

PENGUJIAN PERANGKAT LUNAK (DPH2C2)

**PROGRAM STUDI D3 MANAJEMEN INFORMATIKA – UNIVERSITAS TELKOM
SEMESTER GENAP TAHUN AKADEMIK 2016-2017**

PERTEMUAN 5

MATERI : WHITE BOX TESTING – BAGIAN 1

White Box Testing (Bagian 1)



Kajian 2

Jenis	Kajian	Kompetensi Dasar	Pokok Bahasan	Sub Pokok Bahasan	Materi Bahasan
Teori	Kajian 2: Teknik Pengujian White Box	Mahasiswa mampu membuktikan teknik pengujian white box	Basis Path Testing	Basis Path Testing,	Basis Path Testing, sejarah, definisi dan Notasi
Teori			Flowgraph dan Cyclomatic Complexity	Flowgraph dan Cyclomatic Complexity	<ul style="list-style-type: none">- regions- cyclomatic complexity- Flowgraph kondisional- Flowgraph pengulangan- independent path
Teori			Basis Path Worksheet dan data uji	Basis Path Worksheet dan data uji	<ul style="list-style-type: none">- kegunaan basis path worksheet- cara melengkapi basis path worksheet- cara menentukan data uji

Definisi White Box Testing (review)



- ▶ *White box testing* merupakan strategi pengujian (*testing*) yang diterapkan pada mekanisme internal suatu sistem atau komponen (IEEE, 1990). Strategi ini digunakan untuk ‘melihat’ mekanisme internal dari suatu produk perangkat lunak, khususnya untuk mengamati struktur dan logika kode-kode program yang ditulis

Nama Lain

- ▶ Beberapa sinonim dari *White Box Testing* (Cognizant, 2010), diantaranya:
 1. *Glass box testing,*
 2. *Clear box testing,*
 3. *Open box testing,*
 4. *Transparent box testing,*
 5. *Structural testing,*
 6. *Logic driven testing,*
 7. *Design based testing.*

Proses White Box Testing

Tahap 1	Input	<ul style="list-style-type: none">• Requirement• Functional spesification• Detailed designing of documents• Proper source code
Tahap 2	Processing Unit (Analyzing internal working only)	<ul style="list-style-type: none">• Perform risk analysis to guide whole testing process• Proper test plan• Execute test cases and communicate results
Tahap 3	Output (Prepare final report)	<ul style="list-style-type: none">• Test cases document and result• Acceptance test document

Proses White Box Testing

- ▶ Kode-kode program yang dituliskan berfungsi untuk menggambarkan struktur dan alur logika (*logical path*) untuk suatu operasi tertentu, sehingga perlu dilakukan pengujian untuk memastikan bahwa struktur dan alur logika yang telah dibuat akan menghasilkan suatu nilai yang benar atau valid

Yang menjadi perhatian

- ▶ Struktur dan alur logika yang akan diuji menggunakan strategi ini meliputi (Pressman, 2001):
 1. Alur independen yang terdapat pada suatu komponen program,
 2. Keputusan-keputusan logika baik dari sisi yang bernilai benar (*true*) maupun salah (*false*),
 3. Alur pengulangan,
 4. Struktur data internal untuk memastikan kebenaran nilai-nilainya.

Keputusan Logika

- ▶ Keputusan-keputusan logika dihasilkan dari suatu pengujian kondisi (*conditional testing*) yang melibatkan variasi dari sintaks algoritma 'If ... Then ...' seperti:
 1. If ... Then ... Else ...
 2. If ... Then ... Else if ... Then ...
 3. Case ... Of ...

Alur Pengulangan

- ▶ Alur pengulangan digunakan untuk membentuk suatu struktur operasi yang akan dieksekusi secara berkali-kali dengan jumlah eksekusi terbatas. Struktur pengulangan (*loop*) ini dituliskan dengan variasi sintaks algoritma sebagai berikut:
 1. While ... Do ...
 2. Repeat ... Until ...
 3. For ... To ... Do ...

