O relacjach i algorytmach

Zenon Gniazdowski

Warszawska Wyższa Szkoła Informatyki

zgniazdowski@wwsi.edu.pl

Streszczenie
Podczas wykładu zostaną omówione relacje dwuczłonowe, a także sposoby ich reprezentacji w postaci macierzy lub grafu. Grafy zostaną użyte do pokazania relacji w postaci przemawiającej do wyobraźni. Tymczasem postać macierzowa umożliwi konstrukcję algorytmów do badania własności relacji. Zostaną także pokazane przykłady relacji wraz z przedstawieniem ich własności i określeniem ich typów.

Spis treści

1 WSTĘP

Relacja jest podstawowym pojęciem matematycznym [2, 3], także użytecznym w informatyce. Na przykład, w językach programowania występują operatory relacji =, ≠, <, ≤, >, ≥. Od tego, czy pewna relacja zachodzi, jest uzależnione wykonywanie instrukcji warunkowej.

Jako inny przykład można podać relacyjne bazy danych, gdzie dane są grupowane w relacje, które są reprezentowane, jako wiersze w tabelach. Relacja, o której tu mowa, jest uogólnieniem relacji dwuczłonowej, która zostanie przedstawiona w dalszej części wykładu.

Z kolei w eksploracji danych mówi się o danych otrzymywanych w wyniku pomiarów. Z pomiarem związana jest skala pomiarowa. Skale pomiarowe są definiowane przez różne typy relacji. Również w eksploracji danych wykorzystuje się teorię zbiorów przybliżonych (ang. *rough sets*), a w niej relację nierozróżnialności.

Relację dwuczłonową można przedstawić w postaci macierzy kwadratowej, która jednocześnie jest macierzą sąsiedztwa grafu zdefiniowanego przez tę relację. Macierzowa reprezentacja relacji umożliwia tworzenie algorytmów do badania własności relacji. Takie podejście ma istotne znaczenie z punktu widzenia dydaktyki informatyki. Informatyk powinien umieć skojarzyć pojęcia teoretyczne z praktycznym ich zastosowaniem. W szczególności, powinien umieć przedstawiać modele matematyczne w postaci zalgorytmizowanej.

W niniejszym wykładzie zostaną omówione relacje dwuczłonowe, ich własności i typy, a także algorytmy służące do ich badania. Ponieważ kluczem do zrozumienia niektórych algorytmów jest znajomość pewnych operacji na macierzach zostaną one krótko przedstawione w dodatku na końcu wykładu.

2 ILOCZYN KARTEZJAŃSKI ZBIORÓW

Rozważa się dwa zbiory X i Y. Zbiór wszystkich uporządkowanych par (x, y) elementów należących odpowiednio do tych zbiorów, nazywa się **iloczynem (produktem) kartezjańskim** zbiorów X i Y. Iloczyn kartezjański oznacza się jako $X{\times}Y$. Można to zapisać w następujący sposób:

$$X{\times}Y = \{(x, y) : x \in X \text{ i } y \in Y\}\,. \tag{1}$$

Jeżeli dodatkowo zachodzi równość $X = Y$, to zamiast $X{\times}Y$ można napisać X^2.

Niech za przykład posłuży iloczyn kartezjański dwóch zbiorów X i Y, z których pierwszy zawiera cztery cyfry: $X = \{1, 2, 3, 4\}$, zaś drugi zawiera trzy litery: $Y = \{a, b, c\}$. Iloczynem kartezjańskim jest zbiór wszystkich par (*cyfra*, *litera*):

$$X{\times}Y = \{1, 2, 3, 4\} \times \{a, b, c\} = \begin{Bmatrix} (1, a) & (1, b) & (1, c) \\ (2, a) & (2, b) & (2, c) \\ (3, a) & (3, b) & (3, c) \\ (4, a) & (4, b) & (4, c) \end{Bmatrix}. \tag{2}$$

Przykład ten można także zinterpretować graficznie, jak na rysunku 1. Jeżeli w tabeli wiersze oznaczymy elementami jednego zbioru, a kolumny – elementami drugiego zbioru, to punkt na przecięciu odpowiedniego wiersza i kolumny reprezentuje parę (*cyfra*, *litera*).

	a	b	c
1	■	■	■
2	■	■	■
3	■	■	■
4	■	■	■

Rysunek 1.
Graficzna interpretacja iloczynu kartezjańskiego

Innym przykładem iloczynu kartezjańskiego jest zbiór punktów na płaszczyźnie *OXY*, oznaczany jako R^2. Pojęcie iloczynu kartezjańskiego można rozszerzyć na większą liczbę wymiarów. Dla przykładu, iloczyn $R \times R \times R = R^3$ oznacza zbiór punktów w przestrzeni trójwymiarowej.

Kolejnym przykładem iloczynu kartezjańskiego jest zbiór indeksów elementów macierzy. Macierz jest skończonym zbiorem elementów, zapisywanym w postaci prostokątnej tablicy o *m* wierszach i *n* kolumnach:

$$\begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix} . \qquad (3)$$

Adres elementu w macierzy składa się z dwóch liczb, z których pierwsza wskazuje na numer wiersza, a druga na numer kolumny, w których ten element się znajduje. Na przykład $a_{24} = 1$ oznacza, że element znajdujący się w drugim wierszu i czwartej kolumnie macierzy jest równy 1. Dokładniej rzecz ujmując, macierz można zdefiniować jako funkcję określoną na iloczynie kartezjańskim zbiorów $\{1, 2, \dots, m\}$ oraz $\{1, 2, \dots, n\}$:

$$f : \{1, 2, \dots, m\} \times \{1, 2, \dots, n\} \rightarrow R . \qquad (4)$$

3 RELACJA DWUCZŁONOWA

Dla danych zbiorów *X* i *Y* relacją dwuczłonową na iloczynie kartezjańskim $X \times Y$ jest dowolny podzbiór tego iloczynu. Przykładem relacji może być przydział przedmiotów, których uczą nauczyciele w pewnej szkole artystycznej. W przykładzie tym można wyodrębnić dwa zbiory. Pierwszy z nich, to zbiór nazwisk nauczycieli: *X* = {*Kowalski*, *Kwiatkowski*, *Malinowski*, *Nowak*, *Wiśniewski*}. W zbiorze drugim znajdą się przedmioty, które są w szkole nauczane: *Y* = {*gimnastyka*, *rytmika*, *śpiew*, *taniec*}. Oto przykładowy przydział nauczycieli do przedmiotów: {(*Nowak*, *rytmika*), (*Nowak*, *śpiew*), (*Nowak*, *taniec*), (*Kwiatkowski*, *gimnastyka*), (*Malinowski*, *rytmika*), (*Malinowski*, *taniec*), (*Kowalski*, *taniec*), (*Wiśniewski*, *gimnastyka*), (*Wiśniewski*, *śpiew*)}. Przydział ten jest przykładem relacji. Widać, że jest to tylko pewien podzbiór iloczynu kartezjańskiego $X \times Y$. Iloczyn kartezjański zawierałby dwadzieścia par (*nazwisko*, *przedmiot*), tymczasem przedstawiona relacja zawiera tylko dziewięć par.

