PENGUJIAN PERANGKAT LUNAK (DPH2C2)

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

PERTEMUAN 5 & 6

MATERI : BASIS PATH TESTING

Telkom University

Basis Path Testing

Telkom University

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	- regions - cyclomatic complexity - Flowgraph kondisional - Flowgraph pengulangan - independent path
Teori			Basis Path Worksheet dan data uji	Basis Path Worksheet dan data uji	- kegunaan basis path worksheet - cara melengkapi basis path worksheet - cara menentukan data uji

Telkom University

Techniques for Testing - Dynamic

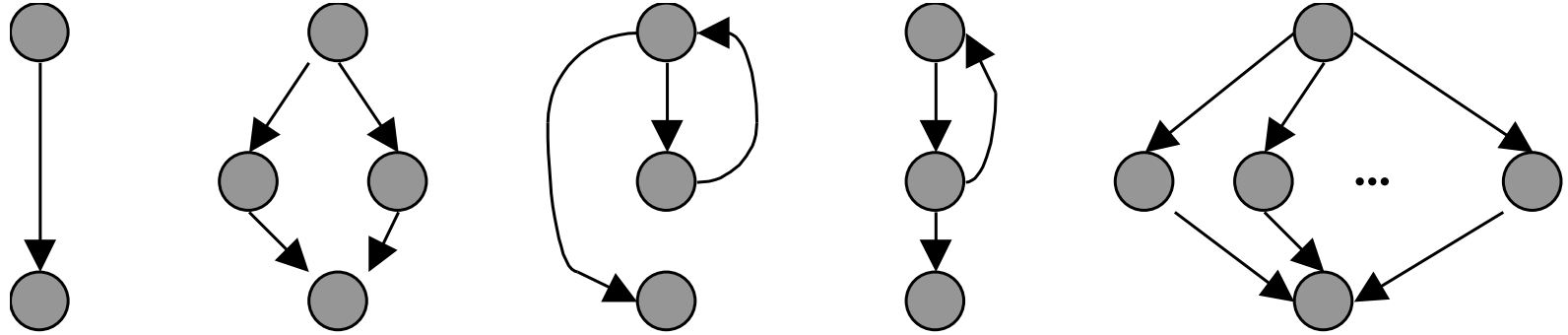
- Basis Path Testing
  - a white-box testing technique, proposed by Tom McCabe, 1976
  - to derive a logical complexity measure of a procedural design, and use this measure as a guide for defining a basis set of execution paths
  - test cases derived to exercise every *statement and branch* in the program at least once during testing (statement/branch coverage)
  - if every condition in a compound condition is considered, *condition* coverage can be achieved
  - Steps:
    - Draw a (control) *flow graph*, using the flowchart or the code
    - Calculate the *cyclomatic complexity*, using the flow graph
    - Determine the *basis set* of linearly independent paths
    - Design *test cases* to exercise each path in the basis set

Telkom University

Basis Path Testing

- Flow Graph
  - used to depict program control structure
  - can be drawn from a flowchart (a procedural design representation)
  - can be drawn from a piece of source code
  - Flow Graph Notation
    - a flow graph composed of edges and nodes
    - an edge starts from a node and ends to another node

Sequence if-then-else While Repeat-until Case



Telkom University

Basis Path Testing

- Cyclomatic Complexity
  - a software metric that provides a quantitative measure of the logical complexity of a program
  - Basis set: is a maximal linearly independent set of paths through a graph
  - An independent path: is any path through a program that introduces at least one new set of processing statements or a new condition (I.e. at least one new edge in a flow graph)
  - Cyclomatic complexity defines the number of independent path in the basis set of a program
  - gives an upper bound for the number of tests that must be conducted to achieve statement/branch/condition coverage
  - How to calculate cyclomatic complexity:

$$cc = e - n + 2p$$

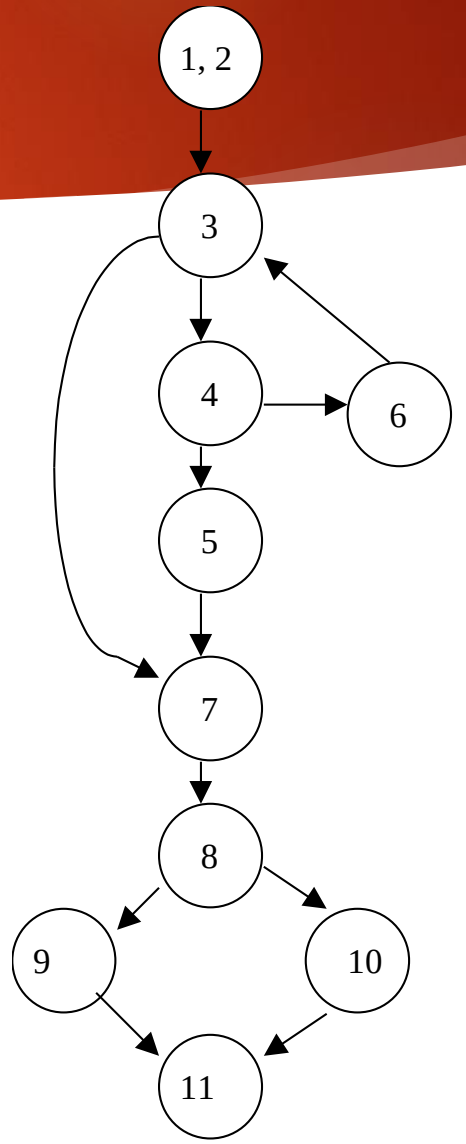
  - e - number of edges; n - number of nodes; p - number of components;
  - if all nodes in a graph are connected, then p = 1, thus

$$cc = e - n + 2$$

Telkom University

Basis Path Testing: Example 2

1. Draw a flow graph
   - see slide 6-24: source code, flow graph
2. Calculate cyclomatic complexity
   - e = 12; n = 10; p = 1
   - cc = 12 - 10 + 2 x 1 =4
3. Determine a basis set of independent paths
   - expect to specify 4 independent paths
   - p1: *1-2-3-7-8-9-11*
   - p2: 1-2-3-*4-5*-7-8-9-11
   - p3: 1-2-3-4-5-7-8-*10*-11
   - p4: 1-2-3-4-6-3-7-8-10-11 (1 or more times)
   - HOWEVER: by reading source code, we found
     - 3-7 => 10; 5 => 9
     - p1 and p3 must be modified



Telkom University

Basis Path Testing: Example 2

- 3. Determine a basis set of independent paths
  - if p3 modified, it would be the same as p2. Thus p3 should be deleted.
  - But the new paths introduced by p3 (8-10-11) must be covered by other paths! We found p4 covers them.
  - Modify p1, delete p3, we can have three independent paths
    - p1: 1-2-3-7-8-10-11
    - p2: 1-2-3-4-5-7-8-9-11
    - p3: 1-2-3-4-6-3-7-8-10-11
  - if you study the program carefully, you will find the following is better
    - p1: 1-2-3-7-8-10-11 (insert x when a[] is empty)
    - p2: 1-2-3-4-5-7-8-9-11(insert x when a[1]=x)
    - p3: 1-2-3-4-6-3-4-5-7-8-9-11 (insert x when a[i]=x,i>1, n>=i)
    - p4: 1-2-3-4-6-3-7-8-10-11 (insert x when a[] is not empty and x is not in a[]; p4 does not introduce any new edge but it exercises a new combination of the program logic!)

Telkom University

Basis Path Testing: Example 2

- 4. Design test cases
  - Path 1 test case: 1-2-3-7-8-10-11 (insert x when a[] is empty)
    - input data: n=0; x=8; a[1]=0; b[1]=0;
    - expected results: a[1]=8; b[1]=1; n=1;
  - Path 2 test case: 1-2-3-4-5-7-8-9-11(insert x when a[1]=x)
    - input data: n=3; x=9; a[1]=9; a[2]=2; a[3]=3;b[1]=2;b[2]=5;b[3]=8;
    - expected results: b[1]=3
  - Path 3 test case: 1-2-3-4-6-3-4-5-7-8-9-11 (insert x when a[i]=x,i>1, n>=i)
    - input data: n=3;x=3;a[1]=9;a[2]=2;a[3]=3;b[1]=3;b[2]=2;b[3]=8;
    - expected results: b[3]=9
  - Path 4 test case: 1-2-3-4-6-3-7-8-10-11 (insert x when a[] is not empty and x is not in a[])
    - input data: n=3;x=6;a[1]=9;a[2]=2;a[3]=3;b[1]=3;b[2]=2;b[3]=8;
    - expected results: a[4]=6; b[4]=1; n=4;