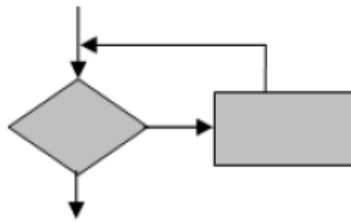
Bentuk Pengujian White Box

1. Loop Testing
2. Branch Testing
3. Data Flow Testing
4. Control Flow Testing
5. Basis Path Testing

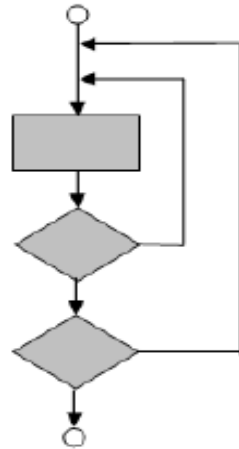
Loop Testing

- ▶ Tipe Pengujian White Box yang fokus pada validitas struktur pengulangan
- ▶ Struktur pengulangan sebenarnya mudah untuk diuji, kecuali jika muncul kebergantungan antar loop, di antara loop atau di dalam kode yang mengandung loop.
- ▶ Ada 4 jenis Loop/pengulangan:
 1. *Simple Loop*
 2. *Nested Loop*
 3. *Concatenated Loop*
 4. *Unstructure Loop*

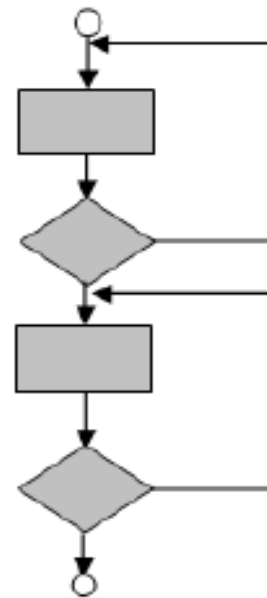
Loop Testing



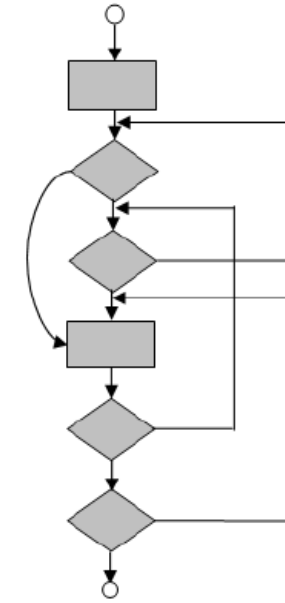
Simple Loop



Nested Loop



Concatenated Loop



Unstructured Loop

Branch Testing

- ▶ Kata lainnya adalah: *conditional testing* atau *decision testing*
- ▶ Pengujian ini memastikan bahwa setiap kemungkinan outcome dari kondisi diuji minimal sekali.
- ▶ Pada pengujian ini test cases di desain untuk melatih control flow branches atau decision points dalam sebuah modul. Semua cabang pada cabang tertentu, diuji minimal sekali (Sharon, 2009).

Contoh Branch Testing

- ▶ Jika diketahui kode sbb:

Read Length (l)

Read Breadth (b)

IF l > 4 THEN Print "invalid length" ENDIF

IF b > 5 THEN Print "invalid breadth" ENDIF

*AREA= (l*b)*

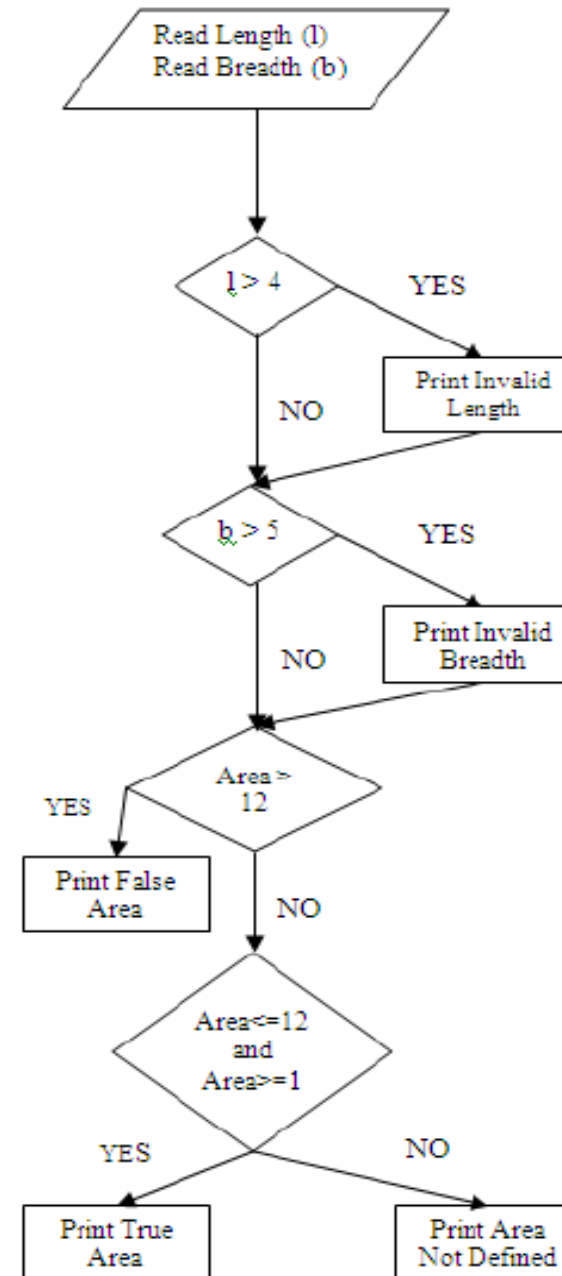
Print ("Area="Area)