3.1 REPREZENTACJA RELACJI

Rozważa się dwa zbiory: zbiór $X = \{x_1, x_2, \dots, x_m\}$ składający się z *m* elementów oraz zbiór $Y = \{y_1, y_2, \dots, y_n\}$ zawierający *n* elementów. Niech na ich iloczynie kartezjańskim $X \times Y$ będzie określona pewna relacja ρ. Relacja ta może być reprezentowana w postaci macierzy. Wiersze macierzy odpowiadają kolejnym elementom zbioru *X*, zaś kolumny – elementom zbioru *Y*. Macierzą relacji ρ jest zerojedynkowa macierz $[R_{ij}]$, zawierająca *m* wierszy i *n* kolumn. Jej elementy określone są następującym wzorem:

$$R_{ij} = \begin{Bmatrix} 1, & gdy & x_i \rho y_j \\ 0, & gdy & \neg(x_i \rho y_j) \end{Bmatrix} . \qquad (5)$$

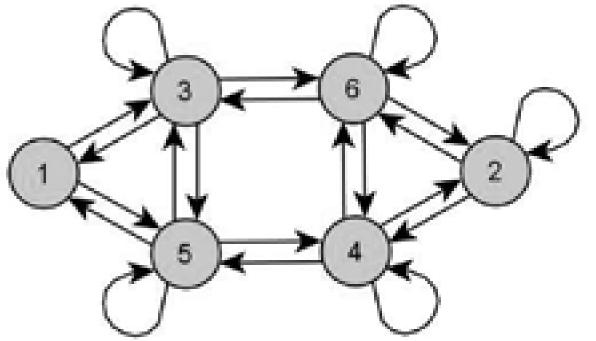
gdzie $i = 1, 2, \dots, m$, $j = 1, 2, \dots, n$. W powyższej formule wyrażenie $x_i \rho y_j$ czyta się: element x_i jest w relacji z elementem y_j, zaś wyrażenie $\neg(x_i \rho y_j)$ czyta się: nie jest prawdą, że element x_i jest w relacji z elementem y_j.

Jeżeli $X=Y$, to relacją w zbiorze *X* jest pewien podzbiór iloczynu kartezjańskiego $X \times X$. W dalszej części wykładu będą rozważane wyłącznie relacje w *n*-elementowym zbiorze *X*, reprezentowane przez kwadratową macierz o rozmiarze $n \times n$.

Przykład 1: Jako pierwszy przykład, zostanie pokazana relacja ρ określona na sześcioelementowym zbiorze $X = \{1, 2, 3, 4, 5, 6\}$. Dla dowolnych dwóch elementów $x, y \in X$, relacja jest zdefiniowana w następujący sposób: $x \rho y \Leftrightarrow x + y$ jest *liczbą złożoną*, a zatem element *x* jest w relacji z elementem *y* wtedy i tylko wtedy, gdy ich suma $x + y$ jest liczbą złożoną. Macierz tej relacji (ozn. *R*) ma następującą postać:

$$R = \begin{bmatrix} 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 \end{bmatrix} . \qquad (6)$$

Jeżeli elementy zbioru potraktować jako węzły grafu, to zdarzenie, iż element *i*-ty jest w relacji z elementem *j*-tym, można na grafie przedstawić przy pomocy łuku skierowanego od węzła *i* do węzła *j*. Otrzymany graf jest grafem skierowanym, zaś macierz relacji jest jednocześnie macierzą sąsiedztwa grafu. Na rysunku 2 jest przedstawiony graf relacji opisanej macierzą (6).



Rysunek 2.
Graf relacji opisanej macierzą (6)

4 WŁASNOŚCI RELACJI

Relacja dwuczłonowa może mieć następujące własności: może ona być zwrotna, przeciwzwrotna, symetryczna, antysymetryczna, przechodnia, spójna. Własności relacji ujawnią się w macierzy lub w grafie. Korzystając z reprezentacji macierzowej, można algorytmizować sprawdzanie, czy relacja ma daną własność. Reprezentacja grafowa relacji wpływa na wyobraźnię i ułatwia proces tworzenia algorytmu.

Do tworzenia algorytmów zostanie użyty język C++, w którym całkowite wartości 1 albo 0 można traktować jako logiczne wartości *prawda* albo *fałsz*. Wobec tego, dla zerojedynkowych elementów macierzy relacji mogą być stosowane operacje arytmetyczne lub operacje logiczne. Kryterium wyboru będzie prostota i zwięzłość algorytmu.

W przedstawionych algorytmach pojawi się statyczna tablica `R[SIZE][SIZE]`. Stała `SIZE` oznacza maksymalną liczność zbioru, na którym jest zdefiniowana relacja. Kwadrat tej wielkości określa złożoność pamięciową przedstawianych algorytmów. Z drugiej strony, złożoność czasowa będzie funkcją liczby naturalnej n, będącej aktualną licznością zbioru, na którym jest definiowana relacja. Przy czym aktualny rozmiar zbioru jest nie większy niż rozmiar maksymalny: $n \leq$ `SIZE`.

W algorytmach zostanie w sposób milczący wykorzystana jeszcze jedna własność języka C++. W opisie matematycznym numery wierszy lub kolumn macierzy są indeksowane liczbami od 1 do n. Tymczasem w języku C++ n wierszy i n kolumn tablicy dwuwymiarowej są indeksowane liczbami od 0 do $n - 1$.

