang identifikasi permasalahan dalam perkuliahan pemrograman. Riset tersebut bertujuan untuk mengidentifikasi tingkat kesulitan belajar paling populer di kalangan pelajar. Aspek yang dianalisa adalah konten materi dari perkuliahan dan aspek pembelajarannya. Ternyata diperoleh data bahwa konsep pemrograman yang paling sulit adalah tipe struktur data, selain pointer dan referensi serta parameter. Struktur data merupakan subjek penting dalam bidang komputasi, namun pelajar masih mengalami kesulitan dalam memahami, mengeksekusi dan mengonsep algoritmanya (Pathania dan Singh, 2014). Struktur data dapat dibagi menjadi dua bagian, yaitu struktur data linear dan non-linear. Struktur data linear terdapat larik, antrian, tumpukan dan *linked list*, sedangkan non-linear terdapat pohon dan graf. Kemudian untuk hasil analisa dari aspek pembelajaran, bahwa yang paling signifikan pelajar sangat terbantu melalui contoh-contoh eksekusi kode program, karena merupakan implementasi dari konsep yang dipelajarinya.

Visualisasi telah banyak direkomendasikan menjadi kakas penting untuk mengatasi kesulitan pelajar dalam memahami konsep pemrograman (Cetin dan Andrews-Larson, 2016; Sorva, 2012; Sorva dkk., 2013; Gračanin dkk., 2005; Guo, 2013). Melalui interaksi visual, manusia lebih cenderung menangkap lebih banyak informasi yang diterima dibandingkan melalui indera lainnya (Ware, 2004). Oleh sebab itu visualisasi telah menjadi bagian penting dalam proses belajar mengajar pemrograman (Sorva dkk., 2013), sehingga menjadi solusi dalam proses pembelajaran (Guo, 2013). 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 dari konsep langkah kerja perangkat lunak, sedangkan VP terkait pada cara kerja eksekusi kode program dan proses struktur datanya. Melalui VP dapat memberikan pemahaman lebih baik kepada pelajar mengenai implementasi dari materi ajar pemrograman (Sorva, 2012).

Bahasa pemrograman C dan C++ digunakan dalam perkuliahan struktur data teknik informatika di STEI ITB (Inggriani Liem, 2008). Kedua bahasa pemrograman itu dipilih karena merupakan bahasa yang banyak digunakan dalam industri perangkat lunak saat ini. Oleh sebab itu, kedua bahasa pemrograman ini dipilih untuk digunakan terhadap kakas VP yang akan dikembangkan.

Teknologi web dan internet saat ini telah banyak mengubah cara belajar di dunia (Bonk, 2009). Perkembangan pesat teknologi web memberi kemudahan kepada pelajar untuk meningkatkan kemampuannya di mana dan kapan pun. *E-learning* menjadi portal untuk belajar bagi siapapun tanpa kendala jarak. Melalui teknologi ini, kakas VP yang akan dikembangkan dapat dengan mudah dan langsung digunakan oleh pelajar mana pun di dunia.

Perkembangan VP sebanyak 40 kakas dijelaskan pada Bab 11 disertasi Sorva (2012) sejak tahun 1983-an. Namun kakas VP berbasis web untuk bahasa C dan C++ masih sangat langka. Pada umumnya pengembangan kakas VP ini dibangun dengan *Java Virtual Machine* (JVM) (Helminen dan Malmi, 2010). *The Teaching Machine* (*www.theteachingmachine.org*) merupakan kakas VP berbasis *Java Applet* web yang mendukung bahasa C++ dan Java. Kekurangannya butuh waktu lagi bagi pelajar untuk memasang *plugin Java applet*, terutama saat digunakan di ruang laboratorium komputer yang berjumlah banyak. Alasan para pengembang menggunakan Java karena dapat beroperasi pada berbagai sistem operasi.

*Online Python Tutor* (OPT) (Guo, 2013) merupakan kakas VP yang pada awalnya ditahun 2010 dikembangkan untuk bahasa Python. Namun saat ini telah mendukung banyak bahasa pemrograman seperti Java, C, C++, Ruby dan JavaScript. Kelebihan kakas VP ini cukup sederhana untuk menggunakannya. Pelajar hanya mengunjungi situs *pythontutor.com* pada browser di perangkat komputer atau piranti bergerak seperti *smartphone* dan *tablet*. Kemudian pelajar bisa langsung mengeksekusi kode program yang sudah ada contohnya di web tersebut atau dengan cara mengetiknya sendiri pada kode editor yang disediakan. Namun, kakas ini masih terbatas pada visualisasi struktur data linear, sehingga untuk mendukung pembelajaran pemrograman graf belum bisa dilakukan.

*Jype* (Helminen dan Malmi, 2010; Sorva, 2012) merupakan kakas VP yang menggunakan metode *Matrix framework* untuk memvisualkan bahasa Python secara otomatis dalam program struktur data linear dan non-linear, seperti larik dan pohon. Kakas ini dibangun dengan Java dan beroperasi di web. Namun kakas ini sudah tidak dikembangkan lagi dan kode sumbernya tidak ditemukan di internet (Guo, 2013; Sorva, 2012). *Matrix framework* (Korhonen dkk., 2004) dibangun berdasar empat konsep visual, yaitu *visual container*, *visual component*, *visual reference*, dan *visual data*. *Visual container* berupa struktur yang kompleks, menyimpan nilai variabel (node, indeks, dan lainnya) yang saling terhubung dengan cara tertentu. Kemudian tiap variabel di dalam *visual container* terdapat *visual component* yang berguna untuk memvisualisasikan variabel. *Visual reference* sebagai penghubung antar *visual component* dan *visual data* yang berisi tipe data primitif atau struktur data yang lebih kompleks dengan atribut-atributnya.

Saat ini kakas VP berbasis web untuk C/C++ yang mampu melakukan visual graf belum tersedia. Sehingga pada penelitian ini, penulis akan mengembangkan kakas VP untuk graf berbasis web dari kode sumber OPT, karena selain telah mendukung teknologi web untuk *e-learning*, kakas ini bersifat bebas (*free*) dan bersumber terbuka (*open source*) (Guo, 2013). Agar dapat melakukan visualisasi graf secara otomatis ketika eksekusi kode program C/C++, maka kakas VP akan dipadukan dengan metode *Matrix framework*.

