

BAB 1

PENDAHULUAN

1.1 Latar Belakang

Kurikulum didefinisikan sebagai seperangkat rencana dan pengaturan mengenai capaian pembelajaran, an lulusan, bahan kajian, proses, dan penilaian yang digunakan sebagai pedoman penyelenggaraan program studi menjadi sarana utama untuk mencapai tujuan tersebut. Penyusunan kurikulum 2018 berpegang pada prinsip bahwa kurikulum yang baik adalah kurikulum yang tidak hanya kokoh, secara teoretis konseptual dapat dipertanggungjawabkan, namun juga secara praktis dapat dilaksanakan. Selain itu kurikulum juga harus cukup fleksibel agar dapat mengakomodasi perubahan-perubahan, namun tanpa kehilangan ciri atau kekhasan dari program studi. Dalam penyusunan kurikulum 2018 program studi Informatika secara khusus juga memperhatikan Kerangka Kualifikasi Nasional Indonesia (KKNI) yang tertuang dalam Peraturan Presiden no 8 tahun 2012. KKNI merupakan pernyataan kualitas SDM Indonesia, di mana tolok ukur kualifikasinya ditetapkan berdasarkan capaian pembelajaran (*learning outcomes*) yang dimilikinya. Tahapan penyusunan kurikulum 2018 meliputi kegiatan sebagai berikut:

1. Melakukan evaluasi diri dan pelacakan lulusan.
2. Merumuskan profil lulusan.
3. Menentukan capaian pembelajaran.
4. Menentukan bahan kajian.
5. Menyusun matriks pembelajaran dan bahan kajian.
6. Membentuk mata kuliah.
7. Menyusun struktur kurikulum dan menentukan metode pembelajaran.

Teknologi baru sekarang memungkinkan untuk membangun layanan yang menjawab pertanyaan-pertanyaan secara otomatis. Sebagian besar data yang diperlukan untuk menjawab pertanyaan-pertanyaan dihasilkan oleh badan-badan publik. Namun, seringkali data yang diperlukan belum tersedia dalam bentuk yang mudah digunakan. Data terbuka berbicara tentang bagaimana membuka potensi dari informasi resmi dan lainnya untuk mengaktifkan layanan-layanan baru. Gagasan dari data terbuka itu sendiri bertujuan agar setiap orang bebas untuk mengakses dan menggunakan ulang untuk berbagai tujuan - sudah bergulir dalam beberapa tahun ini. Data terbuka itu sendiri memiliki arti yaitu data yang dapat secara bebas digunakan, digunakan ulang dan didistribusi ulang oleh siapapun - hanya patuh, umumnya, pada keharusan untuk menyebutkan siapa penciptanya dan berbagi dengan lisensi yang sama. Defini Terbuka memberikan rincian yang tepat apa yang dimaksud data terbuka. Ringkasannya adalah:

1. **Ketersediaan dan Akses:** data harus tersedia secara keseluruhan dan tidak lebih dari pada biaya reproduksi yang masuk akal, akan lebih baik bila bisa dilakukan dengan pengunduhan melalui internet.

2. **Penggunaan-ulang dan Distribusi ulang:** data harus disediakan di bawah ketentuan yang mengizinkan untuk penggunaan-upang dan pendistribusian ulang termasuk memadukan dengan kumpulan data lainnya.

3. **Partisipasi Universal:** setiap orang harus diperbolehkan untuk menggunakan, menggunakan-ulang dan mendistribusi ulang - tidak boleh ada diskriminasi terhadap bidang kerja atau perseorangan atau kelompok.

Untuk menampung data terbuka dapat digunakan *github* sebagai salah satu penampung untuk menyimpan data. *Github* sebagai *open source* di dalamnya dapat menyimpan data dalam *format JSON*. *JSON* digunakan sebagai acuan dalam pembuatan pohon kurikulum 2018. *Format JSON* bakal diubah ke dalam *DOT Language* untuk menghasilkan graf. Penggunaan graf ditujukan agar mempermudah dalam melihat kurikulum baru. Untuk mem *visualisasi* kan graf digunakan *viz.js*, *Viz.js* ini nantinya akan membantu dalam menghasilkan graf yang akan di tampilkan.

1.2 Rumusan Masalah

Berdasarkan latar belakang masalah yang telah dijelaskan, rumusan masalah pada penelitian ini adalah:

1. Bagaimana menerjemahkan perangkat lunak dalam bentuk *word* ke bentuk *JSON*.
2. Bagaimana membuat perangkat lunak dari bentuk *JSON* ke dalam graf.