4.1 ZWROTNOŚĆ

Relacja ρ określona w zbiorze X jest zwrotna, gdy dla każdego elementu $x \in X$ element ten pozostaje w relacji z samym sobą. Symbolicznie można to przedstawić w następujący sposób:

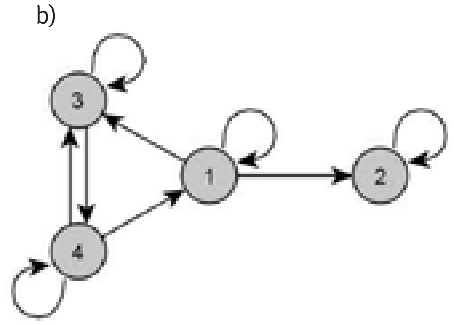
$$\forall x \in X : x \rho x. \qquad (7)$$

W zapisie macierzowym zwrotność przejawia się tym, że wszystkie elementy znajdujące się na przekątnej macierzy R są równe 1. Oznacza to, że w każdym węźle grafu relacji znajduje się pętla. Na rysunku 3 przedstawiono przykład macierzy relacji zwrotnej oraz odpowiadający mu graf.

a)

$$R = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 \end{bmatrix}$$

b)



Rysunek 3.
Przykład relacji zwrotnej: a) macierz relacji, b) graf relacji

Algorytm badania zwrotności powinien sprawdzać, czy na przekątnej macierzy znajdują się same jedynki. Jeżeli pojawi się co najmniej jedno zero, to relacja nie jest relacją zwrotną. Algorytm badania zwrotności zapisany w postaci funkcji w języku C++ ma następującą postać:

```
int zwrotna(int R[SIZE][SIZE],int n)
{  int i;
   for (i=0; i<n; i++)
```



```
      if (R[i][i] == 0) return 0;
   return 1;
}
```

Dominującą operacją w algorytmie jest poszukiwanie zer na przekątnej macierzy relacji. Gdy relacja jest zwrotna, liczba przeszukiwań będzie równa aktualnej liczności zbioru, w którym jest definiowana relacja. Liczba ta wynosi n i jest górnym oszacowaniem złożoności obliczeniowej algorytmu.

4.2 PRZECIWZWROTNOŚĆ

Relacja ρ określona w zbiorze X jest przeciwzwrotna, gdy żaden element $x \in X$ nie jest w relacji z samym sobą. Symbolicznie przeciwzwrotność można wyrazić w następujący sposób:

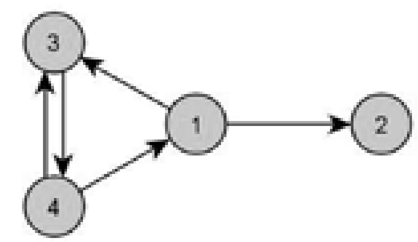
$$\forall x \in X : \neg(x \rho x). \qquad (8)$$

W zapisie macierzowym przeciwzwrotność przejawia się tym, że na głównej przekątnej macierzy relacji znajdują się same zera. Na grafie ujawni się to w ten sposób, że żaden jego węzeł nie będzie miał pętli. Przykład relacji przeciwzwrotnej jest pokazany na rysunku 4.

a)

$$R = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix}$$

b)



Rysunek 4.
Przykład relacji przeciwzwrotnej: a) macierz relacji, b) graf relacji

Algorytm badania przeciwzwrotności powinien poszukiwać jedynek na przekątnej macierzy. Jeżeli pojawi się co najmniej jedna jedynka, to relacja nie jest relacją przeciwzwrotną. Funkcja badająca przeciwzwrotność ma następującą postać:

```
int przeciwzwrotna(int R[SIZE][SIZE],int n)
{  int i;
   for (i=0; i<n; i++)
      if (R[i][i]) return 0;
   return 1;
}
```

W algorytmie badania przeciwzwrotności dominującą operacją jest poszukiwanie jedynki na przekątnej macierzy relacji. Gdy relacja jest przeciwzwrotna, liczba przeszukiwań będzie równa aktualnej liczności zbioru, na którym jest definiowana relacja. Liczba ta wynosi n i jest górnym oszacowaniem złożoności obliczeniowej.

4.3 SYMETRIA

Relacja ρ określona w zbiorze X jest symetryczna, gdy dla każdych dwóch elementów $x, y \in X$ z faktu, że x jest w relacji z elementem y – wynika, że y jest w relacji z elementem x. Można to przedstawić tak:

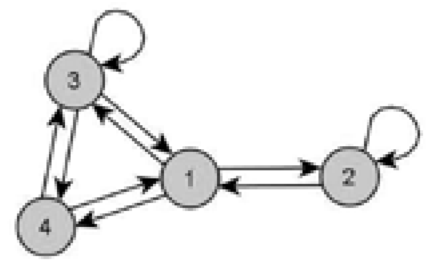
$$\forall x, y \in X : x \rho y \Rightarrow y \rho x. \qquad (9)$$

W zapisie macierzowym symetria relacji przejawia się w symetrii macierzy relacji. Macierz R jest symetryczna, gdy jest równa swojej transpozycji: $R = R^T$. Obraz grafu takiej relacji charakteryzuje się tym, że wszystkie łuki między dwoma różnymi węzłami grafu biegną w dwóch kierunkach. Rysunek 5 przedstawia przykład relacji symetrycznej.

a)

$$R = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix}$$

b)



Rysunek 5.
Przykład relacji symetrycznej: a) macierz relacji, b) graf relacji

Algorytm badania symetrii relacji polega na przechodzeniu wszystkich elementów R_{ij} znajdujących się ponad główną przekątną macierzy relacji i sprawdzaniu, czy są im równe elementy R_{ji} leżące pod przekątną. Jeżeli pojawi się sytuacja, że $R_{ij} \neq R_{ji}$, to relacja nie jest symetryczna. Algorytm ten ma następującą postać:

```
int symetryczna(int R[SIZE][SIZE],int n)
{  int i,j;
   for (i=0; i<n–1; i++)
      for (j=i+1; j<n; j++)
         if (R[i][j] != R[j][i]) return 0;
   return 1;
}
```

W algorytmie badania symetrii operacją dominującą jest porównywanie elementu leżącego nad przekątną macierzy z odpowiadającym mu elementem leżącym pod przekątną. Nad przekątną macierzy znajdują się wiersze od pierwszego do przedostatniego. Przy czym w wierszu pierwszym jest $n - 1$ elementów, w wierszu kolejnym o jeden mniej, a w ostatnim wierszu nad przekątną jest już tylko jeden element. W skrajnym przypadku, gdy relacja będzie symetryczna, wszystkie elementy nadprzekątniowe biorą udział w porównywaniu. Liczba porównań jest równa liczbie tych elementów, a więc jest równa sumie ciągu arytmetycznego składającego się z kolejnych liczb od 1 do $n - 1$. Suma ta jest równa $n(n - 1)/2$ i jest górnym oszacowaniem złożoności obliczeniowej algorytmu badania symetrii relacji.

4.4 ANTYSYMETRIA

Relacja ρ określona w zbiorze X jest antysymetryczna, jeżeli dla każdych dwóch elementów $x, y \in X$ z faktu, że x jest w relacji z elementem y i y jest w relacji z elementem x wynika, że elementy x i y są identyczne. Symbolicznie zapisujemy to w następujący sposób:

$$\forall x, y \in X : x \rho y \wedge y \rho x \Rightarrow x = y. \qquad (10)$$

W macierzy relacji antysymetrycznej każdemu elementowi $R_{ij} = 1$ spoza przekątnej towarzyszy element $R_{ji} = 0$. Jest to równoważne następującemu warunkowi:

$$\forall (i, j)_{i \neq j}\ R_{ij} \wedge R_{ji} = 0. \qquad (11)$$

Na grafie ujawni się to w ten sposób, że jeżeli między dwoma różnymi węzłami istnieje łuk, to między tymi węzłami nie ma łuku biegnącego w przeciwnym kierunku. Na rysunku 6 pokazano przykład relacji antysymetrycznej.