## I.2 Rumusan Masalah

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

1. Apakah penggunaan *Matrix framework* cocok terhadap pengembangan kakas VP berbasis OPT untuk graf dalam bahasa C dan C++ ?
2. Bagaimana mengembangkan kakas VP berbasis OPT dengan menggunakan *Matrix framework* untuk graf dalam bahasa C dan C++ ?
3. Apakah penggunaan *Matrix framework* dapat meningkatkan kinerja VP ?
4. Bagaimana kakas dapat mendeteksi adanya kode program graf atau pohon sehingga proses visualisasi sesuai dengan algoritmanya ?
5. Apakah metode *Matrix framework* dapat dikembangkan untuk mendukung visualisasi graf ?
6. Bagaimana peran VP yang dikembangkan terhadap proses pembelajaran algoritma dan pemrograman untuk graf ?

## I.3 Tujuan

Sebagai solusi dari masalah yang telah diuraikan, maka tujuan dari penelitian tesis ini adalah :

1. Menghasilkan kakas VP berbasis web untuk bahasa C dan C++ yang dapat menunjang proses pembelajaran pemrograman untuk graf.
2. Menghasilkan modifikasi *Matrix framework* yang dapat mendukung VP untuk graf sesuai algoritma yang sedang digunakan.

## I.4 Batasan Masalah

Latar belakang dan rumusan masalah telah diuraikan, maka untuk memfokuskan kegiatan penelitian tesis ini diperlukan batasan, sehingga cakupan pembahasan dan penyelesaian tidak meluas. Adapun batasan masalah pada tesis ini adalah :

1. Pengembangan kakas VP fokus terhadap visual graf.
2. Implementasi kakas VP berdasar kode sumber dari OPT.
3. Bahasa pemrograman yang dapat dieksekusi oleh kakas adalah C dan C++.
4. Kode program yang dapat divisualisasi minimal memiliki syarat terbentuknya data objek dalam graf.

## I.5 Metodologi Penelitian

Penelitian dimulai dengan studi literatur, eksplorasi, dan identifikasi masalah untuk memformulasikan pertanyaan-pertanyaan yang berkaitan dengan permasalahan belajar pemrograman graf dan perkembangan kakas VP. Tinjauan pustaka dari penelitian sebelumnya, buku, jurnal dan artikel ilmiah, dan situs internet yang berelasi dengan pengembangan kakas VP serta konsep dalam pembelajaran algoritma pemrograman graf, sehingga penelitian pada tesis ini akan dilaksanakan melalui beberapa tahapan sebagai berikut.

1. Studi Literatur dan Eksplorasi

Tahap ini dilakukan pendalaman mengenai konsep umum pembelajaran pemrograman, pedagogi, VP, proses cara berpikir komputer (*computational thinking*) (Sorva, 2012), serta implementasi dari kakas yang telah dibangun. Studi literatur menghasilkan kesimpulan awal yang dapat dijadikan landasan dalam perumusan masalah dan pengembangan kakas. Kemudian dilakukan eksplorasi terhadap kakas-kakas VP yang telah ada.

1. Analisis Masalah dan Perancangan Sistem

Tahap ini dilakukan kajian terhadap masalah yang diteliti kemudian dirumuskan hipotesis untuk disusun solusi yang mungkin dapat diterapkan. Selanjutnya, akan dirumuskan suatu skema untuk melakukan eksplorasi kode program dan mengonversi kode tersebut menjadi model animasi dalam sebuah tampilan grafis diagram pohon dan graf.

1. Pengembangan Kakas

Hasil rancangan sistem pada tahap sebelumnya akan diimplementasikan dalam pembangunan kakas VP berbasis web, kemudian akan dilakukan pengujian dan evaluasi dari hasil implementasi tersebut.

1. Pengujian dan Evaluasi Kakas

Hasil pengembangan kakas perlu diuji dengan memasukkan berbagai jenis kode program dan tipe data, kemudian menganalisis hasil dari visualisasi tersebut. Selain itu, kakas perlu dievaluasi secara empiris dengan bantuan responden untuk menilai efektivitas kakas dalam membantu pemahaman terhadap kode program yang sedang dipelajarinya.

## I.6 Sistematika Penulisan

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

1. Bab I Pendahuluan

Bab ini berisi latar belakang, rumusan masalah, tujuan, serta metodologi yang digunakan pada pengerjaan tesis, serta sistematika pembahasan dari laporan tesis.

1. Bab II Tinjauan Pustaka

Bab ini berisi uraian tentang konsep belajar mengajar pemrograman, peran dan perkembangan kakas visualisasi dari hasil penelitian sebelumnya yang berkaitan dengan masalah yang dikaji, sehingga memberikan gambaran perkembangan terhadap masalah yang akan diteliti.

1. Bab III Analisis dan Perancangan

Bab ini berisi analisis terhadap masalah yang akan diteliti, kemudian dirumuskan hipotesis untuk disusun solusi yang mungkin dapat diterapkan, sehingga untuk mendukung hipotesis tersebut, dibangun pula rancangan dari sistem yang akan dikembangkan.

1. Bab IV Pengembangan Kakas

Bab ini menjelaskan proses pengembangan dan implementasi kakas visualisasi kode program berdasarkan rancangan yang telah dibangun sebelumnya pada Bab III.

1. Bab V Pengujian dan Evaluasi Kakas

Bab ini menguraikan proses dan hasil pengujian serta evaluasi dari kakas yang dikembangkan pada tesis ini. Pengujian dilakukan untuk membuktikan bahwa kakas berhasil dikembangkan lebih baik dibandingkan sebelumnya. Evaluasi kakas dilakukan untuk menilai efektivitas dan kualitas dari visualisasi untuk mendukung pemahaman dalam pembelajaran algoritma pemrograman.

1. Bab VI Penutup

Bab ini berisi kesimpulan dalam tesis serta saran untuk pengembangan kakas lebih lanjut dari penelitian tesis ini.

# Bab II Tinjauan Pustaka

Pada bab ini dijelaskan mengenai ruang lingkup struktur data graf dan algoritmanya, terminologi visualisasi perangkat lunak, peran kakas VP dalam proses pembelajaran pemrograman, dan skema detail dari *Matrix framework*.

## II.1 Struktur Data

Graf dan pohon merupakan salah satu pokok bahasan dalam Matematika Diskrit dan konsep dasar yang harus dipahami oleh pelajar dalam bidang informatika atau ilmu komputer. Dalam perkuliahan Algoritma Pemrograman dan Struktur Data, terdapat pula topik tentang graf, namun lebih cenderung kepada implementasi terhadap pemrogramannya. Karena memang pelajar bidang informatika atau ilmu komputer dituntut untuk dapat mampu membuat kode program dalam menyelesaikan suatu permasalahan tertentu.