1.3 Tujuan

Berdasarkan rumusan masalah di atas, maka tujuan dari penelitian ini adalah:

1. Membuat terjemahan dari bentuk *word* ke dalam bentuk *JSON*.
2. Membuat perangkat lunak dalam bentuk graf.

1.4 Batasan Masalah

Adapun batasan masalah yang didapat dari tujuan dan rumusan masalah di atas adalah:

1. Perangkat lunak menghasilkan pohon kurikulum.

1.5 Metodologi Penelitian

Dalam penyusunan skripsi ini mengikuti langkah-langkah metodologi penelitian sebagai berikut:

1. Melakukan studi pustaka untuk dijadikan referensi dalam pembangunan perangkat lunak pohon kurikulum.
2. Melakukan studi tentang penggunaan *vis.js* untuk visualisasi pohon kurikulum.
3. Melakukan studi tentang data terbuka.
4. Melakukan studi tentang cara penggunaan *DOT Language*

1.6 Sistematika Penulisan

Keseluruhan bab yang disusun dalam penelitian ini terbagi kedalam bab-bab sebagai berikut:

1. Bab 1 Pendahuluan Bab ini membahas mengenai latar belakang, rumusan masalah, tujuan, batasan masalah, metodologi penelitian dan sistematika penulisan.
2. Bab 2 Dasar Teori Bab ini membahas mengenai pengertian graf, data terbuka, JSON, apa itu DOT *Language*, dan visualisasi menggunakan viz.js.
3. Bab 3 Analisis Bab ini akan membahas mengenai JSON yang dapat dipakai sebagai sumber data terbuka.
4. Bab 4 Perancangan Bab ini akan membahas mengenai perancangan struktur pohon kurikulum untuk mahasiswa, di mana nanti di dalamnya akan berisi mata kuliah, syarat tempuh, dan syarat lulus.
5. Bab 5 Implementasi dan Pengujian Bab ini akan membahas mengenai pengujian, implementasi kode program untuk membuat pohon kurikulum.
6. Bab 6 Kesimpulan dan Saran Bab ini akan membahas mengenai kesimpulan dari penelitian yang telah dilakukan dan saran-saran untuk pengembangan lebih lanjut dari penelitian ini.

BAB 2

DASAR TEORI

Pada bab ini akan diuraikan teori-teori yang berhubungan dengan pembangunan pohon kurikulum. Teori-teori tersebut adalah teori tentang pengertian graf, data terbuka, *JSON*, *DOT language*, dan visualisasi pohon menggunakan *vis.js*.

2.1 Graf

2.1.1 Definisi Graf

Suatu graph didefinisikan oleh himpunan verteks dan himpunan sisi (edge). Verteks menyatakan entitas-entitas data dan sisi menyatakan keterhubungan antara verteks. Biasanya untuk suatu graf G digunakan notasi matematis.

$$G=(V,E)$$

$$G = \text{Graph}$$

$$V = \text{Simpul atau vertex, atau node, atau titik}$$

$$E = \text{Sisi atau garis, atau Edge}$$

V adalah himpunan *verteks* dan E himpunan sisi yang terdefinisi antara pasangan-pasangan verteks. Sebuah sisi antara verteks x dan y ditulis x, y . Suatu graph $H = (V_1, E_1)$ disebut subgraph dari graph G jika V_1 adalah himpunan bagian dari V dan E_1 himpunan bagian dari E .

2.1.2 Istilah dalam Graf

1. *Incident* Jika e merupakan busur dengan simpul-simpulnya adalah v dan w yang ditulis $e=(v,w)$, maka v dan w disebut "terletak" pada e , dan e disebut incident dengan v dan w .

2. *Degree* Di dalam Graph ada yang disebut dengan *Degree*, *Degree* mempunyai 3 jenis antara lain :

- Degree dari suatu verteks x dalam undigraph adalah jumlah busur yang incident dengan simpul tersebut.
- Indegree dari suatu verteks x dalam digraph adalah jumlah busur yang kepalanya incident dengan simpul tersebut, atau jumlah busur yang "masuk" atau menuju simpul tersebut.
- Outdegree dari suatu verteks x dalam digraph adalah jumlah busur yang ekornya incident dengan simpul tersebut, atau jumlah busur yang "keluar" atau berasal dari simpul tersebut.

3. *Adjacent* Pada graph tidak berarah, 2 buah simpul disebut adjacent bila ada busur yang menghubungkan kedua simpul tersebut. Simpul v dan w disebut adjacent. Pada graph berarah, simpul v disebut adjacent dengan simpul w bila ada busur dari w ke v .

4. *Successor dan Predecessor* Pada graph berarah, bila simpul v adjacent dengan simpul w , maka simpul v adalah *successor* simpul w , dan simpul w adalah *predecessor* dari simpul v .