Algorytm badania antysymetrii polega na przechodzeniu przez wszystkie elementy macierzy znajdujące się ponad przekątną i sprawdzaniu warunku (11). Jeżeli ten warunek nie jest spełniony co najmniej jeden raz, to relacja nie jest antysymetryczna. Algorytm badania antysymetrii zapisany w języku C++ ma postać:

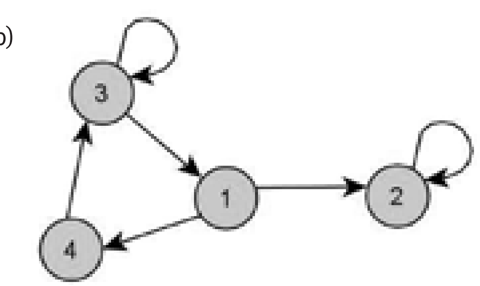
```
int antysymetryczna(int R[SIZE][SIZE],int n)
{  int i,j;
   for (i=0; i<n–1; i++)
      for (j=i+1; j<n; j++)
         if (R[i][j] && R[j][i]) return 0;
   return 1;
}
```

W algorytmie badania antysymetrii dominującą operacją jest sprawdzanie, czy jest równy zero iloczyn boolowski elementów R_{ij} leżących nad przekątną macierzy i odpowiadających im elementów R_{ji} leżących pod przekątną. Gdy relacja jest antysymetryczna, to podobnie jak w przypadku algorytmu badania symetrii, liczba sprawdzeń wynosi $n(n - 1)/2$. Liczba ta jest górnym oszacowaniem złożoności obliczeniowej algorytmu.

a)

$$R = \begin{bmatrix} 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

b)



Rysunek 6.
Przykład relacji antysymetrycznej: a) macierz relacji, b) graf relacji

4.5 PRZECIWSYMETRIA

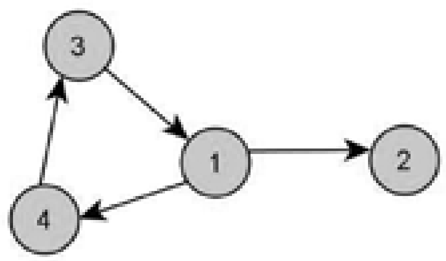
Relacja ρ określona w zbiorze X jest przeciwsymetryczna, gdy dla każdych dwóch elementów $x, y \in X$ z faktu, że x jest w relacji z y wynika, że y nie jest w relacji z x. Można to przedstawić w następujący sposób:

$$\forall x, y \in X : x \rho y \Rightarrow \neg(y \rho x). \qquad (12)$$

a)

$$R = \begin{bmatrix} 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

b)



Rysunek 7.
Przykład relacji przeciwsymetrycznej: a) macierz relacji, b) graf relacji

Przeciwsymetria relacji jest równoważna występowaniu dwóch innych wcześniej zdefiniowanych własności relacji: antysymetrii i przeciwzwrotności. Jeżeli relacja jest przeciwzwrotna i antysymetryczna, to jest przeciwsymetryczna. Oznacza to, że dla badania przeciwsymetrii nie ma potrzeby tworzenia specjalnego algorytmu. Relacja przeciwsymetryczna bywa także nazywana relacją asymetryczną [2]. Na rysunku 7 jest przedstawiony przykład relacji przeciwsymetrycznej.

4.6 PRZECHODNIOŚĆ

Relacja ρ określona w zbiorze *X* jest przechodnia, jeżeli dla dowolnych elementów $x, y, z \in X$ z faktu, że *x* jest w relacji z elementem *y* i *y* jest w relacji z elementem *z* wynika, że *x* jest w relacji z elementem *z*. Korzystając z symboliki matematycznej, przechodniość opisuje się następująco:

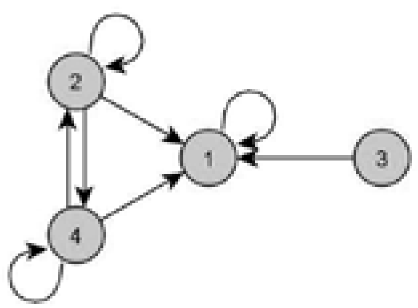
$$\forall x, y, z \in X : x\rho y \wedge y\rho z \Rightarrow x\rho z. \tag{13}$$

Graf relacji przechodniej charakteryzuje się tym, że jeżeli istnieje łuk od węzła *x* do węzła *y* i od węzła *y* do węzła *z*, to istnieje także łuk idący „na skróty” od węzła *x* do węzła *z*. Rysunek 8 przedstawia przykład relacji przechodniej.

a)

$$R = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 \end{bmatrix}$$

b)



Rysunek 8.
Przykład relacji przechodniej: a) macierz relacji, b) graf relacji

Warunek przechodniości można także sformułować inaczej: *jeśli pomiędzy dwoma różnymi węzłami grafu relacji istnieje ścieżka o długości dwóch łuków, to w grafie relacji przechodniej będzie między nimi istniała ścieżka o długości jednego łuku*. Powyższa obserwacja może pomóc w znalezieniu algorytmu badania przechodniości. Element R_{ij} macierzy sąsiedztwa grafu jest równy liczbie ścieżek o długości jednego łuku, biegnących od węzła *i* do węzła *j*. W macierzy sąsiedztwa grafu relacji będzie to co najwyżej jedna ścieżka. Kwadrat macierzy sąsiedztwa zlicza wszystkie ścieżki o długości dwóch łuków [3]. Warunek przechodniości relacji można teraz wyrazić następująco: *jeżeli* $(R^2)_{ij} > 0$, to $R_{ij} = 1$.