Berikut ini akan dijelaskan tentang sejarah graf dan pohon beserta algoritma dan model visualnya.

### II.1.1 Graf

Sejarah mencatat bahwa model graf diterapkan pertama kali sekitar Abad 17 pada tahun 1736, yaitu permasalahan jembatan Königsberg (lihat Gambar II.1). Kota Königsberg (sebelah timur Prussia, Jerman sekarang) yang berubah nama menjadi kota Kaliningrad. Di kota itu terdapat sungai Pregal yang mengalir mengelilingi pulau Kneiphof lalu bercabang menjadi dua buah anak sungai (Rinaldi Munir, 2010).

|  |  |
| --- | --- |
|  |  |
| (a) | (b) |

1. (a) Jembatan Königsberg; (b) Model grafnya (Harary, 1969)

Pada Gambar II.1(a) terdapat tujuh buah jembatan yang saling menghubungkan antar daratan yang dialiri sungai tersebut. Permasalahannya adalah apakah mungkin dapat melalui ketujuh jembatan tersebut tepat satu kali dan kembali ke tempat semula ?. Beberapa penduduk kota tersebut sepakat bahwa tidak mungkin mereka dapat melalui ketujuh jembatan sekali dan kembali ke tempat asal keberangkatan. Tetapi mereka tidak dapat menjelaskan secara ilmiah alasannya. Euler adalah seorang matematikawan Swiss yang berhasil menemukan jawaban terhadap masalah tersebut pada tahun 1736. Ia memberikan bukti sederhana dengan memodelkan masalah ini ke dalam model graf. Titik (noktah) menyatakan daratan yang dihubungkan oleh jembatan, yang disebut simpul (*vertex*). Garis merepresentasikan jembatan, yang disebut sisi (*edge*). Setiap titik terdapat label huruf A, B, C, dan D. Euler membuat model graf seperti terlihat pada Gambar II.1(b).

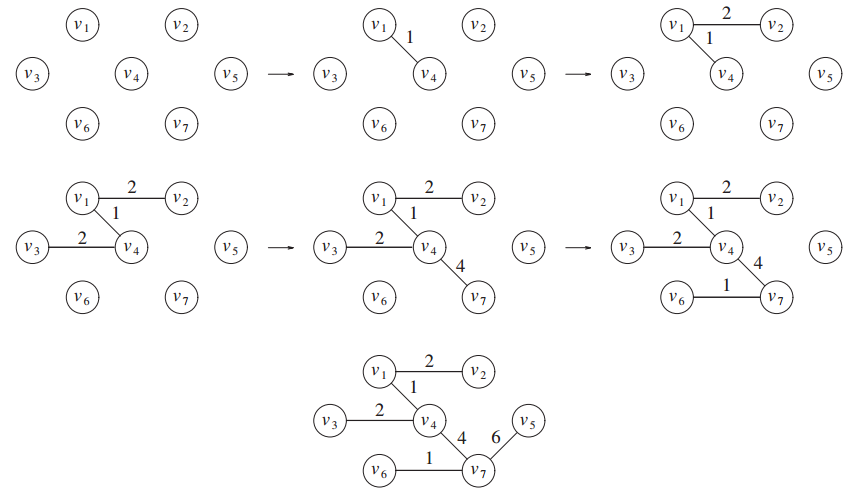
Euler mengemukakan jawabannya bahwa orang tidak mungkin melalui ketujuh jembatan sekali dan kembali ke tempat asal keberangkatan jika derajat setiap simpul tidak seluruhnya genap. Derajat merupakan banyak garis yang bersisian dengan noktah. Contoh simpul C memiliki derajat tiga karena ada tiga buah garis yang bersisian dengannya, simpul B dan D juga berderajat tiga, sedangkan simpul A berderajat lima. Karena semua simpul tidak berderajat genap, maka tidak mungkin dilakukan perjalanan berupa sirkuit (yang selanjutnya disebut sirkuit Euler) pada graf tersebut.

Graf memiliki berbagai macam algoritma dan model visualnya masing-masing, sehingga perlu dijabarkan lebih rinci agar mengetahui konsep dasar terhadap proses kerja dari kakas VP yang akan dibangun. Macam algoritma graf yang digunakan dalam bidang komputer beserta model visualnya dapat dilihat pada Tabel II.1 berikut. Algoritma dan model visual graf yang dijelaskan berikut merupakan yang paling sering digunakan dalam perkuliahan Struktur Data dan Algoritma Pemrograman di STEI ITB (Inggriani Liem, 2008).

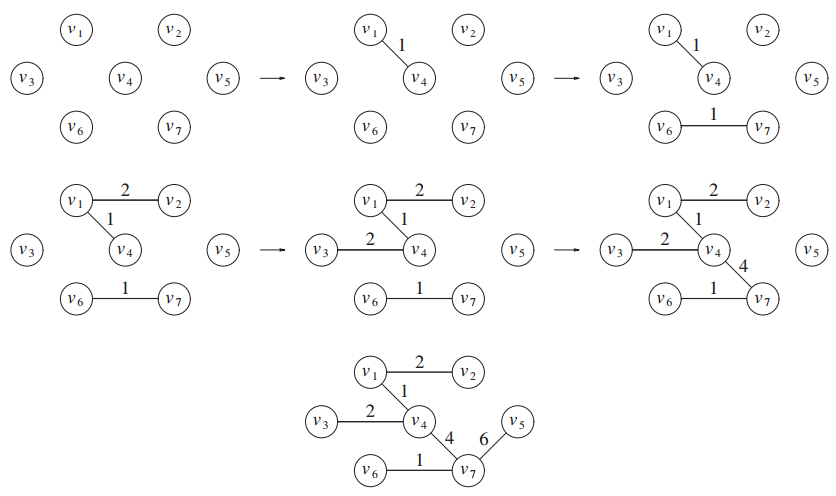
1. Daftar nama algoritma graf dan model visualnya (Valiente, 2002)

| **No.** | **Nama Algoritma Graf** | **Model Visual** |
| --- | --- | --- |
| 1 | Breadth First Search (BFS) |  |
| 2 | Depth First Search (DFS) |  |
| 3 | Prim |  |
| 4 | Djikstra |  |
| 5 | Bellman-Ford | http://2.bp.blogspot.com/-GG_lKtfmonw/UYi73BodmvI/AAAAAAAAAjw/UGlBfBpEKI4/s400/nodenode.png |
| 6 | Floyd-Warshall |  |
| 7 | Kruskal |  |

Tahapan dari ketujuh algoritma tersebut akan dijelaskan lebih rinci pada bagian berikut.



