- Matriks representasi akan sangat membantu dalam menangani masalah yang berkaitan dengan mengidentifikasi dan melacak Path.

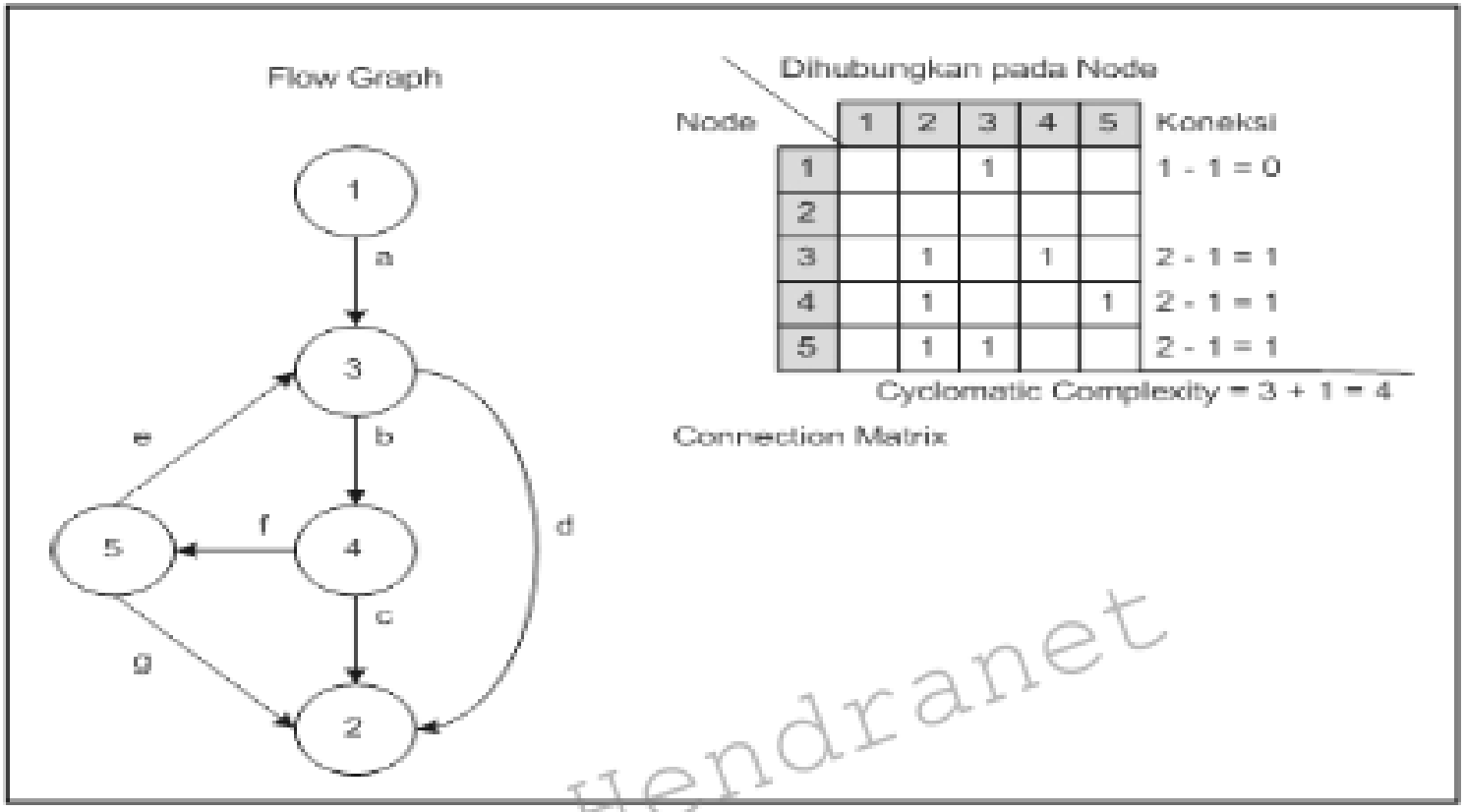
Graph Matrix

- Jumlah baris dan kolom sama dengan node dan identifikasi baris dan kolom sama dengan indentifikasi node, serta isi data menunjukan keberadaan penghubung antar node (edges)

