

Złożoność strukturalna programów, metryki złożoności modułowej

Wzmocnienie powiązań wewnątrz-modułowych prowadzi do zmniejszenia oddziaływań między modułami oraz poprawy struktury oprogramowania.

Metryki rozmiaru [12]

SLOC

Jest to liczba wierszy kodu źródłowego programu liczona niezależnie od liczby instrukcji lub fragmentów instrukcji znajdujących się w każdym wierszu. Nie wlicza się wierszy z komentarzami lub pustych wierszy.

SLOC jest powszechnie używaną metryką do szacowania nakładów pracy nad programem oraz jest mocno skorelowana z testowalnością, konserwowalnością i zrozumiałością.

S/C

Metryka ta jest liczbą wszystkich elementów programu należących do bloków logicznych:

- inicjowanie zmiennych sterujących **int** i=0
- porównanie $i < 10$
- zwiększanie zmiennej sterującej $i++$
- liczba instrukcji w każdym bloku **for** (;;) {...}

Żetony

Jest to zbiór metryk, które określają liczbę:

- η_1 - liczbę typów operatorów (słownik typów operatorów), czyli liczbę: operatorów predefiniowanych (logicznych, arytmetycznych, przypisania, relacyjnych itp.), słowa kluczowe instrukcji (**while**, **if**, **else**, **do**), nazwy funkcji
- η_2 - liczbę typów argumentów (słownik typów argumentów), czyli liczbę: wszystkich symboli reprezentujących dane przy deklaracji i definicji
- η_3 - liczbę wszystkich wystąpień operatorów
- η_4 - liczbę wszystkich wystąpień argumentów

Metryki logicznej struktury programu, czyli przepływu sterowania

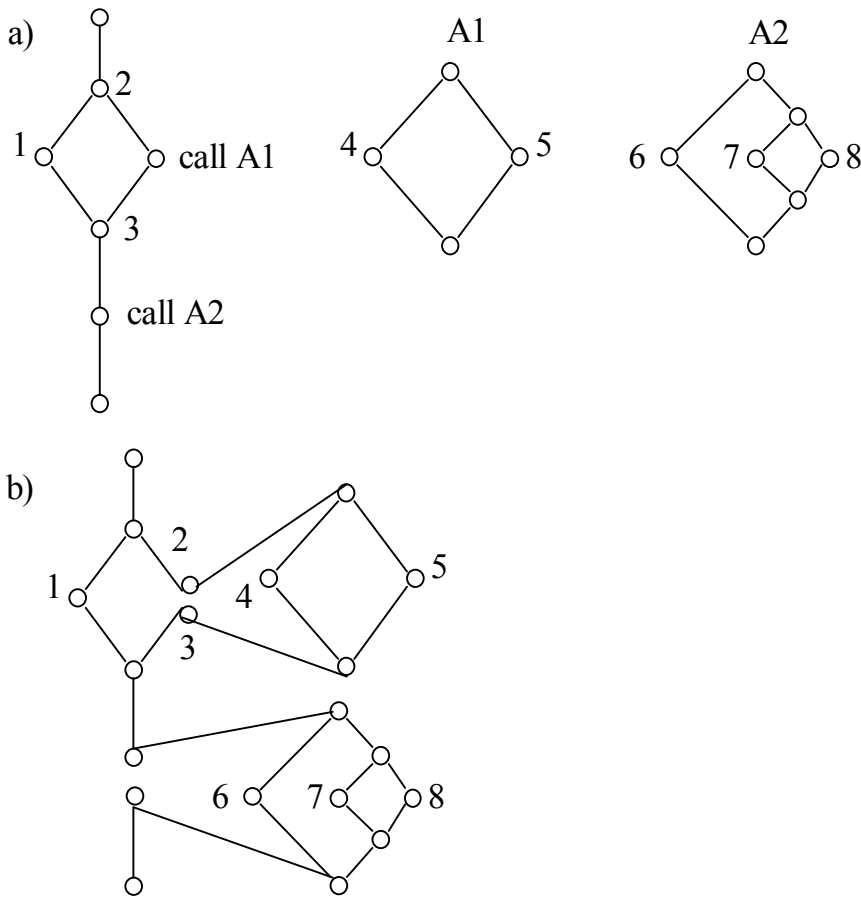
Liczby cyklomatyczne McCabe

$$V_{LI}(G) = e - n + p + 1, \quad V(G) = e - n + 2 * p$$

Liczba ta jest wyznaczana na podstawie grafu przedstawiającego drogi sterowania w programie, gdzie n jest liczbą wierzchołków grafu reprezentujących poszczególne instrukcje, w tym wywołania funkcji, e jest liczbą krawędzi grafu reprezentujących połączenia poszczególnych realizacji instrukcji, p jest liczbą podgrafów rozłącznych, a każda funkcja stanowi niezależny podgraf, którego wywołanie jako wierzchołek jest umieszczony w innym podgrafie.