Gambar ... Langkah algoritma Prim



Gambar ... Langkah algoritma Kruskal

### II.1.2 Pohon

Pohon (Rinaldi Munir, 2010) merupakan graf terhubung yang tidak mengandung sirkuit. Teori pohon (*tree*) merupakan konsep paling penting, karena memiliki banyak terapan di berbagai bidang ilmu komputer maupun bidang lainnya. Konsep ini telah banyak dikaji secara intensif sebagai objek matematika dan terapan dalam kehidupan manusia sehari-hari. Banyak orang menggunakan pohon untuk membuat hirarki seperti silsilah keluarga, struktur organisasi, organisasi pertandingan olahraga, dan lain-lain. Pohon parsing (*parse tree*) digunakan oleh para ahli bahasa untuk menguraikan kalimat.

Sejak tahun 1857, istilah pohon telah lama digunakan oleh matematikawan asal Inggris bernama Arthur Cayley. Ia memanfaatkan pohon untuk menghitung jumlah senyawa kimia. Pohon sebagai struktur data rekursif yang merupakan bagian penting dari perkuliahan Struktur Data.

Untuk pembahasan macam algoritma pohon yang digunakan dalam bidang komputer beserta model visualnya dapat dilihat pada Tabel II.2 berikut ini.

1. Daftar nama algoritma pohon dan model visualnya (Valiente, 2002)

| **No.** | **Nama Algoritma Pohon** | **Model Visual** |
| --- | --- | --- |
| 1 | Binary Search Tree (BST) |  |
| 2 | Digital Search Tree |  |
| 3 | Radix Tree | https://upload.wikimedia.org/wikipedia/commons/thumb/6/63/An_example_of_how_to_find_a_string_in_a_Patricia_trie.png/220px-An_example_of_how_to_find_a_string_in_a_Patricia_trie.png |
| 4 | AVL (Adelson-Velskii dan Landis) *Tree*; terdiri dari *Single Rotation* dan *Double Rotation* |  |
| 5 | Red-Black Tree | Diagram of binary tree. The black root node has two red children and four black grandchildren. The child nodes of the grandchildren are black nil pointers or red nodes with black nil pointers. |
| 6 | Search Tree Traversal; terdiri dari *pre-order*, *in-order*, dan *post-order* | Hasil gambar untuk tree traversal algorithm |
| 7 | Splay Tree |  |

Dari penjelasan berbagai macam algoritma dan model-model visual graf tersebut, dapat ditarik kesimpulan secara umum representasi visualisasi yang dapat dilakukan hanya dua bagian besar (lihat Gambar II.2).

|  |  |
| --- | --- |
|  |  |
| (a) | (b) |

1. Model visual graf secara umum: (a) Pohon dan (b) Graf

## II.2 Revolusi Teknologi Web terhadap Pembelajaran

>> jelaskan jg “worked example” untuk mendukung pembelajaran

Bonk (2009) menulis sebuah buku berjudul “*The World Is Open: How Web Technology Is Revolutionizing Education*” yang berisi tentang teknologi web dan internet yang mengubah cara belajar di dunia pada awal abad 21 ini. Internet telah membuka peluang besar bagi seluruh manusia di bumi. Melalui teknologi web setiap orang dapat belajar di mana dan kapan pun. Pembelajaran elektronik (*e-learning*) telah menjadi bagian penting dari setiap komunitas belajar.

Pada abad 21 ini, keuntungan bagi para pengajar untuk berbagi ilmu bukan merupakan hal yang sulit lagi. Secara nyata kehadiran internet telah mengubah cara berbagi kepada para pelajar di dunia, kelas, sekolah, kampus, dan masih banyak sisi yang berpotensi terkena dampak positifnya. Saat ini dengan internet atau *online sharing* dapat mempengaruhi setiap orang di mana pun dan kapan pun di dunia ini. Khususnya istilah *sharing* (berbagi) telah populer dalam lingkungan pembelajaran yang diberikan melalui teknologi daring secara kolaboratif.

Teknologi telah menjadi bagian penting yang digunakan untuk mempercepat proses pembelajaran. Melalui berbagai perspektif setiap orang yang berbeda

## II.3 Terminologi Visualisasi Perangkat Lunak

Visualisasi perangkat lunak (*Software Visualization*) merupakan 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 dan dinamis, program (*code*), dan data yang diproses. Visualisasi perangkat lunak memiliki tujuan utama untuk menganalisa 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.

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 dari konsep langkah kerja perangkat lunak, sedangkan VP terkait pada cara kerja eksekusi kode program dan proses struktur datanya. Melalui kakas VP dapat memberikan pemahaman lebih baik kepada pelajar mengenai implementasi dari materi ajar (Sorva, 2012).

Visualisasi secara intuitif tampaknya telah menjadi bagian dari sarana pembelajaran yang menarik (Sorva, 2012). Berbagai literatur mendukung pernyataan tersebut (Cetin dan Andrews-Larson, 2016; Guo, 2013; Pathania dan Singh, 2014; Sorva dkk., 2013; Sorva dan Sirkiä, 2010). Penelitian terkait banyak yang mengusulkan visualisasi digunakan dalam lingkungan belajar. Hampir setiap teori pembelajaran merekomendasikan visualisasi sebagai sarana untuk memberikan pemahaman lebih baik kepada pembacanya (Clark dan Mayer, 2011; Mayer, 2014).

Penggunaan sarana visual sebagai alat bantu belajar telah menjadi budaya di seluruh dunia, baik itu dalam pembelajaran komputer atau lainnya. Berdasarkan salah satu survei bahwa penggunaan visualisasi dalam perkuliahan informatika atau ilmu komputer hampir dilakukan setiap hari (Sorva, 2012).

Banyak kakas VP dibuat karena termotivasi dengan munculnya kesulitan belajar untuk memahami konsep algoritma dan pemrograman. Beberapa kesulitan yang ada diantaranya sebagai berikut (Sorva dkk., 2013).