2.2 Data Terbuka

Teknologi sekarang memungkinkan untuk membangun layanan yang menjawab pertanyaan-pertanyaan secara otomatis. Sebagian besar data yang diperlukan untuk menjawab pertanyaan-pertanyaan dihasilkan oleh badan-badan publik. Namun, seringkali data yang diperlukan belum tersedia dalam bentuk yang mudah digunakan. Gagasan dari data terbuka mengarah kepada informasi di mana setiap orang bebas untuk mengakses dan menggunakan ulang untuk berbagai tujuan - sudah bergulir dalam beberapa tahun ini.

2.2.1 Apa itu Data Terbuka

Data terbuka adalah data yang dapat digunakan secara bebas, dimanfaatkan, dan didistribusikan kembali oleh siapapun tanpa syarat, kecuali dengan mengutip sumber dan pemilik data. Selain itu, seluruh data yang dipublikasikan harus mengikuti peraturan perundang-undangan yang berlaku. Kriteria penting dari data terbuka adalah:

1. Ketersediaan dan Akses Data harus tersedia utuh dan bebas biaya. Akan lebih baik jika data dapat diunduh melalui internet. Data juga harus tersedia dalam bentuk yang mudah digunakan dan dapat diolah kembali.
2. Penggunaan dan Pendistribusian Data yang digunakan dan didistribusikan kembali harus memenuhi syarat-syarat yang telah ditentukan.
3. Partisipasi Universal Setiap orang bebas menggunakan dan mendistribusikan kembali *dataset*. Tidak diperkenankan adanya diskriminasi atas bidang usaha, orang, atau kelompok.

Semua kriteria yang ada di dalam data terbuka sangat penting karena menunjukkan kejelasan tentang apa yang dimaksud dengan terbuka itu sendiri. Istilah yang digunakan untuk menjelaskan ketiga kriteria data terbuka adalah *interoperabilitas*. Interoperabilitas sangat penting karena memungkinkan komponen-komponen yang berbeda untuk bisa bekerja sama. Kemampuan untuk mengkomponenisasi komponen-komponen sangatlah esensial untuk membangun sistem yang besar dan kompleks. Tanpa *interoperabilitas* hal ini menjadi tidak mungkin di mana kemampuan untuk berkomunikasi (lintas operasi) sangat berpengaruh terhadap keberhasilan suatu rencana.

Inti dari sebuah "keumuman" *data* merupakan salah satu bagian dari materi "terbuka". *Interoperabilitas* ini merupakan komponen penting untuk merealisasikan praktik utama manfaat dari "keterbukaan": Peningkatan dramatis kemampuan untuk mengkombinasikan sekumpulan data berbeda secara bersama-sama sehingga merangsang pengembangan produk dan layanan yang lebih baik. keterbukaan dapat memastikan bahwa ketika ada dua kumpulan data dari dua sumber berbeda, maka kita dapat menggabungkan data tersebut secara bersama-sama, dan memastikan bahwa data yang kita dapat informasinya benar.

2.2.2 Mengapa Data Terbuka

Data terbuka adalah sumber daya luar biasa yang belum dimanfaatkan sepenuhnya. Banyak individu dan organisasi mengumpulkan berbagai jenis data berbeda dalam rangka untuk melakukan tugas mereka. Pemerintah sangat signifikan dalam hal ini, tidak hanya karena kuantitas dan sentralitas dari data yang dikumpulkan, tetapi juga karena sebagian besar dari data pemerintah adalah bersifat publik secara hukum, dan oleh karena itu bisa dibuat terbuka dan tersedia untuk orang lain untuk dipergunakan. hal itu menjadi menarik karena banyak individu atau kelompok yang ingin mengetahui data yang ada.

Ada banyak area di mana kita bisa mengharapkan data terbuka untuk menjadi sebuah nilai, dan menjadi contoh bagaimana data terbuka telah digunakan. Ada juga kelompok dengan banyak orang berbeda dan organisasi yang dapat meraih keuntungan dari ketersediaan data yang terbuka, termasuk pemerintah itu sendiri. Pada saat yang sama adalah mustahil untuk memprediksi secara

tepat bagaimana dan di mana nilai akan dibuat di masa depan. Sifat alami dari inovasi adalah bahwa pengembangan seringkali datang dari tempat yang tidak mungkin. Hal ini sudah dimungkinkan dengan merujuk pada sejumlah data terbuka yang telah menciptakan nilai. Beberapa nilai ini meliputi:

- Transparansi dan kendali
- Partisipasi
- Penguatan mandiri
- Inovasi
- Efisiensi dan Efektivitas lebih baik dari layanan yang sudah ada
- Pengukuran pengaruh dari kebijakan-kebijakan
- Pengetahuan baru dari kombinasi sumber data dan pola-pola dalam volume data yang besar

2.2.3 Cara Membuka Data

Data Terbuka dapat dibuka para pemegang data. Para pemegang data dapat melakukannya secara mendasar, tetapi juga mencakup masalah-masalah yang tersembunyi dan menjebak. Terdapat tiga aturan kunci yang kami rekomendasikan saat membuka data:

1. Jadikan lebih praktis.
2. Terlibat dari awal dan melibatkan diri sesering mungkin.
3. Mengatasi kekhawatiran umum dan kesalahpahaman.

Ada empat langkah utama dalam membuat data terbuka, yang masing-masing akan dibahas secara rinci di bawah ini. Langkah-langkah tersebut adalah yang paling memungkinkan - banyak dari langkah-langkah tersebut dapat dilakukan secara bersamaan.

1. Pilih Kumpulan Data(s)

Pemilihan kumpulan-kumpulan data(s) yang direncanakan untuk menjadikannya terbuka merupakan langkah pertama – meskipun perlu diingat bahwa seluruh proses pembukaan data akan berulang dan dapat kembali ke langkah ini bila mengalami masalah di kemudian hari. Jika sudah mengetahui persis kumpulan-kumpulan data(s) maka dapat merencanakan untuk dibuka dan dapat langsung ke bagian berikutnya. Bagaimanapun juga, dalam banyak kasus, terutama untuk lembaga-lembaga yang besar, untuk berfokus pada memilih kumpulan data menjadi sebuah tantangan.

2. Menerapkan sebuah Lisensi Terbuka (Keterbukaan Resmi)

Di kebanyakan yurisdiksi terdapat hak kekayaan intelektual di dalam data yang mencegah pihak ketiga dari penggunaannya, penggunaan ulang dan pendistribusian data tanpa izin eksplisit. Bahkan di tempat di mana keberadaan hak hukum serba tidak pasti, penting untuk menerapkan lisensi demi sebuah kejelasan. Dengan demikian, jika kita berencana untuk membuat data tersedia, maka harus menaruh lisensi di atasnya dan jika data menjadi terbuka ini bahkan lebih penting lagi.

3. Menjadikan Data Tersedia

Data terbuka membutuhkan keterbukaan secara teknis sebagaimana keterbukaan yang resmi secara hukum. Khususnya, data harus bisa tersedia secara masal dalam format (yang dapat dibaca mesin).

- **Available**

Data seharusnya dihargai tidak lebih dari biaya reproduksi yang wajar, sebaiknya dijadikan sebagai unduhan gratis dari internet. Model penghargaan ini dapat dicapai karena lembaga anda tidak perlu menangani biaya apapun saat menyediakan data untuk digunakan.

- **In Bulk**

Data harus tersedia dalam kumpulan yang lengkap. Jika anda memiliki daftar yang dikoleksi di bawah aturan undang-undang, seluruh daftar tersebut harus tersedia untuk diunduh.

- **In an open, machine-readable format**

Penggunaan-ulang data yang disediakan oleh sektor publik tidak seharusnya tunduk pada pembatasan paten.

- **Jadikan hingga Mudah untuk ditemukan**

terbitkan di web dan mungkin kelola sebuah pusat katalog untuk membuat daftar dari kumpulan data terbuka.

2.3 JSON

JSON (*JavaScript Object Notation*) adalah format pertukaran data. JSON merupakan format teks yang tidak bergantung pada bahasa pemrograman apapun. Kenapa *JSON*? Karena ukurannya lebih kecil dibanding dengan XML, sifatnya "*self-describing*" dan mudah dimengerti. Formatnya berbasis teks dan terbaca manusia serta digunakan untuk mempresentasikan struktur data sederhana. Format teks dari JSON itu sendiri identik dengan kode untuk membuat objek *JavaScript* memiliki kesamaan dengan Java Script, hanya saja JSON lebih mudah dimengerti.