IF Area>12 THEN Print "False Area"

ELSE IF Area<=12 and Area>=1 THEN Print "True Area"

ELSE IF Area<1 THEN Print "Area Not Defined" ENDIF

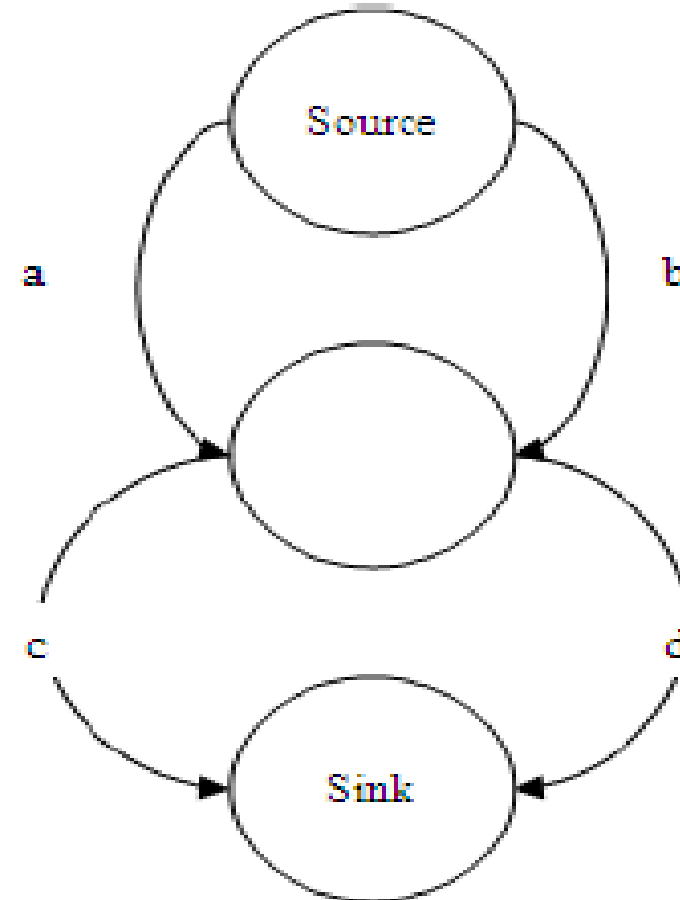
Maka branch testingnya menjadi seperti ini



Data Flow Testing

- ▶ Data Flow Testing adalah pengujian white box yang melihat bagaimana data bergerak dalam sebuah program.
- ▶ Pada pengujian ini, dibuat sebuah control flow graph yang menggambarkan mengenai bagaimana variable pada program didefinisikan dan digunakan
- ▶ Control flow graph adalah diagram yang menggambarkan sebuah program dan eksekusinya

Contoh sederhana dari Data Flow Testing



Control Flow Testing

- ▶ Adalah sebuah strategi pengujian struktural yang menggunakan control flow dari program sebagai model control flow testingnya. Agar dapat menyederhanakan path atau alur program.
- ▶ Penelitian menunjukkan bahwa metode ini menemukan lebih dari 50% dari keseluruhan bugs pada saat unit testing (unit testing didominasi oleh control flow testing).
- ▶ Control flow testing lebih efektif untuk unstructured code daripada structured code.
- ▶ Karena pada structured programming languages/code, bugsnya sudah dapat diminimalisir.