Macierz relacji jest macierzą zerojedynkową. Jej kwadrat – zliczający ścieżki o długości dwóch łuków – może zawierać zera lub liczby większe od zera. Ponieważ dla badania przechodniości nie jest istotne, ile jest ścieżek o długości dwóch łuków, ale czy takie ścieżki istnieją, to zamiast niezerowej liczby ścieżek o długości dwóch łuków w odpowiednich miejscach wynikowej macierzy R^2 wystarczy wstawić jedynkę informującą, że takie ścieżki istnieją, a następnie sprawdzać, czy jedynkom w wynikowej macierzy R^2 towarzyszą jedynki w macierzy *R*. Oznacza to, że operację podnoszenia macierzy *R* do kwadratu, można zastąpić operacją boolowskiego mnożenia macierzy. Wtedy warunek przechodniości relacji będzie spełniony, gdy:

$$R * R \leq R. \tag{14}$$

W wyrażeniu (14) operacja (*) oznacza mnożenie boolowskie macierzy [3]. W zwykłym mnożeniu macierzy znanym z algebry liniowej, wynikowy element mnożenia znajdujący się w *i*-tym wierszu i *j*-tej kolumnie jest liczony jako iloczyn skalarny *i*-tego wiersza i *j*-tej kolumny w macierzy *R*, co można zapisać w następującej postaci:

$$(R^2)_{ij} = \sum_{k=1}^{n} R_{ik} R_{kj} \;. \tag{15}$$

W mnożeniu boolowskim zamiast iloczynu i sumy w wyrażeniu (15) używa się odpowiednio iloczynu i sumy logicznej. Oznaczając wynik boolowskiego mnożenia macierzy jako *B*, pojedynczy element tej macierzy można wyrazić w następujący sposób: $B_{ij} = R_{i1} \wedge R_{1j} \vee R_{i2} \wedge R_{2j} \vee \ldots \vee R_{in} \wedge R_{nj}$. W zapisie zwartym wyrażenie ma postać:

$$B_{ij} = \bigcup_{k=1}^{n} R_{ik} \wedge R_{kj} \;. \tag{16}$$

Nierówność w wyrażeniu (14) oznacza, że w zerojedynkowej macierzy *B* jedynki mogą się znajdować co najwyżej w tych miejscach, w których znajdują się jedynki w macierzy *R*. Algorytm badania przechodniości składa się z dwóch faz. W fazie pierwszej wykonywana jest operacja (16), zaś w fazie drugiej sprawdzany jest warunek (14):

```
int przechodnia(int R[SIZE][SIZE], int n)
{  int B[SIZE][SIZE];
   int i,j,k;
//mnożenie boolowskie: B = R*R – wz. (16)
   for (i=0; i<n; i++)
      for (j=0; j<n; j++)
      {  B[i][j] = 0;
         for (k=0; k<n; k++)
            B[i][j] = B[i][j] || (R[i][k]&&R[k][j]);
      }
//sprawdzenie warunku (14)
   for (i=0; i<n; i++)
      for (j=0; j<n; j++)
         if (R[i][j] <B[i][j]) return 0;
   return 1;
}
```

Przedstawiony algorytm można dalej uprościć:

- Ponieważ znajdowanie kolejnych elementów macierzy *B* odbywa się niezależnie od innych elementów tej macierzy, nie ma potrzeby wyliczać całej macierzy przed przystąpieniem do sprawdzania warunku (14). Zamiast tego można wyliczać kolejny element macierzy *B* i sprawdzić, czy jest równy 1. Jeżeli tak jest, to także trzeba sprawdzić, czy odpowiadający mu element macierzy *R* jest równy 1. Jeżeli równość nie zachodzi, to można zakończyć działanie algorytmu, gdyż relacja nie jest relacją przechodnią.
- W wyrażeniu (16) można zrezygnować z sumowania. Jeżeli kolejny iloczyn wynosi zero, sumowanie nic nie zmieni. Jeżeli iloczyn jest równy jeden, należy przerwać pętlę liczącą iloczyny.
- Ponieważ cała macierz *B* nigdy nie jest potrzebna, a tylko jej kolejne elementy są lokalnie wyliczane, to można także zrezygnować z macierzy *B* na rzecz zmiennej lokalnej *B*, która przechowuje boolowską sumę iloczynów (16).

Po uproszczeniach, ostateczna wersja algorytmu ma postać:

```
int przechodnia(int n, int R[SIZE][SIZE])
{  int i,j,k;
   int B;
   for (i=0; i<n; i++) for (j=0; j<n; j++)
      {  B = 0;
         for (k=0; (!B) && (k<n); k++) B = R[i][k] && R[k][j];
         if (R[i][j] < B) return 0;
      }
   return 1;
}
```

W algorytmie badania przechodniości dominującą operacją jest obliczanie wartości co najwyżej n^2 wartości *B*. Dodatkowo, dla znalezienia pojedynczego elementu *B* potrzeba obliczyć maksymalnie *n* iloczynów. Stąd szacowana górna złożoność algorytmu wynosi n^3 iloczynów boolowskich.

4.7 SPÓJNOŚĆ

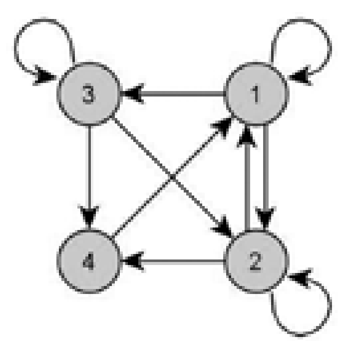
Relacja ρ określona w zbiorze *X* jest spójna, jeżeli dla dowolnych dwóch elementów $x, y \in X$ element *x* pozostaje w relacji z elementem *y* lub element *y* pozostaje w relacji z elementem *x*. Spójność wyrażona symbolicznie ma postać:

$$\forall x, y \in X: x\rho y \ \vee \ y\rho x \ \vee \ x = y. \qquad (17)$$

a)

$$R = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

b)



Rysunek 9.
Przykład relacji spójnej: a) macierz relacji, b) graf relacji

W zapisie macierzowym spójność przejawia się tym, że jeżeli poza przekątną w macierzy relacji zachodzi $R_{ij} = 0$, to odpowiednio $R_{ji} = 1$. Oznacza to, że dla relacji spójnej zawsze zachodzi następujący warunek:

$$\forall (i, j)_{i \neq j}\ R_{ij} \vee R_{ji} = 1. \qquad (18)$$

Graf relacji spójnej charakteryzuje się tym, że pomiędzy dwoma różnymi jego węzłami istnieje łuk co najmniej w jednym kierunku. Na rysunek 9 pokazano przykład relacji spójnej.

Algorytm badania spójności relacji polega na przebiegu przez wszystkie elementy macierzy znajdujące się ponad przekątną i sprawdzaniu, czy spełniony jest warunek (18):

```
int spojna(int R[SIZE][SIZE],int n)
{  int i,j;
   for (i=0; i<n-1; i++)
      for (j=i+1; j<n; j++)
         if (!(R[i][j] || R[j][i])) return 0;
   return 1;
}
```

W algorytmie badania spójności operacją dominującą jest sprawdzanie, czy logiczna suma elementów leżących nad przekątną macierzy (R_{ij}) i odpowiadających im elementów spod przekątnej (odpowiednio R_{ji}) jest równa 1. Gdy relacja jest spójna, liczba sprawdzeń – będąca jednocześnie górnym oszacowaniem złożoności algorytmu – wynosi (podobnie jak w przypadku algorytmów badania symetrii i antysymetrii) co najwyżej $n(n-1)/2$.