Metryka $V(G)$ uwypukla istnienie funkcji za pomocą składnika $2 * p$, $V_{LI}(G)$ natomiast wywołanie funkcji traktuje na równi z innymi instrukcjami.

Przykład 8.1



$$\begin{aligned} \text{a) } V(G) &= e - n + 2 * p \\ &= 19 - 18 + 2 * 3 = 7 \\ V_{LI}(G) &= e - n + p + 1 = \\ &= 19 - 18 + 3 + 1 = 5 \end{aligned}$$

$$\begin{aligned} \text{b) } V(G) &= V_{LI}(G) = \\ &= e - n + 2 = \\ &= 23 - 20 + 2 = 5 \end{aligned}$$

Zbiór wszystkich decyzji unikatowych:
 $|\{1\ 6; 2\ 4\ 3\ 7; 2\ 5\ 3\ 6; 2\ 5\ 3\ 8, 1\ 7\}| = 5$

Ścieżki pozostałe dają się wyprowadzić z podstawowych 5-u
 $\{1\ 8\} = \{1\ 6\} +$
 $\{2\ 5\ 3\ 8\} - \{2\ 5\ 3\ 6\}$

Rys. 1. Przepływ sterowania a) trzy rozłączne podgrafy b) wersja jednografowa

	Main	funkcja A1	funkcja A4	Całość	liczba decyzji	całkowita l. decyzji
$V(G)$	2	2	3	7	5	5
$V_{LI}(G)$	2	2	3	5	5	5

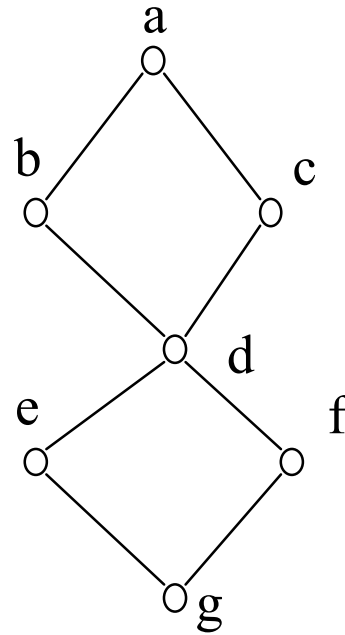
```
#include <stdio.h>
#include <math.h>
void A1 (float a, float b, float c, int& B, float& x1, float& x2)
{float pom=2*a, d= b*b-4*a*c;
 if (d < 0) B=1;
 else
 {B=2; d=sqrt(d); x1=(-b-d)/pom; x2=(-b+pom)/pom;}}
void A2(int B)
{ if (B<1) printf("Brak rownania kwadratowego\n");
 else
 if (B==1) printf("Brak pierwiastkow rzeczywistych\n");
 else printf("Rownanie ma pierwiastki rzeczywiste\n");}
void main()
{float a=1,b=2,c=1, x1, x2; int B;
 if (a==0) B=0;
 else A1(a,b,c,B,x1,x2);
 A2(B);}
```

Przykład 8.2

Problem pętli

a) dwie pętle sekwencyjne

```
a: while (x >= 0)
c: { x=x-y;           (gdy a==true)
    }
b:           (gdy a==false)
d: while (y >= 10) (koniec a)
f: { x=x+1;         (gdy d==true)
    y=y-1;
    }
e:           (gdy d==false)
g:           (koniec d, koniec programu)
```



$$V(G)=e-n+2*p=3$$

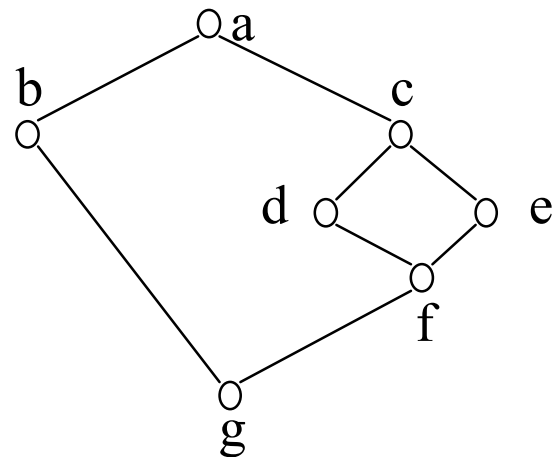
$$V_{LI}(G)=e-n+p+1=8-7+2=3$$

$$SLOC=7$$

$$S/C=7$$

b) podwójna pętla zagnieżdżona

```
a: while (x >= 0)
    { x=x-y;           (gdy a==true)
c:   while (y >= 10) (gdy a==true)
e:   { x=x+1;         (gdy c==true i a==true)
      y=y-1;
    }
d:           (gdy c==false i a==true)
f:           (koniec c i a==true)
    }
b:           (gdy a==false)
g:           (koniec a, koniec programu)
```