1. **Persepsi statis terhadap pemrograman**. Umumnya para pelajar memahami konsep program tertentu (seperti objek, rekursi) hanya merupakan bagian dari kode program, bukan memahaminya sebagai suatu komponen aktif yang utuh saat eksekusi kode program.
2. **Kesulitan memahami cara kerja mesin komputer**. Hambatan utama bagi pemula adalah memahami notasi mesin (Boulay, 1986), yaitu sebuah abstraksi dari komputer dalam eksekusi kode program dengan bahasa pemrograman tertentu. Bahasa pemrograman dan paradigma yang berbeda dapat mengasosiasikan dengan notasi mesin yang berbeda pula.
3. **Salah paham terhadap konsep dasar pemrograman**. Berbagai literatur pendidikan komputasi terdapat banyak laporan tentang kesalahpahaman pelajar yang dilakukan saat mempelajari konsep pemrograman (Clancy dkk., 2003; Sorva, 2012). Permasalahannya menyangkut eksekusi kode program yang tersembunyi dalam notasi mesin dan konsep algoritma yang tidak jelas dalam kode program, seperti referensi variabel dan *pointer*, objek, rekursi, variabel kontrol perulangan, dan sebagainya.
4. **Kesulitan penelusuran dan langkah program**. Kesulitan yang dialami pelajar dengan menelusuri program langkah demi langkah telah banyak dilaporkan di berbagai studi multinasional dan literatur (Sorva dkk., 2013). Laporan tersebut menyatakan bahwa pelajar gagal memahami pernyataan program secara sekuensial dan belum mampu memahami proses pertukaran nilai variabel dalam tiga baris sederhana.

## II.4 Hasil Eksplorasi Kakas Visualisasi Program

>> jelaskan jg kakas VA yang sudah berkembang pesat, sehingga VP perlu pengembangan lebih lanjut karena masih jarang dan banyak manfaatnya.

>> penentuan bahasa pemrograman, kenapa pilih C dan C++?

>> penentuan dipilihnya framework yang akan dikembangkan, mengapa Matrix?

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

Pemilihan bahasa pemrograman ...

Pada Tabel II.3 menunjukkan perkembangan kakas VP khusus untuk bahasa pemrograman C dan C++ yang masih aktif pada terakhir dekade tahun ini. Sorva (2012) menjelaskan secara rinci 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++ untuk ditampilkan pada Tabel II.3. Beberapa kakas diperoleh dari literatur lain seperti kakas LEONARDO,

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

| **No.** | **Tahun** | **Nama** | **Dukungan Bahasa** | **Visual Graf** | **Berbasis Web** |
| --- | --- | --- | --- | --- | --- |
| 1 | 1996 | GRASP / jGRASP | Ada, Java, C, C++, Objective-C, VHDL | Binary Tree |  |
| 2 | 2000 | The Teaching Machine | C++, Java |  | **√** |
| 3 | 2000 | LEONARDO | C |  |  |
| 4 | 2002 | PlanAni | Pascal, Java, C, Python |  |  |
| 5 | 2003 | Jeliot 2000 / Jeliot 3 | Java, C, Python |  |  |
| 6 | 2004 | OGRE | C++ |  |  |
| 7 | 2005 | VIP | C++ |  |  |
| 8 | 2005 | ViLLE | Java, C++, Python, PHP, JavaScript, pseudocode |  |  |
| 9 | 2010 | Online Python Tutor | Python, Java, C, C++, Ruby, JavaScript | **-** | **√** |
| 10 | 2011 | CSmart | C |  |  |

>> jelaskan juga perkembangan selain matrix framework yang pernah dibuat oleh peneliti sebelumnya, jika ada

>> apa ada kakas VP bkn utk c/c++ tapi mendukung visual graf? Dan atau berbasis web

## II.5 Pentingnya Belajar Eksekusi Kode dalam Pemrograman

Banyak pelajar di dunia tidak mau untuk belajar eksekusi kode program (Sorva, 2012). Padahal dalam mempelajari pemrograman, harus mempelajari eksekusi kode program agar dapat memahami algoritma dan pemrograman secara utuh. Eksekusi kode program dapat memberikan pemahaman lebih baik kepada pelajar mengenai implementasi dari konsep algoritma (Sorva, 2012). Melalui eksekusi kode, pelajar dapat teruji tingkat pemahamannya terhadap konsep algoritma yang telah dipelajarinya. Seperti halnya mengerti dan paham tentang konsep cara berenang, namun tidak pernah praktik langsung ke dalam air untuk berenang.

Ketika sebuah solusi atau algoritma telah dibuat, maka sebuah kode program tetap harus ditulis. Saat ini komputer banyak berperan penting dalam kehidupan manusia sehingga masalah yang dihadapi menjadi lebih besar dan kompleks, yang membutuhkan pengembangan kode program lebih rumit lagi. Tujuan dari penelitian tesis ini adalah untuk menghasilkan kakas VP yang dapat memberikan kemampuan analisa pemrograman dan algoritma secara simultan kepada para pelajar terhadap graf, sehingga mereka dapat mengembangkan program dengan kapasitas keefisienan yang maksimum.

Dalam pembelajaran pemrograman sebagai disiplin ilmu yang memerlukan ketelitian tinggi ada dua hal penting yang saling melengkapi (Sorva, 2012), yaitu:

* Pemrograman merupakan pendekatan secara komputasi yang eksekusinya didefinisikan dalam sebuah mesin komputer. Dalam dunia nyata, intuisi pelajar sebenarnya dapat memahami eksekusi program khususnya pada sistem interaktif, seperti potongan-potongan kode program yang dibuat oleh pelajar dapat langsung terlihat hasil eksekusinya. Melalui teknik tersebut pelajar tidak lagi bingung apa yang akan terjadi pada program yang dibuatnya sehingga dapat mempercepat proses pemahamannya.
* The computation-based approach presents programming as a way to define

executions on machines. It grounds the student’s intuition in the real world

by means of actual executions on real systems. This is especially effective

with an interactive system: the student can create program fragments and

immediately see what they do. Reducing the time between thinking “what

if” and seeing the result is an enormous aid to understanding. Precision

is not sacrificed, since the formal semantics of a program can be given in

terms of an abstract machine.

* The logic-based approach presents programming as a branch of mathematical logic. Logic does not speak of execution but of program properties, which is a higher level of abstraction. Programs are mathematical constructions that obey logical laws. The formal semantics of a program is given in terms of a mathematical logic. Reasoning is done with logical assertions. The logic-based approach is harder for students to grasp yet it is essential for defining precise specifications of what programs do.

Like Structure and Interpretation of Computer Programs, by Abelson, Sussman,

& Sussman [1, 2], our book mostly uses the computation-based approach. Concepts are illustrated with program fragments that can be run interactively on an accompanying software package, the Mozart Programming System [129]. Programs are constructed with a building-block approach, bringing together basic

concepts to build more complex ones. A small amount of logical reasoning is introduced in later chapters, e.g., for defining specifications and for using invariants to reason about programs with state.