5 TYPY RELACJI

Własności relacji dwuczłonowej mogą konstytuować jej typ:

- Jeżeli relacja jest zwrotna, symetryczna i przechodnia, to jest relacją równoważności. Relacja równoważności dzieli zbiór na rozłączne klasy abstrakcji (klasy równoważności). Jako przykład można przedstawić następującą relację równoważności: dwa samochody na pobliskim parkingu są ze sobą w relacji, gdy mają ten sam kolor. Można sprawdzić, że tak zdefiniowana relacja jest zwrotna, symetryczna i przechodnia. Klas abstrakcji jest tyle, na ile różnych kolorów są pomalowane samochody: jedną z klas abstrakcji stanowią samochody czarne, inną czerwone, jeszcze inną srebrne itd.
- Kolejnym typem jest relacja częściowego porządku. Jest to relacja zwrotna, antysymetryczna i przechodnia. Zbiór z relacją częściowego porządku jest zbiorem częściowo uporządkowanym. W zbiorze częściowo uporządkowanym porządkowanie (sortowanie w sensie danej relacji) jest możliwe tylko w ramach pewnych podzbiorów. Przykładem relacji częściowego porządku jest relacja podzielności w zbiorze liczb całkowitych dodatnich nie większych niż 100.

  Formą przedstawienia relacji jest jej graf. Tymczasem skończony zbiór uporządkowany można także przedstawić w postaci tzw. diagramu Hassego [1, 2, 3]. Dla danej relacji porządkującej diagram ten powstaje przez zredukowanie grafu relacji. Najpierw należy zredukować wszystkie pętle charakteryzujące zwrotność. W drugiej kolejności należy odrzucić wszystkie łuki charakteryzujące przechodniość. Otrzymany po redukcji graf należy narysować tak, aby wszystkie jego strzałki były skierowane do góry. Po tej operacji należy zlikwidować wszystkie groty strzałek na końcach łuków [1, 3].

- Z kolei mocniejsze wymagania spełnia relacja porządku liniowego. Jeżeli relacja jest relacją porządku częściowego (a więc jest relacją zwrotną, antysymetryczną i przechodnią) i jednocześnie jest spójna, to jest to relacja porządku liniowego. Porządek liniowy jest własnością silniejszą niż porządek częściowy. Umożliwia on porządkowanie (sortowanie w sensie danej relacji) całego zbioru. Przykładem relacji porządku liniowego może być relacja ≥ w zbiorze liczb całkowitych dodatnich nie większych niż 100.

Podobnie jak w przypadku relacji porządku częściowego, relację porządku liniowego można również przedstawić w postaci diagramu Hassego, który tym razem ma postać linii.

Relacja równoważności umożliwia badanie, czy dwa elementy w zbiorze są równe (należą do tej samej klasy abstrakcji), czy też są różne (należą do różnych klas abstrakcji). W ramach tej relacji nie ma możliwości porządkowania elementów np. w sensie sprawdzania, czy jeden element poprzedza drugi element (w sensie rozważanej relacji). Tymczasem relacje porządku dają takie możliwości.

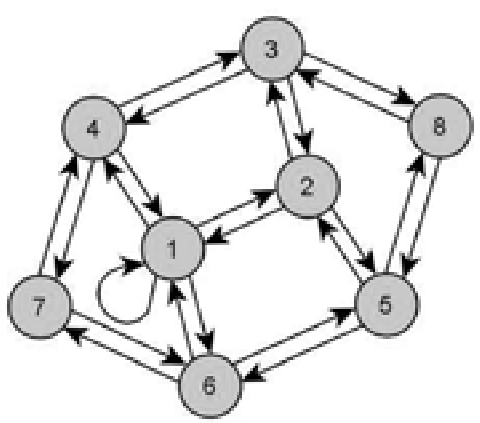
6 PRZYKŁADY RELACJI

Dla pokazania różnych własności i typów relacji zostaną przedstawione dalsze przykłady kilku relacji. W każdym z przykładów zostaną wskazane własności relacji oraz (o ile istnieje) jej typ.

Przykład 2: Dany jest zbiór $X = \{1, 2, 3, 4, 5, 6, 7, 8\}$. Relacja w zbiorze X jest zdefiniowana w następujący sposób: $x \rho y \Leftrightarrow x + y$ jest *liczbą pierwszą*, $x, y \in X$. Macierz tej relacji ma postać:

$$R = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} . \qquad (19)$$

Na rysunku 10 przedstawiono graf relacji (19). Relacja ta jest relacją symetryczną: dla macierzy zachodzi związek $R = R^T$, zaś w grafie pomiędzy węzłami biegną łuki w obie strony. Symetria jest jedyną własnością omawianej relacji, ponieważ ta relacja nie jest zwrotna (7), przeciwzwrotna (8), antysymetryczna (10), przechodnia (13) i spójna (17).



Rysunek 10.
Graf relacji opisanej macierzą (19)

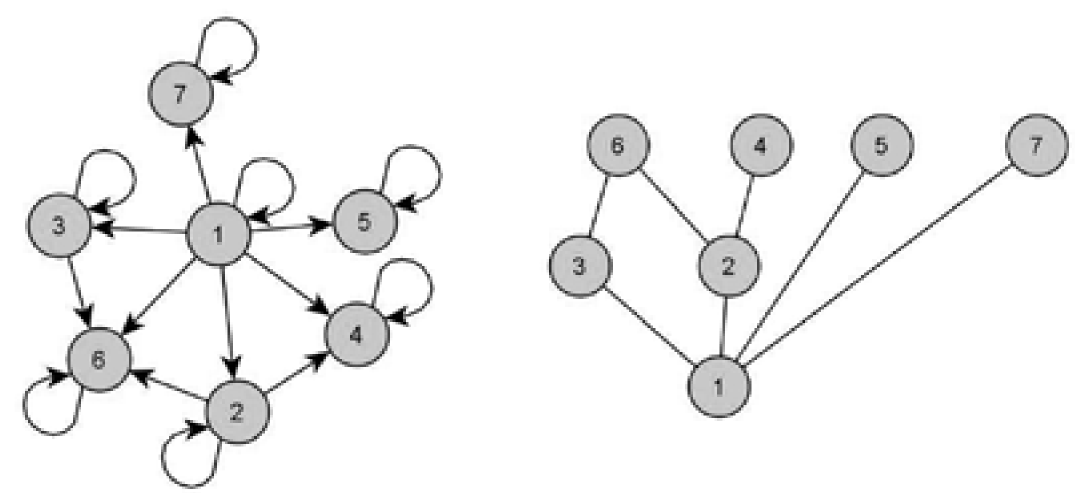
Przykład 3: Dla danego zbioru $X = \{1, 2, 3, 4, 5, 6, 7\}$, relacja jest zdefiniowana w następujący sposób: $x \rho y \Leftrightarrow$ *x jest dzielnikiem liczby y*, $x, y \in X$. Macierz relacji ma postać:

$$R = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} . \qquad (20)$$

Rysunek 11a przedstawia graf relacji (20). Pokazana w przykładzie relacja jest zwrotna – każda z liczb jest swoim własnym dzielnikiem. Jest to także relacja antysymetryczna – jeżeli liczba *x* jest dzielnikiem różnej od siebie liczby *y*, to liczba *y* nie jest dzielnikiem liczby *x*. W grafie będzie to widoczne w ten sposób, że pomiędzy węzłami biegną łuki tylko w jedną stronę. Relacja jest także przechodnia – zachodzi zależność (14). W ten sposób spełnione są warunki definiujące relację częściowego porządku.