Basis Path Testing

- ▶ *Basis path testing* merupakan suatu metoda yang digunakan dalam teknik *white box testing*.
- ▶ Metoda ini pertama kali diusulkan oleh Tom McCabe tahun 1976
- ▶ Metoda *basis path* ini sangat bermanfaat bagi seorang penguji perangkat lunak dalam menentukan:
 1. Ukuran kompleksitas logika dari suatu struktur program, *procedure* atau *function*.
 2. Menggunakan nilai kompleksitas untuk menentukan *basis set* (himpunan dasar) alur logika yang akan dieksekusi.
- ▶ Dengan metode ini, penguji perangkat lunak dapat menentukan kasus uji yang diterapkan pada *basis set* dan menjamin bahwa setiap baris program akan dieksekusi minimal satu kali.

Basis Path Testing

- ▶ Metoda *basis path testing* ini memerlukan masukan berupa *source code* atau algoritma dari suatu perangkat lunak. Setelah *source code* atau algoritma ini didapatkan, maka langkah-langkah yang dilakukan dalam menjalankan metoda ini adalah sebagai berikut:
 1. Menggambarkan alur logika menggunakan notasi yang telah ditentukan. Gambar alur logika ini disebut sebagai *flow graph*.
 2. Menentukan *cyclomatic complexity*.
 3. Menentukan *basis set* dari alur-alur yang independen.
 4. Membuat data uji untuk kemudian dieksekusi pada setiap alur.

Notasi Flowgraph pada Basis Path Testing

Sequence



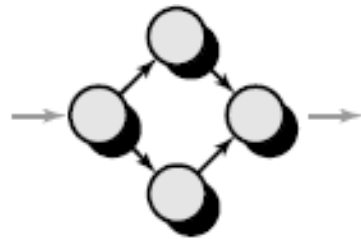
While



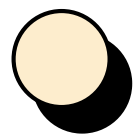
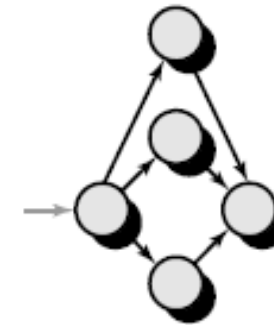
Until



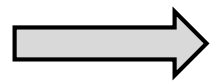
If



Case



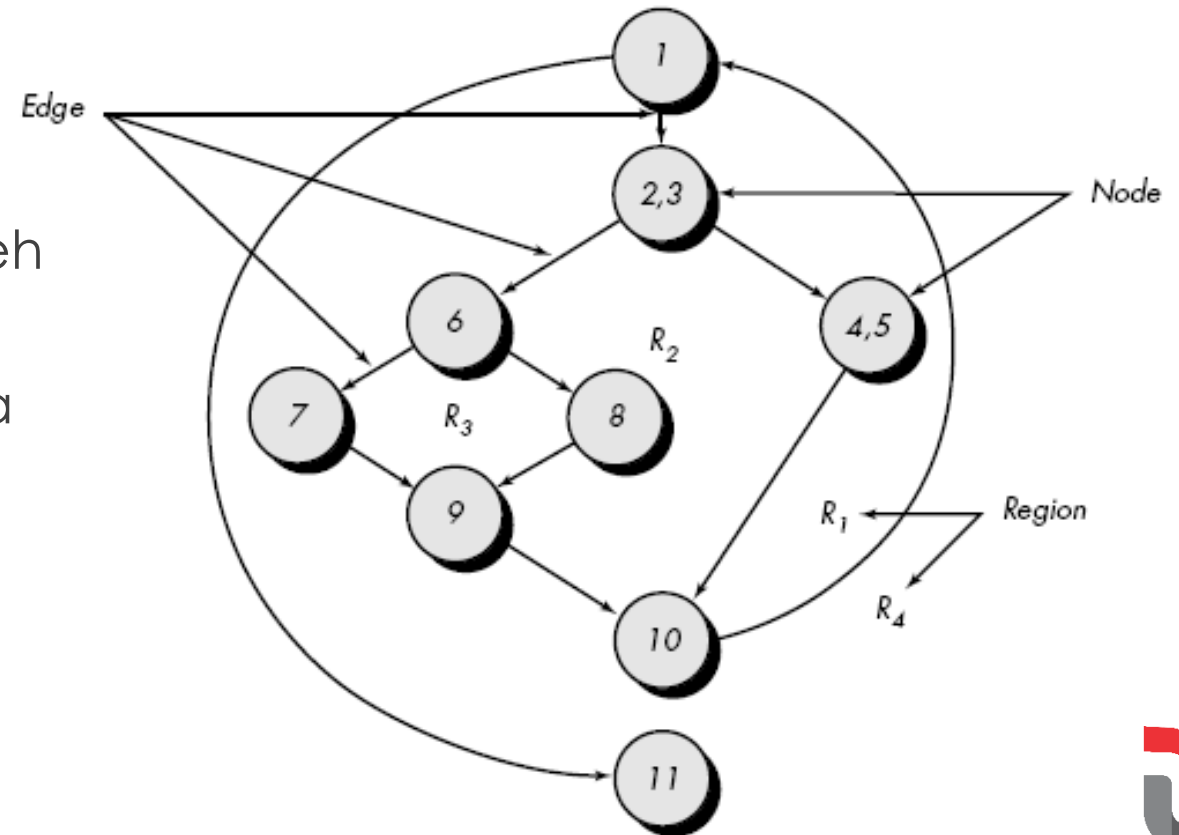
Lingkaran (flow graph node)
Menyatakan satu atau beberapa statement prosedural