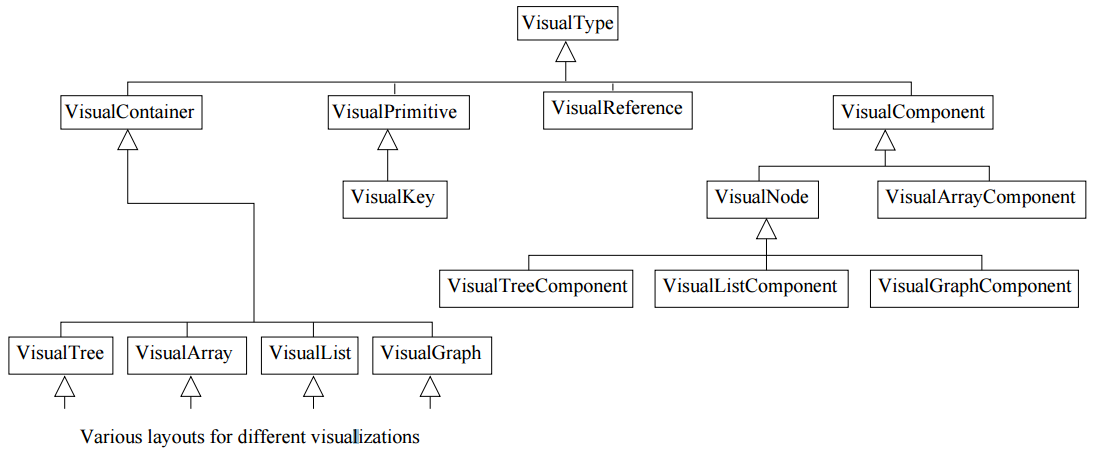
## II.6 Matrix Framework

Pada subbab ini akan dibahas konsep *Matrix framework* yang dikembangkan oleh Korhonen dkk. (2004). *Framework* ini dibangun untuk mendukung desain dan implementasi visualisasi pada tingkat abstrak. Awal pengembangannya diimplementasi menggunakan bahasa pemrograman Java. Kemudian dipadukan dengan *XML library* yang berguna untuk membaca konfigurasi berkas *Matrix*. Berikut ini akan dijelaskan enam bagian yang ada di dalam *Matrix framework*, yaitu visualisasi, struktur, konstruksi visual, animasi, simulasi, dan tampilan antarmuka pengguna.

### II.6.1 Visualisasi

Visualisasi struktur data dalam *Matrix framework* berisi empat konsep dasar visual, yaitu *visual container*, *visual component*, *visual reference* dan *visual data*. *Visual container* berupa struktur yang kompleks, menyimpan nilai variabel (node, indeks, dan lainnya) yang saling terhubung dengan cara tertentu. Kemudian tiap variabel di dalam *visual container* terdapat *visual component* yang berguna untuk memvisualisasikan variabel. *Visual reference* sebagai penghubung antar *visual component* dan *visual data* yang berisi tipe data primitif atau struktur data yang lebih kompleks dengan atribut-atributnya. Atribut kunci yang kompleks direpresentasikan menggunakan *visual container* yang bersarang di dalam *visual component*.

Pada Gambar II.3 menunjukkan hirarki *class Matrix framework* yang digunakan untuk implementasi visualisasi. *Class* pada tingkat kedua dari gambar hirarki tersebut merupakan empat konsep visual yang telah dijelaskan sebelumnya. VisualContainer berhubungan dengan *visual container*, VisualComponent dengan *visual component*, VisualReference dengan *visual reference*, dan VisualPrimitive dengan *visual data*.



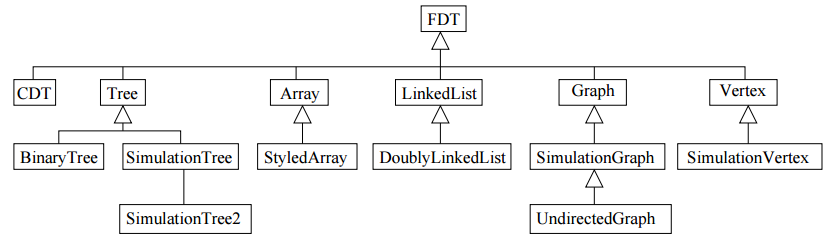
1. Hirarki *class* visualisasi *Matrix framework* (Korhonen dkk., 2004)

Visualisasi dengan tipe data berbeda membutuhkan setiap fungsi yang berbeda pula, seperti tipe data *arrays*, graf, *list*, atau pohon. Pada kelas VisualContainer terdiri dari empat sub kelas (lihat Gambar II.3), yaitu VisualTree, VisualArray, VisualList, dan VisualGraph. Setiap sub kelas berkorespondensi dengan salah satu tipe data yang berada di atasnya. Setiap kelas menangani fungsinya masing-masing terhadap visualisasi dan manipulasi tipe data tertentu. Sebagai contoh, kelas VisualGraph berperan ketika pengguna menambahkan sisi (*edge*), sedangkan kelas VisualTree memastikan bahwa graf yang ditampilkan adalah pohon, yang berarti tidak terdapat sisi atau garis yang melintang (*cross*), atau sisi yang mengarah kembali (*back-edges*) ke noktah sebelumnya. Selain itu, pada masing-masing kelas utama terdapat satu atau lebih sub kelas yang menangani tipe data tertentu dalam menguraikan visualisasi dan menggambarkan *layout* tertentu.

Setiap visualisasi tipe data memiliki satu atau lebih *layout* tertentu.

### II.6.2 Struktur

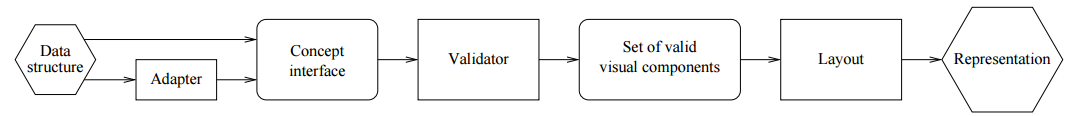
Matrix framework menggunakan sejumlah *concept interfaces* untuk membedakan antara *fundamental data types* (FDT) dan *abstract data types* (ADT). *Concept interfaces* diimplementasi menggunakan interface Java. Hirarki dari struktur *interfaces* yang digunakan dapat dilihat pada Gambar II.4. *Interfaces FDT* merupakan *superclass* (*root*) dari semua konsep *interfaces*. Sebuah objek yang dikenali oleh *Matrix framework* akan dapat divisualisasikan dari implementasi *interfaces FDT* ini. Untuk objek yang tidak dapat dikenali (bukan objek elemen *FDT*), maka akan disimpan dan divisualisasikan sebagai objek data primitif (hanya merepresentasikan sebuah *string* dari elemen tersebut).