Telkom University

Transformasi dalam Flow Graph pada Basis Path Testing

```
i = 1;
total.input = total.valid = 0;
sum = 0;
DO WHILE value[i] <> -999 AND total.input < 100
        increment total.input by 1;
                IF value[i] >= minimum and value[i] <= maximum
                        THEN increment total.valid by 1;
                                sum = sum + value[i]
                        ELSE skip
                ENDIF
                increment I by 1;
ENDDO

IF total.valid > 0
        THEN average = sum / total.valid;
        ELSE average = -999;
ENDIF

END average
```

Telkom University

Transformasi dalam Flow Graph pada Basis Path Testing

Petakan setiap baris dalam algoritma/badan program dalam sebuah node. INGAT !!! Satu node dapat terdiri atas satu instruksi/baris algoritma, dapat juga terdiri atas beberapa instruksi algoritma yang tidak memiliki nilai BOOLEAN.

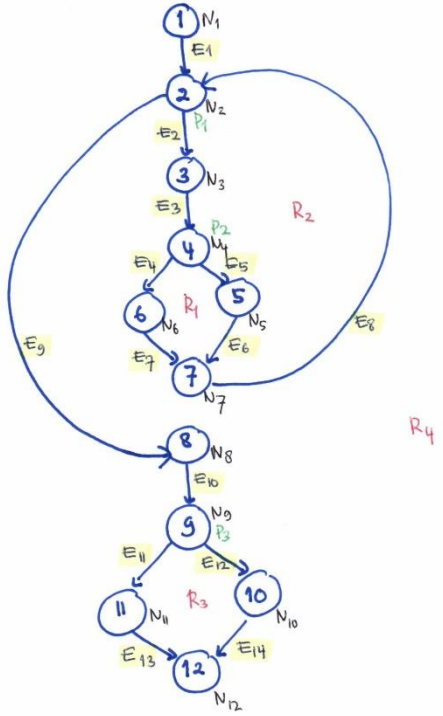
```
i = 1; (1)
total.input = total.valid = 0; (1)
sum = 0; (1)
DO WHILE value[i] <> -999 AND total.input < 100 (2)
        increment total.input by 1; (3)
                IF value[i] >= minimum and value[i] <= maximum (4)
                        THEN increment total.valid by 1; (5)
                                sum = sum + value[i] (5)
                        ELSE skip (6)
                ENDIF (7)
                increment I by 1; (7)
ENDDO (8)

IF total.valid > 0 (9)
        THEN average = sum / total.valid; (10)
        ELSE average = -999; (11)
ENDIF (12)

END average (12)
```

Telkom University

Transformasi dalam Flow Graph pada Basis Path Testing



Σ Node (N) = 12
Σ Edge (E) = 14
Σ Predicate Node (P) = 3
Σ Region (R) = 4

Cyclomatic Complexity (V(G)) = P + 1 = 3 + 1 = 4
= E − N + 2 = 14 − 12 + 2 = 4
V(G) = R
= 4
=

Telkom University

Latihan

Buatlah transformasi algoritma disamping dalam bentuk flowgraph dengan dihitung berapa jumlah Node (N), Predicate Node (P), Region (R), Edge (Edge) dan V(G)

```
Public void tes_cyclomatic(int a, int b, int c){
int total,a;
  While a <= 10 {
    Count++;
    If (b < 10) {
      cut = b/100 * c;
      cut = cut * -1;
    }
    Else if (b > 10) {
          cut = b/100 * c;
        }
        Else {
          cut = 0;
        }
  a++;
  total = 1000 + cut;
  }
}
```

Telkom University