Panah (edges / links)
Menyatakan aliran kendali (sama dengan panah flowchart)

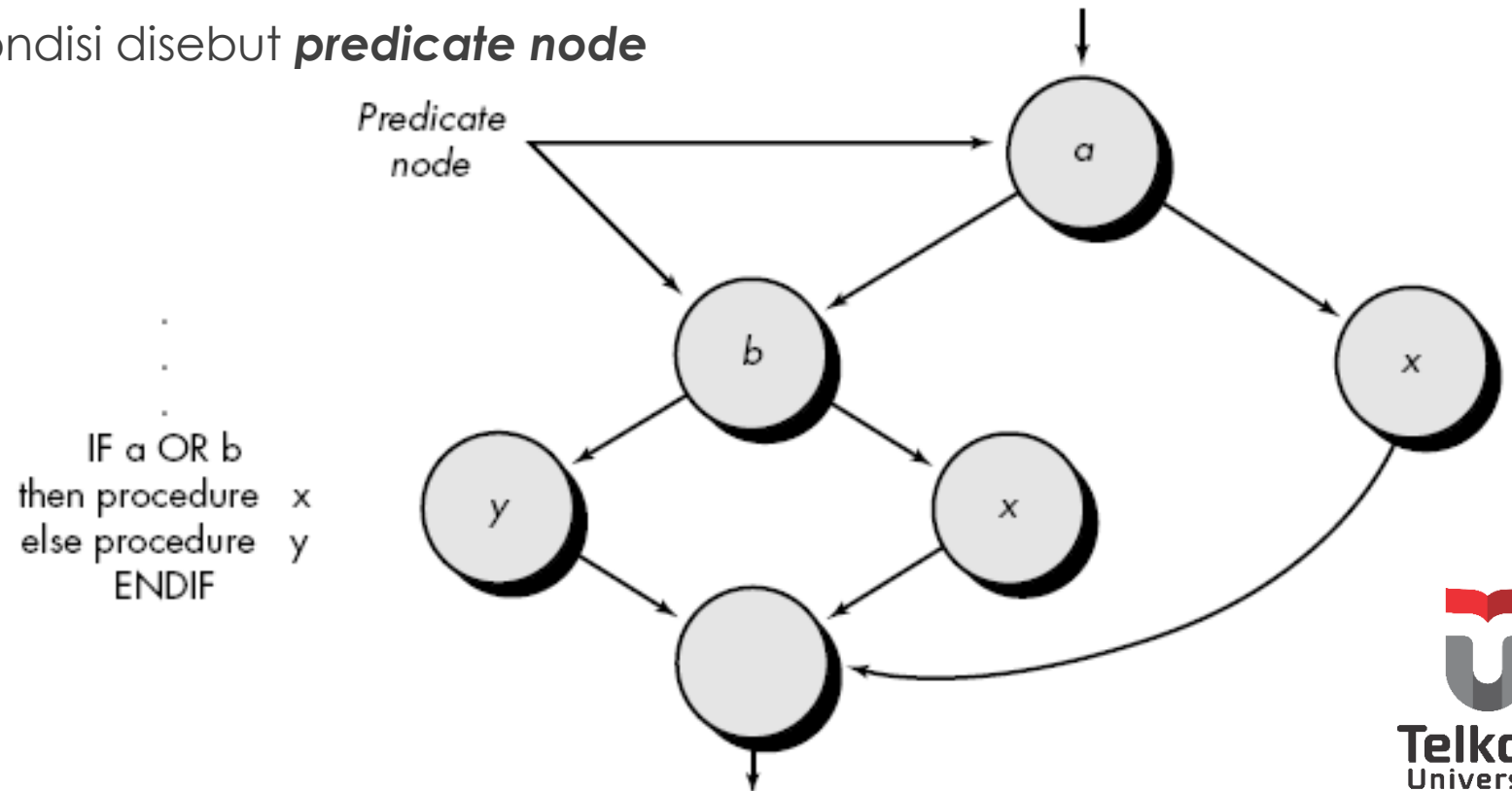
Istilah Pada Flowgraph - Regions

- ▶ Adalah suatu area yang dibatasi oleh edges dan nodes
- ▶ Pada saat menghitung regions, area di luar graph ikut ditambahkan



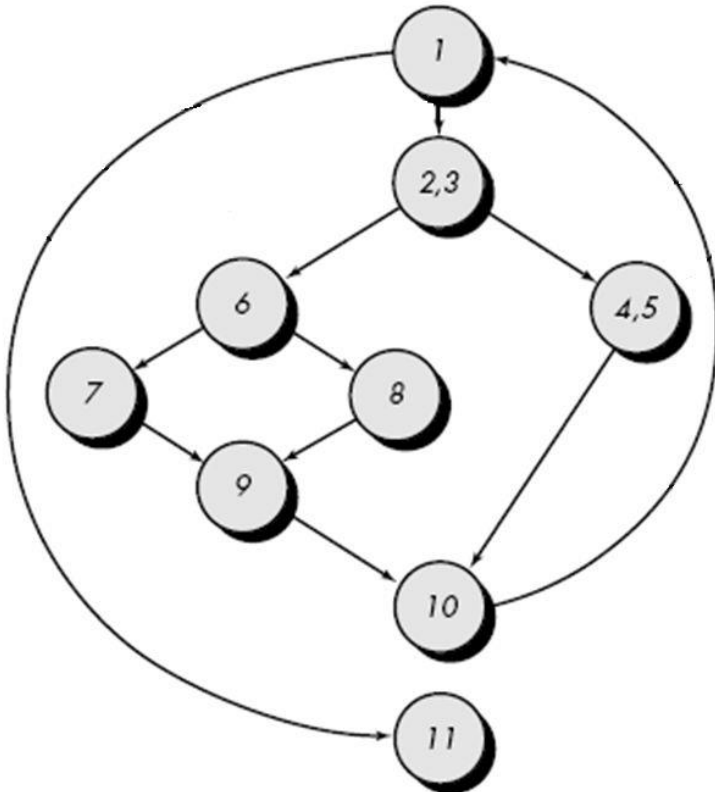
Compound Conditions (Istilah pada Flowgraph)

- ▶ Kondisi majemuk terjadi apabila satu atau lebih dari operator Boolean (AND, OR, NOR, NAND) muncul pada pernyataan kondisi.
- ▶ Setiap *node* yang terdapat kondisi disebut **predicate node**



Independent Path (Istilah pada Flowgraph)

- ▶ Adalah jalur yang melintasi minimal satu kumpulan pernyataan baru atau sebuah kondisi baru pada program.



Path 1 : 1 – 11

Path 2 : 1 – 2 – 3 – 4 – 5 – 10 – 1 – 11

Path 3 : 1 – 2 – 3 – 6 – 8 – 9 – 10 – 1 – 11

Path 4 : 1 – 2 – 3 – 6 – 7 – 9 – 10 – 1 – 11

1-2-3-4-5-10-1-2-3-6-8-9-10-1-11 **bukan**
Independent path, karena merupakan
kombinasi dari *paths* yang sudah
didefinisikan.

Cyclomatic Complexity (Istilah pada Flowgraph)

▶ Untuk menentukan banyaknya *independent path* yang merupakan *basis path*

▶ Rumus:

$$V(G) = E - N + 2$$

$$V(G) = P + 1$$

▶ E : Edge

▶ N : Node

▶ P : Predicate node

Latihan

Tentukan hal-hal berikut:

1. Predicate node
2. Jumlah region
3. Independent path
4. $V(G)$