1. Turunan hirarki *concept interfaces Matrix* (Korhonen dkk., 2004)

### II.6.3 Konstruksi Visual

Proses representasi visual untuk struktur data diilustrasikan seperti pada Gambar II.5. Proses visualisasi dibagi menjadi tiga tahap, yaitu *Adaptation*, *Validation*, dan *Laying out*. Pada proses *Adaptation* (bagian *Adapter*) terjadi pencocokan visualisasi struktur data terhadap konsep visual *interfaces*. Namun keadaan ini akan membuat sulit bagi *programmer* untuk mengimplementasi *interface* mana yang efisien sesuai kebutuhan visualnya. Oleh karena itu, sistem menyediakan *Adapter* untuk menangani visualisasi terhadap beberapa konsep struktur data seperti yang telah dijelaskan pada Gambar II.3 sebelumnya.



1. Proses representasi visual (Korhonen dkk., 2004)

Sebagai contoh, *Binary heap* diimplementasi dari sebuah *array* dan dapat divisualisasikan dalam bentuk representasi *array* atau *binary tree*.

### II.6.4 Animasi

Animasi dalam *Matrix framework* dibagi menjadi dua bagian besar: *forward animation*, berfungsi untuk merepresentasikan secara visual struktur data yang akan terjadi perubahan pada langkah berikutnya, dan *backward animation*, berfungsi untuk menampilkan kembali visualisasi struktur data yang telah terjadi perubahan pada langkah sebelumnya.

Sistem dapat secara otomatis mendukung *forward animation* terhadap semua jenis bentuk struktur data terhadap *concept interfaces* yang telah dijelaskan pada Sub bab II.6.2. Untuk mendukung *backward animation* dibutuhkan fitur lain yang cukup sulit dalam memrogramnya. Sebuah animasi dalam *Matrix framework* disimpan dalm bentuk rangkaian. Sehingga walaupun pengguna menggunakan fitur *backward animation*, sistem akan mengembalikan bentuk struktur data yang relevan sesuai konfigurasi awal. Namun, hal tersebut sangat sulit menyimpan perubahannya yang dibuat secara acak sebagai data objek Java. Oleh sebab itu, variabel yang berisi struktur data harus disimpan ke objek khusus yang disebut MemoryStructure. Objek-objek tersebut juga merepresen-tasikan beberapa data primitif Java, objek *array*, dan mendukung penyimpanan pada tiap perubahan yang terjadi pada struktur data. Hal itu akan sulit dilakukan pada *framework* jika tanpa objek khusus tersebut untuk menggunakan fitur *backward animation*.

### II.6.5 Simulasi

Setiap operasi simulasi yang dilakukan oleh pengguna untuk menginterpretasikan sebuah visualisasi dilakukan melalui VisualType (lihat Gambar II.2). Dari VisualType ditentukan bentuk mana yang cocok dengan struktur data yang akan direpresentasikan. Kemudian akan memanggil salah satu sub metode yang berada dibawahnya. Setelah sub metode tersebut cocok dan sukses untuk representasi struktur data, VisualType akan ditandai sebagai *class* yang *invalid*. Sistem akan mengabaikan VisualType dan dilanjutkan proses ke sub metode terkait untuk dilakukan *lay out* dan merender ulang visual struktur datanya.

### II.6.6 Tampilan Antarmuka Pengguna

*Graphical user interface (GUI)* atau tampilan antarmuka pengguna merupakan bagian utama aplikasi untuk interaksi antara sistem dan pengguna menggunakan *Matrix framework*. GUI berisi beberapa bagian Panel yang berkaitan dengan fitur visualisasi struktur data atau komponen lain.

## II.7 Kesimpulan Awal Berdasarkan Studi Literatur

Studi literatur yang telah dilakukan memberikan pemahaman mendasar mengenai konsep graf dan macam algoritmanya yang telah berkembang, revolusi proses pembelajaran melalui teknologi web, terminologi visualisasi perangkat lunak, kakas VP yang telah berkembang hingga saat ini, serta peran penting eksekusi kode program dalam pembelajaran pemrograman.

# Bab III Analisis dan Perancangan

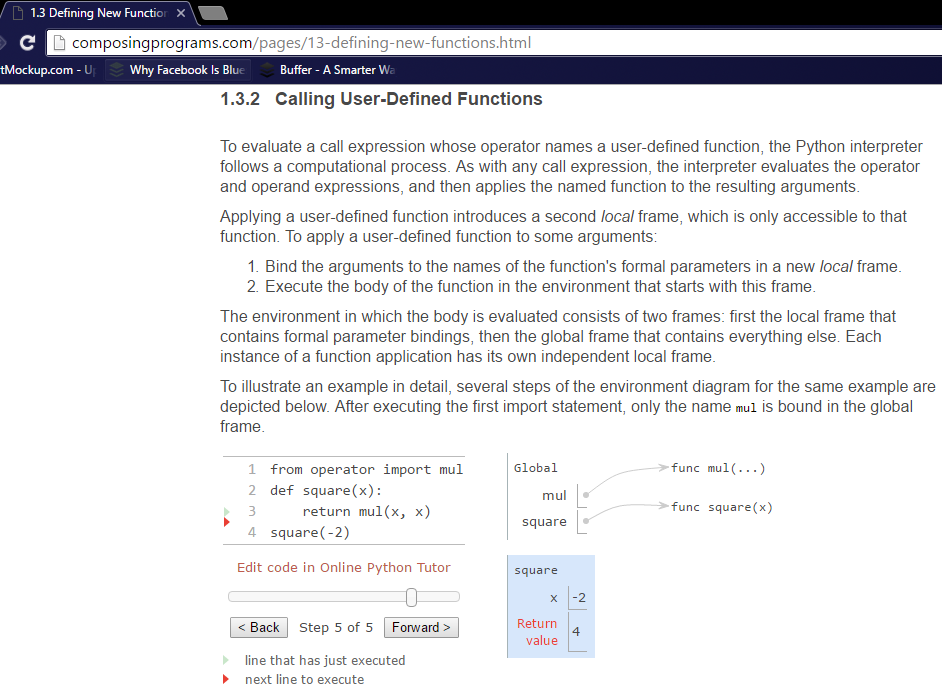
Bab ini menguraikan rumusan metode atau langkah-langkah yang diperlukan untuk menerjemahkan kode program menjadi visual graf. Selanjutnya dilakukan penentuan ruang lingkup terhadap proses visualisasi graf.