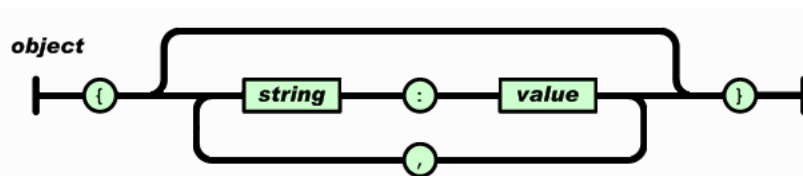
2.3.1 Struktur JSON

JSON terbuat dari dua struktur:

1. Kumpulan pasangan nama/nilai. Pada beberapa bahasa, hal ini dinyatakan sebagai objek (*object*), rekaman (*record*), struktur (*struct*), kamus (*dictionary*), tabel hash (*hash table*), daftar berkunci (*keyed list*), atau *associative array*.
2. Daftar nilai terurutkan (*an ordered list of values*). Pada kebanyakan bahasa, hal ini dinyatakan sebagai larik (*array*), vektor (*vector*), daftar (*list*), atau urutan (*sequence*).

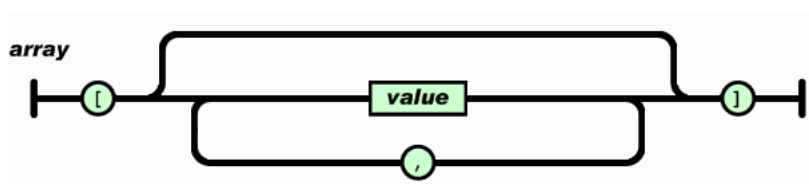
JSON menggunakan bentuk sebagai berikut:

1. **Objek** adalah sepasang nama/nilai yang tidak terurutkan. Objek dimulai dengan { (kurung kurawal buka) dan diakhiri dengan } (kurung kurawal tutup). Setiap nama diikuti dengan : (titik dua) dan setiap pasangan nama/nilai dipisahkan oleh , (koma).



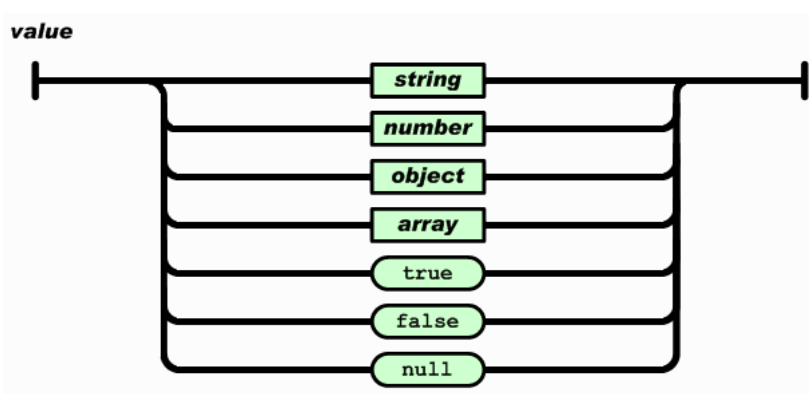
Gambar 2.1: Objek

2. **Larik** adalah kumpulan nilai yang terurutkan. Larik dimulai dengan [(kurung kotak buka) dan diakhiri dengan] (kurung kotak tutup). Setiap nilai dipisahkan oleh , (koma).



Gambar 2.2: Larik

- 1 3. **Nilai**, dapat berupa sebuah *string* dalam tanda kutip ganda, atau angka, atau true atau false
 2 atau null, atau sebuah objek atau sebuah larik. Struktur-struktur tersebut dapat disusun
 3 bertingkat.



Gambar 2.3: Nilai

4 2.3.2 Contoh Sintaks

5 Contoh berikut menunjukkan representasi JSON untuk suatu objek yang mendeskripsikan seseorang.

```

6 {
7   "namaDepan": "Budi",
8   "namaBelakang": "Subudi",
9   "alamat": {
10      "namaJalan": "Jl. Sudirman 15A",
11      "kota": "Jakarta Selatan",
12      "provinsi": "DKI Jakarta",
13      "kodePos": 11111 },
14   "nomerTelepon": [
15      "021 555-1234",
16      "021 555-4567"
17   ]
18 }
```

19 2.4 DOT Language

20 *DOT* adalah bahasa yang dapat digunakan untuk menampilkan grafik secara teks, sehingga dapat
 21 diproses melalui titik untuk membuat grafik sebagai representasi grafis dalam format yang berbeda
 22 seperti .ps, .pdf, dll. *DOT* telah dikembangkan sebagai bagian dari proyek *Graphviz*, yang merupakan
 23 kumpulan alat untuk visualisasi grafik.