Na rysunku 11b przedstawiono diagram Hassego dla relacji (20), otrzymany z redukcji grafu pokazanego na rysunku 11a. Odrzucenie łuków przedstawiających zwrotność nie niesie żadnych trudności. W przypadku redukcji linii charakteryzujących przechodniość usunięto łuki będące „drogami na skróty” od węzła 1 do węzła 4, a także od węzła 1 do węzła 6. Po usunięciu grotów strzałek pozostał diagram Hassego.

a) b)



Rysunek 11.
Reprezentacja relacji (20): a) graf relacji, b) diagram Hassego

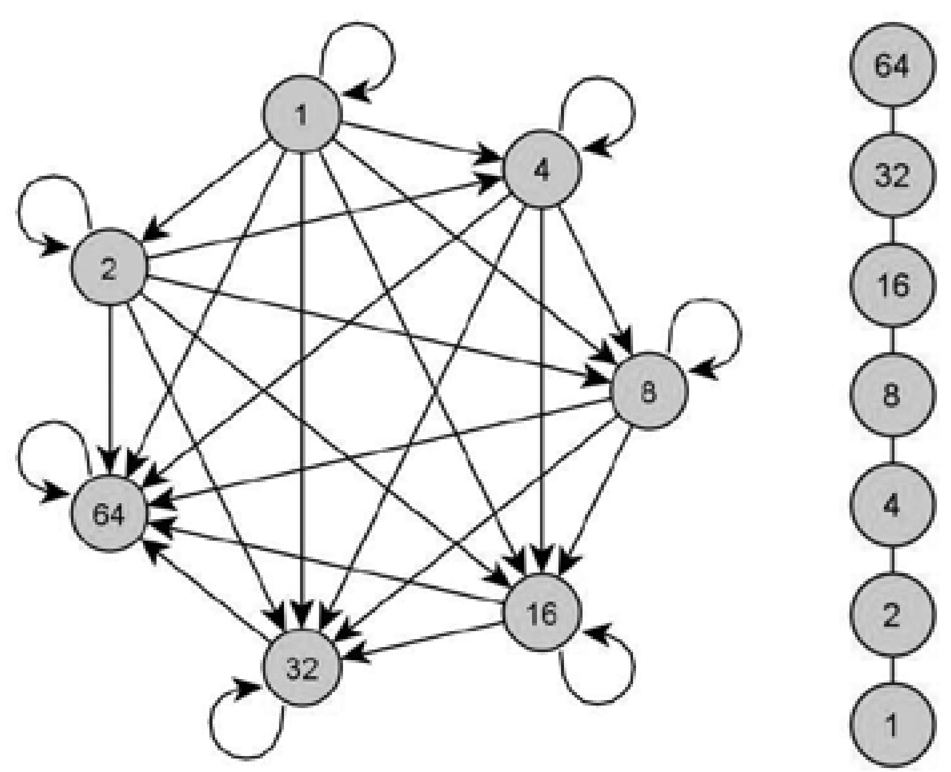
Tam, gdzie istnieje porządek, tam pewne elementy poprzedzają inne elementy. Gdy relacja jest relacją częściowego porządku, jedne elementy poprzedzają inne tylko w ramach podzbiorów. Oznacza to, że pewne podzbiory danego zbioru mogą być porządkowane (sortowane) w sensie tej relacji. Z diagramu Hassego można odczytać, które elementy poprzedzają się wzajemnie, a więc które podzbiory można porządkować w sensie omawianej relacji: {1,3,6}, {1,2,6}, {1,2,4}, {1,5} oraz {1,7}.

Przykład 4: Dla danego zbioru $X = \{1, 2, 4, 8, 16, 32, 64\}$, relacja jest zdefiniowana w następujący sposób: $x \rho y \Leftrightarrow$ *x jest dzielnikiem liczby y*, $x, y \in X$. W stosunku do poprzedniego przykładu różnica polega jedynie na rozpatrywaniu różnych zbiorów. Przyjmując, że kolejnym elementom zbioru X odpowiadają kolejne numery wierszy i kolumn w macierzy relacji, macierz ta ma następującą postać:

$$R = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}. \qquad (21)$$

Rysunek 12a pokazuje graf relacji (21). Przedstawiona relacja jest zwrotna, antysymetryczna, przechodnia i spójna. W przeciwieństwie do relacji z przykładu 3, która była relacją porządku częściowego, jest to relacja porządku liniowego. Oznacza to, że cały zbiór można uporządkować (posortować) w sensie danej relacji, co widać na diagramie Hassego (rys. 12b).

a) b)



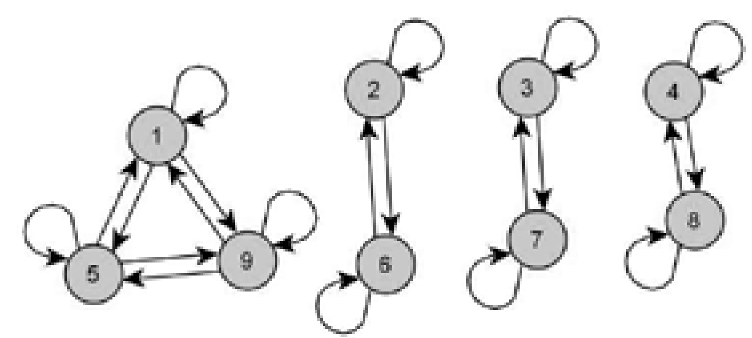
Rysunek 12.
Reprezentacja relacji (21), a) graf relacji, b) diagram Hassego

Przykład 5: Dla danego zbioru $X = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$, relacja jest zdefiniowana w następujący sposób: $x \rho y \Leftrightarrow (x \equiv y)$ *mod* 4, $x, y \in X$. Tak zdefiniowaną relację nazywa się niekiedy relacją kongruencji modulo 4.