Pembobotan:

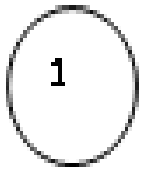
- Kemungkinan jalur (*Edge) akan dilalui / dieksekusi.*
- *Waktu proses* diharapkan pada jalur selama proses transfer dilakukan.
- Memori yang dibutuhkan selama proses transfer dilakukan pada jalur.
- Sumber daya (resources) *yang dibutuhkan selama proses transfer dilakukan pada jalur.*

Flow graph → graph matrix



Example 1:

For a graphical representation:



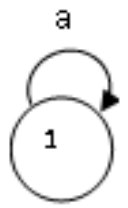
Corresponding Relation Matrix and also, Connection Matrix representation is:

	1
1	0

There is no path segment in this representation from node 1 to node 1.

Example 2:

For a graphical representation:



Corresponding Relationship Matrix Representation is:

	1
1	A

And, the Connection Matrix is:

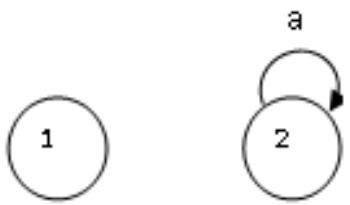
	1
1	1

The path segment from node 1 to node 1 is a.

Alternatively, the path segment using link weights notation from node 1 to node 1 is a_{11}.

Example 3:

For a graphical representation:



Corresponding Relationship Matrix representation is:

	1	2
1	0	0
2	0	a

And, the Connection Matrix is:

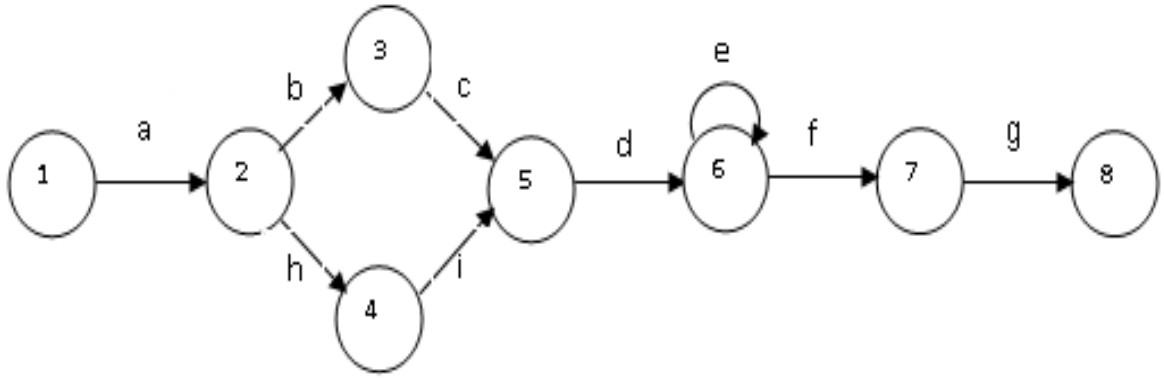
	1	2
1	0	0
2	0	1

The path segment from node 1 to node 2 is a.

Alternatively, the path segment using link weights notation from node 1 to node 2 is a_{22}.

Example 4:

For a graphical representation:



One path segment from node 1 to node 7 is:

abcdf

Alternatively, the path segment using link weights notation from node 1 to node 7 is:

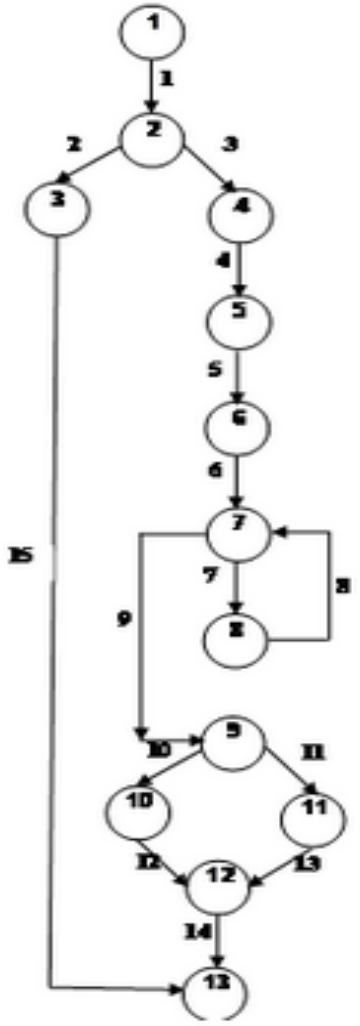
$a_{12}\ a_{23}\ a_{35}\ a_{56}\ a_{67}$

```
function sdivisor (n: integer): integer;
  var d {current divisor and member of odd sequence},
  R{integer less than or equal to square root of n}: integer;
  begin {finds the smallest exact divisor of an integer n, returns 1 if n prime}
   {assert: n>0}
  if not odd(n) then
    sdivisor:=2
 else
begin {terminate search for smallest divisor at sqrt (n)}
  r: = trunc(sqrt(n));
  d: =3;
  {invariant: d= < r+1^no odd integer in [3..d-2] exactly
  divides n}
  while (n mod d < >0) and (d < r) do d: = d+2;
  {assert: d is smallest exact divisor or n^d = < r V (d = < r+1)^n
  is prime}
  if n mod d =0 then
    sdivisor :=d
  else
    sdivisor :=1
  end
end
```

Step 1: Flow Graph Representation

Above program element is represented in the form of flow graph as follows:

Fig. 11.1 Case Study- Flow Graphical Representation of Program Element



Step 2: Cyclomatic Complexity Calculation

In the above flow graph,

Number of nodes, n= 15

Number of Edges, e= 13

Cyclomatic Complexity, V (g) = e – n + 2

= 15 – 13 + 2

= 4

Alternatively, Step 2: Cyclomatic Complexity Calculation can be done using Matrix Representation by arriving at Matrix equivalent representation of Fig. 11.1 Case Study- Flow Graphical Representation of Program Element as follows:

- Construct Matrix for a given Graph
- Perform Row Sum and subtract 1. While doing it ignore rows with no 1's.
- Add all these Row Sum values and add 1 in order to get Cyclomatic Complexity value.

The entire exercise is shown in the following table.

	1	2	3	4	5	6	7	8	9	10	11	12	13	Row Sum -1
1		1												0
2			1	1										1
3													1	0
4					1									0
5						1								0
6							1							0
7								1	1					1
8							1							0
9										1	1			1
10												1		0
11												1		0
12													1	0
13														
										Column Sum +1				**4**

Thus, Cyclomatic Complexity value is 4 which match with the value that we obtained from graphical representation.

Corresponding Relationship Matrix representation is:

	1	2	3	4	5	6	7	8
1		a						
2			b	h				
3					c			
4					i			
5						d		
6						e	f	
7								g
8								

And, the Connection Matrix is:

	1	2	3	4	5	6	7	8
1		1						
2			1	1				
3					1			
4					1			
5						1		
6						1	1	
7								1
8								

Step 3: Recommendation as a Tester about the Code

- For business applications, testers can use following heuristics as the basis for recommending on quality of code.
- If Cyclomatic value is less than 5, the code is well written and simple. Thus, code can be accepted and also, it can be allocated to lesser experienced tester for testing.
- If Cyclomatic value is any value between 5 to 10, the code is well written and complex. Thus, code can be accepted and also, it can be allocated to experienced tester for testing.
- If Cyclomatic value is any value greater than 10, the code is badly written and is not well structured. Thus, code can be returned back to developer with a recommendation to rewrite the code by keeping focus on structuring and modularity.
- In the given example, since Cyclomatic Complexity is 4, the code is well written and lower complexity. Code is easy to maintain. Testing can be managed by tester with lesser expertise.

Step 4: Independent Paths

- Following are 4 independent paths that represent non-repeating single-entry and single-exit paths from start node to end node:
  - 1, 2,3,13
  - 1,2,4,5,6,7,8,7,9,10,12,13
  - 1,2,4,5,6,7,9,10,12,13
  - 1,2,4,5,6,7,9,11,12,13