2.4.1 Dasar Menggambar Graf

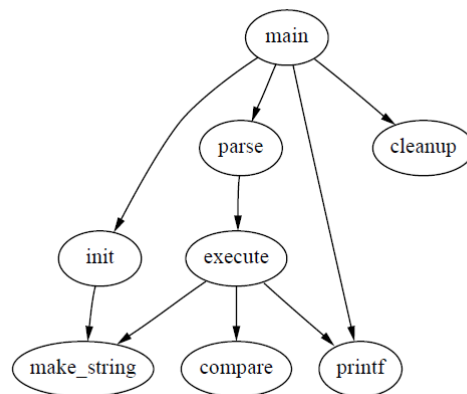
Dot mengambil empat langkah utama dalam menggambar grafik. Langkah pertama menetapkan diskrit peringkat ke node dalam gambar atas ke bawah, menentukan peringkat di koordinat Y. Tepi yang membentang lebih banyak dari satu peringkat dipecah menjadi rantai simpul dan tepi unit. Langkah kedua *node* dalam barisan untuk menghindari penyeberangan. Langkah ketiga menetapkan koordinat *node* X untuk disimpan dibaris terpendek. Langkah terakhir rute tepi splines. Grafik menggunakan dot memiliki tiga jenis *item*: grafik, simpul, dan tepi. Grafik sendiri memiliki dua bentuk yaitu grafik (tidak diarahkan) atau digraph (diarahkan). Karena dot membuat *layout* grafik yang diarahkan maka contoh dalam kasus ini menggunakan digraph.

Gambar graph1 adalah contoh grafik dalam bahasa dot. Baris 1 memberi nama dan jenis grafik. Baris berikut membuat node, tepi, atau subgraf, dan atur atribut. Nama merupakan identifier C, nomor, atau kutipan C. Sebuah simpul diciptakan pertama kali namanya muncul di *file*. Tepian dibuat saat node berada bergabung dengan operator tepi \rightarrow . Pada contoh, baris 2 membuat tepi lalu mengurai dari *parse* ke *execute*. Untuk menjalankan dot pada file ini (dimisalkan graph1.dot) dapat mengetikkan *dot -Tpsgraph1.dot -o graph1.ps* dan akan menghasilkan gambar graph1.

```

1: digraph G {
2: main -> parse -> execute;
3: main -> init;
4: main -> cleanup;
5: execute -> make_string;
6: execute -> printf;
7: init -> make_string;
8: main -> printf;
9: execute -> compare;
10: }

```



Gambar 2.4: Gambar Graph1

2.4.2 Atribut Menggambar

Dalam membuat graf dibutuhkan beberapa atribut untuk menyempurnakan gambar. Atribut tersebut berisi

1. Bentuk dan Label.

Pada bentuk dan label nantinya akan ditentukan *node* akan berbentuk apa dan label pada node akan berisi apa. Secara *default* bentuk dari node sendiri adalah elips. Tetapi ada bentuk lain yang diberikan untuk *node* yaitu kotak, lingkaran, polygon, dll.

2. Tampilan Graf

Simpul dan tepi memiliki atribut warna dan gaya. Penggunaan warna dalam membuat graf memiliki beberapa syarat. Pertama hindari menggunakan terlalu banyak warna cerah. Kedua, ketika node dipenuhi warna gelap label nampaknya lebih mudah dibaca dengan *fontcolor* = putih dan *fontname* = *Helvetica*. Ketiga, menentukan ruang warna dengan mendefinisikan *nodecolor*, *edgecolor*, atau *graphcolor* dalam file library. Misalnya, untuk menggunakan warna RGB, letakkan baris berikut di file lib.ps. / nodecolor setrgbcolor bind def. Gunakan opsi baris perintah -l untuk memuat file ini. *dot -Tps -llib.ps file.dot -ofile.ps*

3. Ukuran Gambar dan Jarak

Seringkali gambar yang dibuat dengan ukuran dan pemisahan *nodes default* terlalu besar untuk target atau untuk ruang yang diizinkan untuk gambar dalam dokumen. Ada beberapa cara untuk mencoba mengatasi masalah ini. Pertama, melihat bagaimana titik pada ukuran tata letak akhir. Tata letak awalnya dibuat secara internal dengan ukuran awal, dengan menggunakan pengaturan *default*. Secara default, *nodes* paling sedikit 0,75 inci dengan lebar 0,5; *font* adalah 14, *nodes* dipisahkan paling sedikit 0,25 dan diberi peringkat oleh 0,5 Tidak ada batasan ukuran atau aspek rasio gambar, jadi jika grafiknya besar, tata letaknya juga besar. Jika tidak menentukan ukuran atau rasio, maka ukuran awal akan dicetak. Cara termudah untuk mengontrol ukuran output gambar adalah dengan mengatur ukuran = x; y pada file grafik (atau pada baris perintah menggunakan -G). Ini menentukan kotak pembatas tata letak akhir.