$$V(G)=e-n+2*p=3$$

$$V_{LI}(G)=e-n+p+1=8-7+2=3$$

$$SLOC=7$$

$$S/C=9$$

Zgodnie z aksjomatem 7 (wykład 7) pętla zagnieżdżona powinna mieć złożoność różną od programu z dwiema sekwencyjnie wykonywanymi pętlami. Jednak zarówno SLOC, $V(G)$, $V_{LI}(G)$ są identyczne w obu rozwiązaniach, natomiast różne są wartości metryki S/C. Wg metryki S/C bardziej złożony jest program z zagnieżdżoną pętlą.

Metryki P_1, P_2 (Henderson-Sellers)

$$P_1 = a_1 * V(G) + a_2 * N^* + a_3 * (v_e - 1)$$

- wersja addytywna

$$P_2 = [a_1 * V(G) + a_2 * N^*] * a_3 * v_e$$

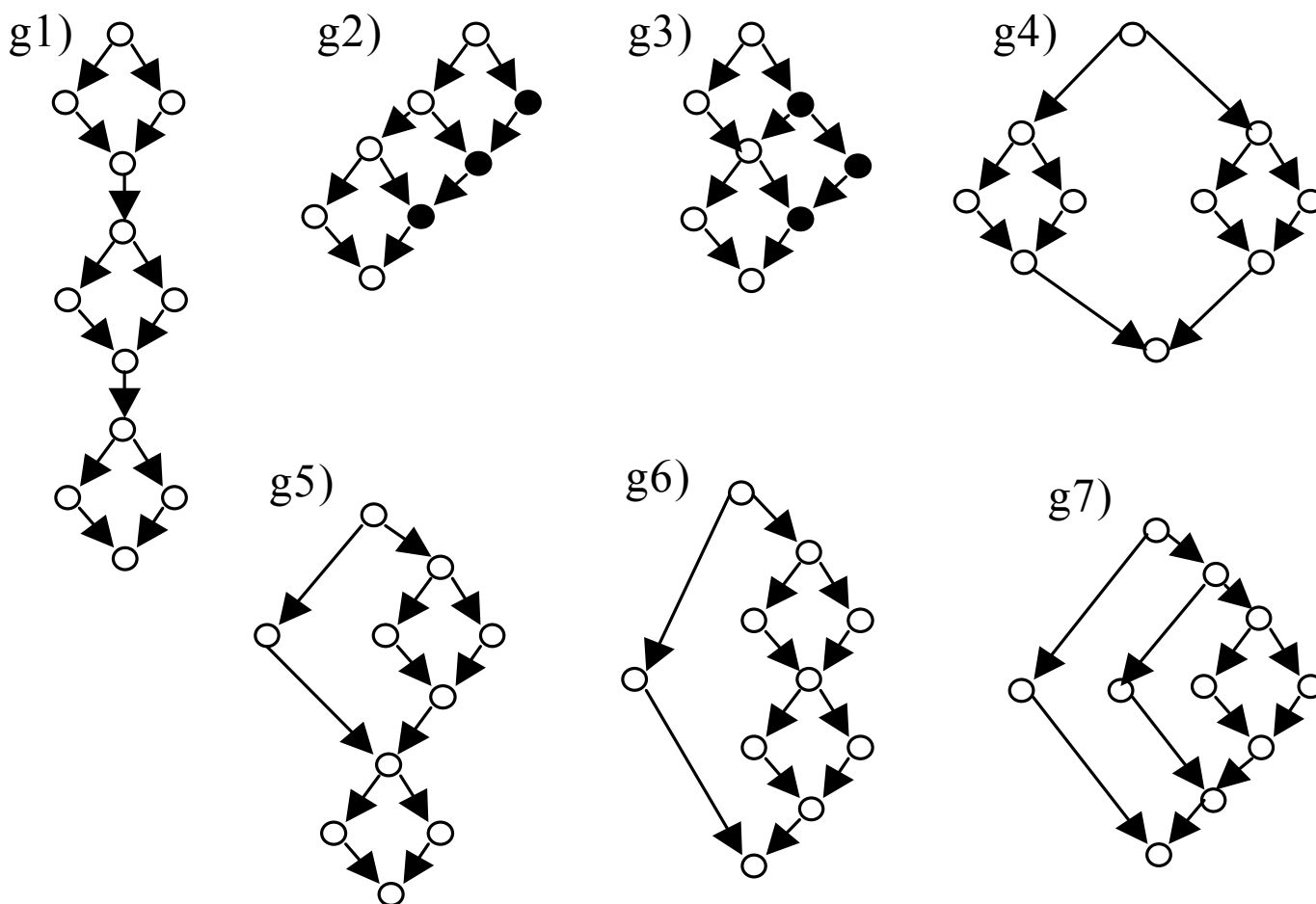
- wersja multiplikatywna

gdzie: a_1, a_2, a_3 są współczynnikami, które dla uproszczenia mają wartość 1
 $N^* = \sum_i P(i)$, gdzie $p(i)$ jest długością ścieżki zagnieżdżenia i-tego predykatu w instrukcjach decyzyjnych

v_e – liczba cykloematyczna wyznaczana dla programów niestrukturalnych

$v_e = d+1$, gdzie d jest liczbą niestrukturalnych rozgałęzień w programie (•)

Przykład 8.3



Rys. 2. Przykłady grafów przepływu sterowania

Tabela z wartościami obliczonych metryk złożoności przepływu sterowania

	g1	g2	g3	g4	g5	g6	g7
V(G)	4	4	4	4	4	4	4
v_e	1	4	4	1	1	1	1
P1	4	7	7	6	5	6	7
P2	4	16	16	6	5	6	7

Metryki P_1, P_2 najlepiej odzwierciedlają złożoność przepływu sterowania w przedstawionych przykładowych grafach struktury logicznej przykładowych programów g1-g7. Małą złożoność mają grafy g1 i g5.

Metryki struktur danych

Liczba zmiennych, które istnieją we wskazanym fragmencie lub całym programie.

Metryki spójności

Metryki spójności określają stopień powiązania między funkcjami modułu i typem przekazywanych danych. Im więcej funkcji przetwarza te same dane (np. zbiór funkcji przetwarzający tablicę lub strukturę wiążaną), tym wyższa spójność modułu.

Typy spójności wewnątrz modułu:

- przypadkowa – oznacza łączenie modułów, które podczas projektowania nie były traktowane jako całość
- logiczne – operacje tworzące logiczną całość np. moduł *stdio* dla operacji we/wy