Penelitian tesis ini dilakukan berdasarkan penelitian sebelumnya yang telah berhasil membuat suatu kakas visualisasi kode program berbasis web tanpa harus memasang aplikasi khusus untuk mengeksekusinya, kakas visualisasi kode program ini disebut *Online Python Tutor* (Guo, 2013). Proyek dari kakas ini bersifat gratis dan bersumber terbuka (*open-source*), sehingga siapa pun dapat mengembangkannya tanpa harus terikat dengan lisensi berbayar. Terdapat tiga karakteristik utama yang membuat kakas *Online Python Tutor* ini modern, unik dan efektif (Guo, 2013), yaitu :

* **Python**: dalam beberapa akhir tahun ini, *Python* telah menjadi bahasa pemrograman yang dianjurkan dalam *computer science* dari banyak universitas terkenal di dunia. Contohnya, dua dari departemen *computer science* terbesar yaitu Institut Teknologi Massachusetts (MIT) dan Universitas California (UC) Berkeley, keduanya telah beralih struktur kurikulum ke Python. Universitas Negeri Michigan berganti dari *C++* ke *Python* dan pergantian ini telah dibuktikan secara empiris bahwa murid-murid pemula dapat belajar serta mempersiapkan dirinya lebih baik untuk mengikuti perkuliahan komputer lanjutan dibandingkan dengan bahasa pemrograman lainnya (Guo, 2013). Python juga digunakan dalam pendidikan daring (*free online course*) yang ditawarkan oleh *Udacity*, *Coursera*, dan *edX*, yang pelajarnya mencapai puluhan ribu.

Walaupun *Python* telah banyak yang menggunakan, hampir tidak ada satu pun kakas visualisasi program untuk *Python* (Guo, 2013). Selain *Online Python Tutor*, hanya ada *UUhistle* (Sorva dan Sirkiä, 2010) dan *Jype* (Helminen dan Malmi, 2010) yang merupakan kakas visualisasi untuk kode program *Python*. (Namun, proyek *Jype* sudah tidak aktif kembali (Sorva, 2012), dan tidak ditemukan sumber kodenya secara daring (Guo, 2013)).

* **Berbasis** **web**: *Online Python Tutor* merupakan satu-satunya kakas visualisasi program yang dapat digunakan secara daring melalui aplikasi selancar web tanpa harus memasang aplikasi khusus lainnya. *UUhistle* dan *Jype* dibuat dalam bahasa pemrograman *Java*, sehingga untuk menggunakannya melalui web harus sudah terpasang versi Java yang sesuai, hal tersebut membutuhkan waktu lagi untuk memasang aplikasi pendukungnya. Secara jelas nampak bahwa kakas visualisasi *Online Python Tutor* lebih sederhana dan cukup cepat prosesnya dengan hanya mengunjungi *URL* dari aplikasi selancar web.
* **Bersifat dapat dilekatkan di mana saja (*embeddable*)**: kakas ini dapat menampilkan visualisasi pada halaman *web* lain dengan memasukkan satu baris kode *JavaScript* setiap satu visualisasi. Saat Philip Guo (2013) melakukan risetnya, ada tiga buku digital interaktif *computer science* yang telah menggunakan *Online Python Tutor* ke dalam pelajarannya, yaitu *CS61A: Structure and Interpretation of Computer Programs*, *Beyond PDF and ePub: toward an interactive textbook*, dan *CS Circles: An In-Browser Python Course for Beginners*. Para pelajar dapat membaca pelajaran dan berinteraksi dengan visualisasi kode program pada halaman *web* yang sama, seperti yang ditunjukkan pada gambar III.1 berikut ini.



|  |  |
| --- | --- |
| Gambar III.1 | Contoh visualisasi *Online Python Tutor* (Guo, 2013) dalam buku digital interaktif (http://cs61a.org). |

Analisis dan Perancangan Struktur File XML Konfigurasi

# Bab IV Pengembangan Kakas

# DAFTAR PUSTAKA

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

Boulay, B.D. (1986): Some Difficulties of Learning to Program, *J. Educ. Comput. Res.*, **2**, 57–73.

Cetin, I. (2016): Preservice Teachers’ Introduction to Computing: Exploring Utilization of Scratch, *J. Educ. Comput. Res.*, 735633116642774.

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

Clancy, M., Titterton, N., Ryan, C., Slotta, J. and Linn, M. (2003): New roles for students, instructors, and computers in a lab-based introductory programming course, *ACM SIGCSE Bulletin*, ACM, 132–136, diperoleh melalui situs internet: http://dl.acm.org/citation.cfm?id=611951 (accessed 21 July 2016).

Clark, R.C. and Mayer, R.E. (2011): *E-learning and the science of instruction : proven guidelines for consumers and designers of multimedia learning - 3rd ed.*, San Francisco, CA, Pfeiffer.

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.

Harary, F. (1969): *Graph theory*, Canada, USA, Addison-Wesley Publishing Company, Inc.

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.

Inggriani Liem (2008): *Diktat Struktur Data*, Edisi 2008, Bandung, Teknik Informatika, Institut Teknologi Bandung.

Korhonen, A., Malmi, L., Silvasti, P., Karavirta, V., Lönnberg, J., Nikander, J., Stålnacke, K. and 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.

Margulieux, L.E., Catrambone, R. and Guzdial, M. (2016): Employing subgoals in computer programming education, *Comput. Sci. Educ.*, **26**, 44–67.

Mayer, R.E. (2014): *The Cambridge handbook of multimedia learning - Second Edition*, New York, USA, Cambridge University Press.

Pathania, U. dan Singh, A. (2014): Visualization Tool for Tree and Graph Algorithms with Audio Comments, Int. J. Softw. Web Sci. IJSWS, 14, 51–58.

Piteira, M. and 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 (accessed 8 September 2016).

Rinaldi Munir (2010): *Matematika Diskrit Edisi 3*, Bandung, Informatika Bandung.

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

Sorva, J. dan Sirkiä, T. (2010): UUhistle: a software tool for visual program simulation, Proceedings of the 10th Koli Calling International Conference on Computing Education Research, ACM, 49–54.

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.

Valiente, G. (2002): *Algorithms on Trees and Graphs*, Berlin, Heidelberg, Springer Berlin Heidelberg, diperoleh melalui situs internet: http://link.springer.com/10.1007/978-3-662-04921-1 (accessed 15 September 2016).

Ware, C. (2004): *Information visualization: perception for design, 2nd edition*, 2nd edition, San Francisco, Kanada, Elsevier Inc.