Tabel untuk atribut menggambar sebagai berikut

Name	Default	Values
Node Attributes		
color	black	node shape color
fontcolor	black	type face color
fontname	Times-Roman	PostScript font family
fontsize	14	point size of label
height,width	.5,.75	height and width in inches
label	node name	any string
layer	overlay range	all, <i>id</i> or <i>id:id</i>
shape	ellipse	ellipse, box, circle, doublecircle, diamond, plaintext, record, polygon, epsf
shapefile		external EPSF file if epsf shape
style		graphics options, e.g. bold, dotted, filled

Gambar 2.5: node attributes

Edge Attributes		
color	black	edge stroke color
decorate		if set, draws a line connecting labels with their edges
dir	forward	forward, back, both, or none
fontcolor	black	type face color
fontname	Times-Roman	PostScript font family
fontsize	14	point size of label
id		optional value to distinguish multiple edges
label		label, if not empty
layer	overlay range	all, <i>id</i> or <i>id:id</i>
minlen	1	minimum rank distance between head and tail
style		graphics options, e.g. bold, dotted, filled
weight	1	integer reflecting importance of edge.

Gambar 2.6: edge attributes

Graph Attributes		
center		when true, centers drawing on page
clusterrank	local	may be global or none
color	black	background or cluster outline color
concentrate		enables edge concentrators when TRUE
fontcolor	black	type face color
fontname	Times-Roman	PostScript font family
fontsize	14	point size of label
label		any string
layerseq		<i>id: id: id...</i>
margin	.5, .5	margin included in page
mclimit	1.0	if set to <i>f</i> , adjusts mincross iterations by (<i>f</i>)
nodesep	.25	separation between nodes, in inches.
nslimit		if set to <i>f</i> , bounds network simplex iterations by (<i>f</i>)(<i>number of nodes</i>)
ordering		out (for ordered edges)
orientation	portrait	may be set to landscape
page		unit of pagination, e.g. 8.5, 11
rank		same, min, or max
rankdir	TB	LR (left to right) or TB (top to bottom)
ranksep	.75	separation between ranks, in inches.
ratio		approximate aspect ratio desired, or fill
size		drawing bounding box, in inches

Gambar 2.7: graph attributes

2.4.3 Subgraf dan Pengelompokan

Subgraf memiliki tiga peran di *Graphviz*. Pertama, subgraf dapat digunakan untuk mewakili struktur grafik, yang menunjukkan bahwa simpul dan tepi tertentu harus dikelompokkan bersama. Informasi pada subgraf ditentukan secara semantik tentang komponen grafik. Tepi dibuat dari setiap simpul di sebelah kiri ke setiap simpul di sebelah kanan. Contohnya sebagai berikut

```
A -> {B C}
```

sama dengan

```
A -> B
```

```
A -> C
```

Kedua, subgraf dapat memberikan konteks untuk mengatur atribut. Sebagai contoh, sebuah subgraf dapat menentukan bahwa warna biru adalah warna *default* untuk semua node yang didefinisikan di dalamnya. Dalam konteks gambar grafik, contohnya sebagai berikut

```
subgraf {
  peringkat = sama; A; B; C;
}
```

Subgraf ini menentukan bahwa simpul A, B dan C semuanya harus ditempatkan pada rangking yang sama jika ditarik menggunakan titik.

Ketiga untuk subgraf secara langsung melibatkan bagaimana grafik akan ditata oleh mesin. Jika nama subgraf dimulai dengan *cluster*, *Graphviz* mencatat subgraf sebagai subgraf *cluster* khusus. Jika didukung, mesin akan melakukan tata letak sehingga simpul milik cluster digambar bersama, dengan keseluruhan gambar cluster yang ada di dalam persegi panjang yang melintang. Subgraf *cluster* bukan bagian dari bahasa DOT, namun hanya konvensi sintaks yang dipatuhi oleh mesin.

2.5 Visualisasi Graf dengan Viz.js

Data terbuka adalah data yang terbuka. Tetapi jika dilihat lagi data terbuka yang dimaksud dalam konteks saat ini adalah data yang bebas diakses, data yang bebas digunakan kembali, dan data dalam format *machine-readable*. Data yang bebas diakses dan bebas digunakan kembali sudah cukup jelas. Semua orang boleh dan diizinkan untuk mengakses dan menggunakan data tersebut untuk keperluannya masing-masing. Pada dasarnya data dalam format PDF, image, atau apapun yang bisa dilihat oleh manusia jika masuk ke dalam kriteria ini. Namun bisa dilihat dan dimengerti saja tidak cukup. Maka dari itu ada kriteria ketiga yaitu "data dalam format *machine-readable*". Format *machine-readable* adalah sebuah format yang memiliki struktur jelas, di mana satu data

berada pada satu baris, dan dalam format terbuka (contohnya format CSV atau JSON). Tujuannya satu, agar data yang ada bisa dimengerti oleh mesin. Dalam hal ini komputer. Jika data dapat dimengerti oleh komputer, maka pengolahan data akan lebih baik, lebih cepat, lebih efisien dan lebih akurat.