- czasowe – inicjowanie np. grafiki, a dopiero potem rysowanie figur
- sekwencyjne – wykonywanie ustalonych sekwencji instrukcji np. operacje na plikach fizycznych wymagają zachowania kolejności działań: otwieranie pliku wraz z kojarzeniem z plikiem fizycznym, następnie przetwarzanie (zapis, odczyt) pliku i na koniec zamykanie pliku,
- funkcjonalne – najsilniejszy i najbardziej zalecany stopień osiągniętej spójności. Dotyczy on liczby powiązanych ze sobą funkcji przetwarzających te same dane.

Przy wyznaczaniu spójności funkcjonalnej rozważa się zależności między funkcjami i danymi, dedykowanymi tym funkcjom. W przypadku, gdy dowolna funkcja w rozważanym module wywołuje inną funkcję z tego modułu, należy przypisać tej funkcji wszystkie dane, z którymi jest ona związana.

Wyznaczenie spójności funkcjonalnej oparto się na *grafie dwudzielnym*.

Grafem dwudzielnym $G(V1, V2, E)$ nazywamy taki graf, w którym wierzchołki można podzielić na dwa podzbiory $V1$ i $V2$ w taki sposób, że każda krawędź ze zbioru E grafu G łączy dowolny wierzchołek zbioru $V1$ z dowolnym wierzchołkiem zbioru $V2$.

Zbiór E krawędzi jest zbiorem zbiorów dwuelementowych, z których każdy zawiera wierzchołek z $V1$ i wierzchołek z $V2$, wyznaczające krawędź.

Pełnym grafem dwudzielnym nazywamy taki graf dwudzielny, w którym wszystkie wierzchołki z $V1$ są połączone krawędziami ze zbioru E ze wszystkimi wierzchołkami $V2$.

Składowa spójna grafu niespójnego $G(V1, V2, E)$ jest jego podgrafem dwudzielnym $G'(V1', V2', E')$, takim że zbiór $V1'$ jego wierzchołków należy do zbioru wierzchołków $V1$, zbiór wierzchołków $V2'$ należy do zbioru $V2$, zbiór krawędzi E' należy do zbioru krawędzi E , jednak nie istnieje taka krawędź ze zbioru E' , która łączy dowolny wierzchołek z $V1'$ z dowolnym wierzchołkiem z $V2-V2'$ lub dowolny wierzchołek z $V1-V1'$ z dowolnym wierzchołkiem z $V2'$. Graf lub podgraf spójny zawiera tylko jedną składową spójności.

Przy wyznaczaniu spójności modułu wierzchołki ze zbioru V1 reprezentują funkcje modułu, wierzchołki ze zbioru V2 przetwarzane dane, natomiast krawędzie-związki między danymi i funkcjami.

Miary spójności modułu oznaczają ocenę spójności uzyskanego grafu dwudzielnego.

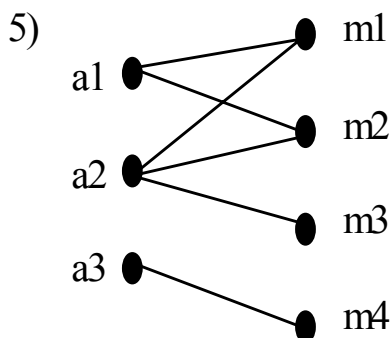
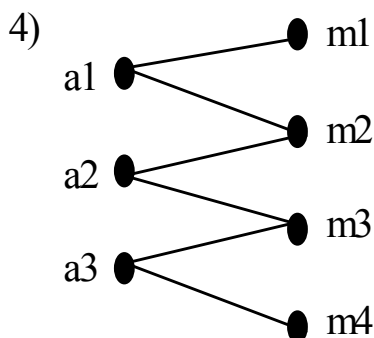
Przykład 8.4

Graf dwudzielny 4): $V1=\{a1,a2,a3\}$, $V2=\{m1,m2,m3,m4\}$,

$E=\{\{a1,m1\}, \{a1,m2\}, \{a2,m2\}, \{a2,m3\}, \{a3,m2\}, \{a3,m4\}\}$

Graf dwudzielny 5): $V1=\{a1,a2,a3\}$, $V2=\{m1,m2,m3,m4\}$,

$E=\{\{a1,m1\}, \{a1,m2\}, \{a2,m1\}, \{a2,m2\}, \{a2,m3\}, \{a3,m4\}\}$



Miara spójności LCOM

$$LCOM = \frac{(\frac{1}{a} \sum_{j=1}^a \mu(A_j)) - m}{1 - m},$$

gdzie m jest liczbą wierzchołków $V1$, a jest liczbą wierzchołków $V2$, natomiast wyrażenie $\mu(A)$ liczbą krawędzi grafu. Maksymalna wartość spójności oznacza wartość 0 metryki, co uzyskuje się przy grafie pełnym ($|V1|*|V2|$ krawędzi)

Metryka spójności LW

Przy wyznaczaniu spójności metryka LW zakłada dwa przypadki:

- (1) istnienie jednej składowej,
- (2) istnienie wielu składowych spójności.

Metryka *Spójność_M* jest zdefiniowana jako

$$Spójność_M = \frac{LW1, \text{ gdy graf funkcyjny jest spójny}}{LW2, \text{ gdy graf funkcyjny jest niespójny}}$$

$$LW1 = 1 + \frac{p - X - Y + 1}{X * Y - X - Y + 1}$$

$$LW2 = \frac{p - k}{(X - 1) * (Y - 1) - 1}$$

Przyjęto $X=|V1|$, $Y=|V2|$, p jest liczbą wszystkich krawędzi, k liczbą składowych spójności. Dla grafów zawierających jeden typ danej ($Y=1$) i jedną funkcję ($X=1$), lub $X=1$ i $Y>1$ lub $Y=1$ i $X>1$, przy założeniu spójności grafu ($k=1$) przyjęto $LW1=1$, natomiast dla grafów niespójnych przyjęto $LW2=0$ dla $Y=2$, $X=2$, $k=2$.

Metoda wyznaczania spójności modułu

Do reprezentowania grafu dwudzielnego wybrano *macierz przyległości* o wymiarach $|V1| \times |V2|$, zdefiniowaną jako $A=[a_{ij}]$, gdzie element macierzy a_{ij} jest równy 1, gdy istnieje połączenie między wierzchołkiem $v_{1i} \in V1$ i $v_{2j} \in V2$, oraz równe 0 w przeciwnym wypadku. Macierz przyległości reprezentuje *tabela przyległości*, w której kolumny odpowiadają $V2$, a wiersze $V1$.