JSON sebagai salah satu format terbuka digunakan untuk membuat graf. Graf ini dihasilkan dengan menggunakan *viz.js* yang merupakan mesin pembaca *DOT*. *DOT* sendiri dibuat mdengan melihat struktur JSON. Penulisan *viz* sebagai visualisi graf itu sendiri sebgai berikut

```
Viz(src , options={ format="svg" , engine="dot" , scale , images=[{ path , width , height
```

- *src* adalah string yang mewakili grafik yang akan ditampilkan dalam bahasa **DOT**.

- *options*

- *format* menetapkan *format* keluaran, dan hasilnya salah satu dari "svg", "xdot", "plain", "ps", "json", atau "png-image-element".

- *engine*, mengatur mesin *Graphviz* untuk digunakan, salah satunya "circo", "dot", "fdp", "neato", "osage", or "twopi".

- *scale*, menetapkan faktor skala untuk format "png-image-element".

- *images*, menentukan dimensi gambar untuk digunakan saat membuat nodes dengan atribut gambar.

- *file*, menentukan file yang tersedia untuk *Graphviz* menggunakan *file system* dalam memori. Ini adalah array dari objek, path, data. File dibuat terhadap direktori kerja.

- *total memory*, mengatur memori yang tersedia.

LAMPIRAN A

KODE PROGRAM

Listing A.1: MyCode.c

```

1 // This does not make algorithmic sense,
2 // but it shows off significant programming characters.
3
4
5 #include<stdio.h>
6
7 void myFunction( int input, float* output ) {
8     switch ( array[i] ) {
9         case 1: // This is silly code
10             if ( a >= 0 || b <= 3 && c != x )
11                 *output += 0.005 + 20050;
12             char = 'g';
13             b = 2^n + ~right_size - leftSize * MAX_SIZE;
14             c = (--aaa + &daa) / (bbb++ - ccc % 2 );
15             strcpy(a,"hello_$@?");
16         }
17         count = ~mask | 0x00FF00AA;
18     }
19
20 // Fonts for Displaying Program Code in LATEX
21 // Adrian P. Robson, nepsweb.co.uk
22 // 8 October 2012
23 // http://nepsweb.co.uk/docs/progfonts.pdf

```

Listing A.2: MyCode.java

```

1 import java.util.ArrayList;
2 import java.util.Collections;
3 import java.util.HashSet;
4
5 //class for set of vertices close to furthest edge
6 public class MyFurSet {
7     protected int id; //id of the set
8     protected MyEdge FurthestEdge; //the furthest edge
9     protected HashSet<MyVertex> set; //set of vertices close to furthest edge
10    protected ArrayList<ArrayList<Integer>> ordered; //list of all vertices in the set for each trajectory
11    protected ArrayList<Integer> closeID; //store the ID of all vertices
12    protected ArrayList<Double> closeDist; //store the distance of all vertices
13    protected int totaltrj; //total trajectories in the set
14
15    /*
16     * Constructor
17     * @param id : id of the set
18     * @param totaltrj : total number of trajectories in the set
19     * @param FurthestEdge : the furthest edge
20     */
21    public MyFurSet(int id,int totaltrj,MyEdge FurthestEdge) {
22        this.id = id;
23        this.totaltrj = totaltrj;
24        this.FurthestEdge = FurthestEdge;
25        set = new HashSet<MyVertex>();
26        ordered = new ArrayList<ArrayList<Integer>>();
27        for (int i=0;i<totaltrj;i++) ordered.add(new ArrayList<Integer>());
28        closeID = new ArrayList<Integer>(totaltrj);
29        closeDist = new ArrayList<Double>(totaltrj);
30        for (int i = 0;i <totaltrj;i++) {
31            closeID.add(-1);
32            closeDist.add(Double.MAX_VALUE);
33        }
34    }
35
36 }

```


LAMPIRAN B

HASIL EKSPERIMEN

Hasil eksperimen berikut dibuat dengan menggunakan TIKZPICTURE (bukan hasil excel yg diubah ke file bitmap). Sangat berguna jika ingin menampilkan tabel (yang kuantitasnya sangat banyak) yang datanya dihasilkan dari program komputer.



Gambar B.1: Hasil 1



Gambar B.2: Hasil 2



Gambar B.3: Hasil 3



Gambar B.4: Hasil 4