1) *Metoda wyznaczenia tabeli przyległości*

- 1.1) W tabeli przyległości kolejnym wierszom przyporządkowano funkcje modułu, natomiast kolumnom przyporządkowano odpowiednio rozpatrywane dane.
- 1.2) Należy rozpatrzyć wszystkie funkcje, które nie wywołują innych funkcji z tego modułu. W przypadku związku, który występuje między funkcją i daną, należy na przecięciu wiersza z kolumną wpisać 1. W przypadku wywołania funkcji przez bieżącą funkcję, należy przepisać wszystkie jedynki do odpowiadających kolumn w wierszu przyporządkowanym rozpatrywanej funkcji.

2) *Metoda wyznaczenia składowych spójności grafu*

- 2.1) Ustal licznik składowych $k = 1$.
- 2.2) Wybierz kolejny wiersz i
 - 2.2.1) Przepisz wszystkie jedynki z pozostałych wierszy, które mają przynajmniej jedną jedynkę w odpowiadającej kolumnie wiersza i .
 - 2.2.2) Sprawdź, czy zmieniła się liczba jedynek w wierszu i . Jeśli tak, przejdź do kroku 2.2.1, w przeciwnym wypadku wykonaj następny krok
 - 2.2.3) Usuń wiersz i i wszystkie wiersze wykryte w punkcie 2.2.1
- 2.3) Sprawdź, czy pozostał jeszcze jakiś wiersz. Jeśli tak, wykonaj $k=k+1$, gdyż wyodrębniono kolejną składową spójności, i przejdź do kroku 2.2. W przeciwnym wypadku zakończ algorytm, gdyż wyodrębniono już wszystkie składowe spójności.

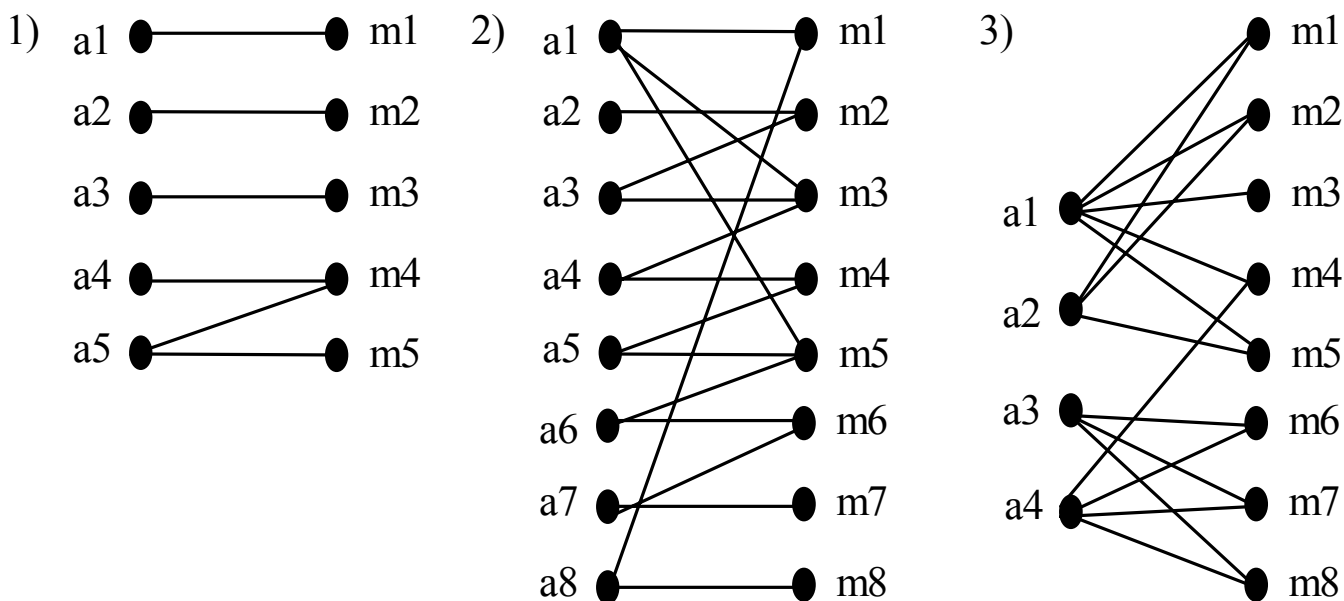
3) *Metoda obliczenia wartości metryk*

- 3.1) Należy wypełnić *tabelę przyległości* grafu dwudzielnego (metoda 1)
- 3.2) Wyznaczyć liczbę składowych spójności k (metoda 2)
- 3.3) Wyznaczyć liczbę jedynek p . Jest to liczba krawędzi grafu dwudzielnego
- 3.4) Należy podstawić do wyrażenia LCOM: $\mu(A)=p$, $m=|V1|$, $a=|V2|$ i obliczyć wartość metryki
- 3.5) Należy podstawić do wyrażenia LW1: p , $X=|V1|$, $Y=|V2|$ i obliczyć wartość metryki, jeśli $k=1$, w przeciwnym wypadku należy wyznaczyć LW2

Zakres wartości dla poszczególnych metryk

Zakres spójności	LW1	LW2	LCOM
Max	2 ($p=X*Y$)	1 ($p=(X-1)*(Y-1)+1, k=2$)	0 ($\mu(A)=p=X*Y$)
Min	1 ($p=X+Y-1$)	0 ($p=k=X=Y$)	1 ($\mu(A)=p=X=Y$)

Przykład 8.5



Rys. 3. Grafy dwudzielne o różnej spójności

Lp	X	Y	p	k	LW1	LW2	LCOM
1 (rys.3)	5	5	6	4	-	0.1333	0.95
2 (rys.3)	8	8	16	1	1.0204	-	0.8571
3 (rys.3)	4	8	15	1	1.1905	-	0.7083
4 (str.6)	3	4	6	1	1	-	0.75
5 (str.6)	3	4	6	2	-	0.8	0.75

Metryki grafów spójnych $LW1$ oraz niespójnych $LW2$ oraz $LCOM$ - przykłady danych

Przykład 8.6

```
#include <alloc.h> //wyznaczanie spójności modułu mkolejka.cpp
#include "mkolejka.h"
//typ Quadruple
void BUILD(Vector vector, int end, int start, int size,
           Quadruple &quadruple)
{ quadruple.vector=vector;
  quadruple.end=end;
  quadruple.start=start;
  quadruple.size=size; }
inline Vector& FIRST(Quadruple& quadruple)
{ return quadruple.vector; }
inline int& SECOND(Quadruple& quadruple)
{ return quadruple.end; }
inline int& THIRD(Quadruple& quadruple)
{ return quadruple.start; }
inline int& FOURTH(Quadruple& quadruple)
{ return quadruple.size; }

//typ kolejka pomocnicza-dynamiczna tablica wskaźników
inline int IS_EMPTY_V(Vector vector) { return vector==NULL; }
inline Vector EMPTY_V(int size) { return new PItem[size]; }
Vector ASSIGN(Vector vector, int& index, PItem pitem)
{ vector[index++]=pitem;
  return vector; }
PItem READ(Vector vector, int index)
{ return vector[index]; }

//kolejka
void EMPTY_F(Quadruple &quadruple)
{ BUILD(EMPTY_V(MAX), 0, MAX, MAX, quadruple); }
int IS_EMPTY_F(Quadruple quadruple)
{ return FOURTH(quadruple)==THIRD(quadruple)
      &&SECOND(quadruple)==0; }
int ADD_F(Quadruple &quadruple, Item item)
{ PItem pitem;
  //int empty=IS_EMPTY_F(quadruple);
  //if (empty||!(SECOND(quadruple)==(THIRD(quadruple)%FOURTH(quadruple))))
  if (!(SECOND(quadruple)==THIRD(quadruple)))
  { pitem=NEW_ITEM(item);
    if (pitem==NULL) return 0;
    if (THIRD(quadruple)==FOURTH(quadruple))
      THIRD(quadruple)%=FOURTH(quadruple);
    ASSIGN(FIRST(quadruple), SECOND(quadruple), pitem);
    SECOND(quadruple)%=FOURTH(quadruple);
    return 1; }
  return 2; }
```

```

int REMOVE_F(Quadruple &quadruple)
{ if (IS_EMPTY_F(quadruple)) return 0;
  THIRD(quadruple)%=FOURTH(quadruple);
  delete READ(FIRST(quadruple),THIRD(quadruple)++);
  if (SECOND(quadruple)==(THIRD(quadruple)%FOURTH(quadruple)))
    { SECOND(quadruple)=0;
      THIRD(quadruple)=MAX;}
  return 1;}

int FRONT_F(Quadruple quadruple,Item& item)
{ if (IS_EMPTY_F(quadruple)) return 0;
  THIRD(quadruple)%=FOURTH(quadruple);
  item=*READ(FIRST(quadruple),THIRD(quadruple));
  return 1;}

//funkcje pomocnicze
PItem NEW_ITEM(Item n_item)
{ PItem pitem;
  pitem = new Item;
  if (pitem!=NULL) *pitem=n_item;
  return pitem;}

int FOR_ONE(Quadruple& quadruple, Pfunction function)
{ if (IS_EMPTY_F(quadruple)) return 0;
  int a=THIRD(quadruple)%FOURTH(quadruple);
  function(*READ(FIRST(quadruple),a));
  return 1;}

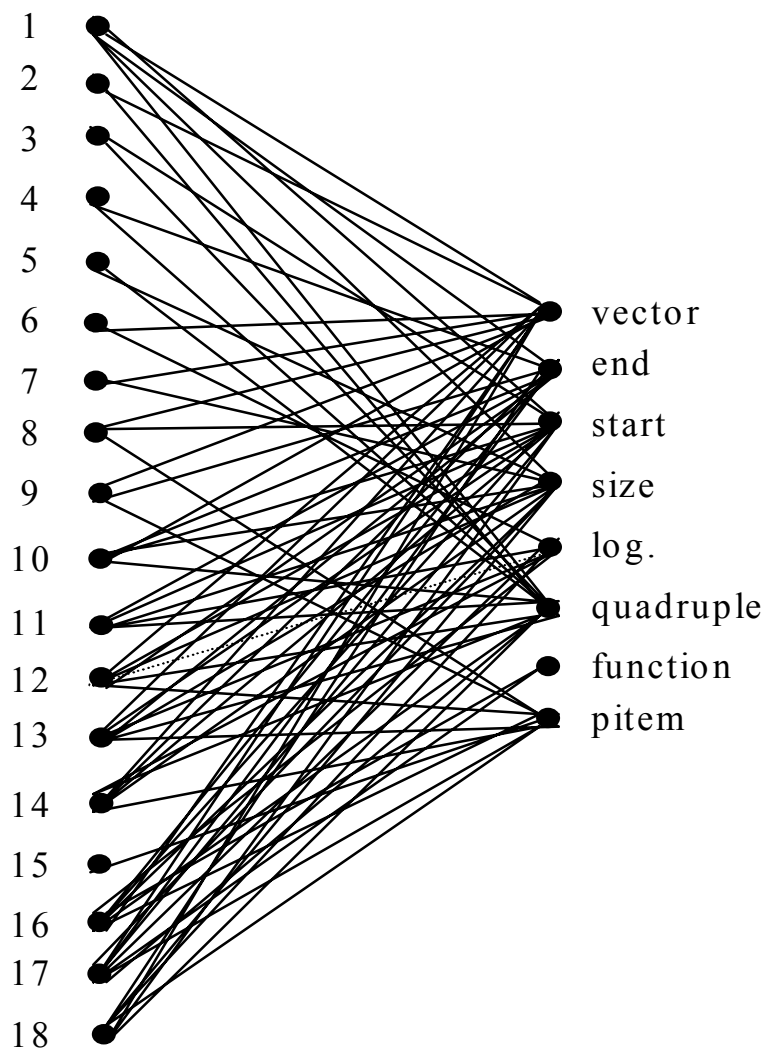
int FOR_EACH(Quadruple& quadruple, Pfunction function)
{ if (IS_EMPTY_F(quadruple)) return 0;
  int a=THIRD(quadruple);
  int b=SECOND(quadruple);
  int c=FOURTH(quadruple);
  if (b==0)b=c;
  do
    { a%=c;function(*READ(FIRST(quadruple),a));a++;
    }while(a!=b);
  return 1;}

void FREE(Quadruple& quadruple)
{ if (!IS_EMPTY_F(quadruple))
  { int a=(THIRD(quadruple)%FOURTH(quadruple));
    int b=SECOND(quadruple);
    int c=FOURTH(quadruple);
    if (b==0) b=c;
    do
      {a%=c;delete READ(FIRST(quadruple),a); a++;
      }while(a!=b);}
  delete[] FIRST(quadruple);}

```

L.p		vector	start	end	size	log.	quadruple	function	pitem
1	BUILD	1	1	1	1		1		
2	FIRST	1					1		
3	SECOND			1			1		
4	THIRD		1				1		
5	FORTH				1		1		
6	IS_EMPTY_V	1				1			
7	EMPTY_V	1			1				
8	ASSIGN	1		1					1
9	READ	1	1						1
10	EMPTY_F	1	1	1	1		1		
11	IS_EMPTY_F		1	1	1	1	1		
12	ADD_F	1	1	1	1	(1)	1		1
13	REMOVE_F	1	1	1	1	1	1		1
14	FRONT_F	1	1	1	1	1	1		1
15	NEW_ITEM								1
16	FOR_ONE	1	1	1	1	1	1	1	1
17	FOR_EACH	1	1	1	1	1	1	1	1
18	FREE	1	1	1	1	1	1		1

Tabela przyległości do wyznaczenia spójności modułu mkolejka.cpp



LW1=1.437 (1.4454)

LCOM=0.4926 (0.4853)

dla p=77 (78), X=18, Y=8