Inaczej mówiąc, zmienne x i y są ze sobą w relacji, gdy mają jednakowe reszty z dzielenia przez 4. Macierz tej relacji wygląda tak:

$$R = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}. \qquad (22)$$

Na rysunku 13 przedstawiono graf relacji (22). Z analizy macierzy oraz grafu wynika, że jest to relacja zwrotna – każdy element jest w relacji z samym sobą, na przekątnej macierzy znajdują się same jedynki, a w grafie każdy węzeł ma pętlę. Jest to także relacja symetryczna – jeżeli liczba x ma resztę z dzielenia przez 4 identyczną z resztą z dzielenia liczby y przez 4, to także liczba y ma resztę z dzielenia przez 4 identyczną z resztą z dzielenia liczby x przez 4. W grafie pomiędzy różnymi węzłami łuki biegną parami i są skierowane w przeciwnych kierunkach. Relacja jest także przechodnia – zachodzi związek (14): w grafie każda ścieżka o długości dwóch łuków może być zastąpiona przez ścieżkę „na skróty" mającą długość jednego łuku. Jest to więc relacja równoważności. Relacja ta dzieli zbiór na cztery podzbiory będące klasami abstrakcji. Każda z klas zawiera elementy będące ze sobą w relacji, czyli takie, które mają identyczne reszty z dzielenia przez 4: {1,5,9}, {2,6}, {3,7}, {4,8}. Na rysunku 13 klasy abstrakcji są widoczne jako cztery podgrafy spójne.



Rysunek 13.
Graf relacji opisanej macierzą (22)

7 PROGRAM DO BADANIA RELACJI

Macierzowa reprezentacja relacji umożliwia algorytmizację badania własności relacji, a w konsekwencji umożliwia utworzenie programu do badania własności i typów relacji. Taki program mógłby być dobrym treningiem programistycznym, przygotowującym do tworzenia w przyszłości bardziej złożonych programów. Sformułowanie odpowiedniego zadania programistycznego przedstawiono w postaci zbliżonej do formy zadań występujących na zawodach algorytmicznych.

Zadanie:
Koledzy Jasia postanowili zagrać w grę planszową, która składa się z wielu etapów. W każdym z etapów odbywa się rozgrywka, po zakończeniu której, aby przejść na wyższy poziom, trzeba rozwiązać pewne dziwne zadanie matematyczne. Jaś chętnie by wziął udział w grze, ale nie potrafi rozwiązać zadania, gdyż nie uważał w czasie spotkań kółka matematycznego. Potrzebuje pomocy. Pomóż mu, pisząc odpowiedni program.

Rozwiązując zadanie, Jasio powinien odpowiedzieć na pytanie, jakie własności i jaki typ ma pewna relacja dwuczłonowa w zbiorze składającym się z nie więcej niż stu elementów. Za każdym razem relacja jest zdefiniowana w następujący sposób: jeżeli *i*-ty element jest w relacji z elementem *j*-tym, to na wejściu w osobnym wierszu pojawi się rozdzielona spacjami para liczb *i* oraz *j*. Liczba wierszy w zestawie danych jest równa liczbie par (*i*,*j*) dla których zachodzi relacja. O wielkości zbioru, na którym określona jest ta relacja można wnioskować na podstawie największej z liczb *i* albo *j* w parach.

Napisz program, którego wynikiem działania będzie wiersz opisujący przy pomocy odpowiednich skrótów oddzielonych spacjami własności relacji (o ile występują) oraz typ relacji (również, o ile występuje), w następującej kolejności:

Z – zwrotna;
PZ – przeciwzwrotna;
S – symetryczna;
AS – antysymetryczna;
PS – przeciwsymetryczna;
P – przechodnia;
SP – spójna;
RR – relacja równoważności;
RCP – relacja częściowego porządku;
RLP – relacja liniowego porządku;
X – żadna z powyższych.

Przykład:
Wejście:
2 3
2 9
3 4
5 7
5 9
6 7
7 8
8 3
Wyjście:
PZ AS

PODSUMOWANIE

Relacja dwuczłonowa w zbiorze może być przedstawiona przy pomocy macierzy zerojedynkowej, która jest jednocześnie macierzą sąsiedztwa grafu relacji. Graf relacji przez odwoływanie się do wyobraźni, daje możliwość badania relacji w prosty i intuicyjny sposób. Tymczasem reprezentacja macierzowa umożliwia algorytmizację badania własności relacji, a w konsekwencji pozwala na napisanie programu do badania własności i typów relacji. Takie podejście daje możliwość przejścia od abstrakcyjnego modelu matematycznego do konkretnego modelu w postaci algorytmu komputerowego. W wykładzie omówiono algorytmy wykorzystujące tę macierz do badania własności relacji takich jak zwrotność, przeciwzwrotność, symetria, antysymetria, przechodniość oraz spójność relacji. Przedstawiono także przykłady różnych relacji, wraz z określeniem ich własności oraz typów. Stwierdzono, że algorytmy do badania relacji, wykorzystujące operacje na macierzach, mogłyby być dobrym treningiem programistycznym, przygotowującym do tworzenia bardziej złożonych programów. Wobec tego zaproponowano zadanie programistyczne w postaci zbliżonej do zadań występujących na zawodach algorytmicznych. Rozwiązaniem tego zadania byłby program wykorzystujący omówione w wykładzie algorytmy badania relacji.

INFORMACJA
W trakcie przygotowywania niniejszego wykładu do rysowania grafów zastosowano program yEd Graph Editor w wersji 3.6.1.1, dostępny na stronie internetowej http://www.yworks.com, należącej do firmy yWorks GmbH.

LITERATURA

1. Bronsztejn I.N., Siemiendiajew K.A., Musiol G., Mühlig H., *Nowoczesne kompendium matematyki*, WN PWN, Warszawa 2007
2. Rasiowa H., *Wstęp do matematyki współczesnej*, WN PWN, Warszawa 2003
3. Ross K.A., Wright C.R.B., *Matematyka dyskretna*, WN PWN, Warszawa 2003

DODATEK

W wykładzie pojawiły się pojęcia i operacje z obszaru algebry liniowej. Zagadnienia te mogą nie być znane adresatom wykładu. Niniejszy dodatek zawiera kilka podstawowych faktów dotyczących tych zagadnień. Wykład był pisany przy założeniu, że jego odbiorcy znają pojęcie tablicy jednowymiarowej i dwuwymiarowej w języku C++. Tablica jednowymiarowa jest tożsama z wektorem w matematyce. Podobnie, tablica dwuwymiarowa w języku C++ jest tożsama z macierzą. Trzeba tylko pamiętać o tym, że jeżeli w opisie matematycznym składniki *n*-elementowego wektora oraz *n* wierszy lub kolumn w macierzy są indeksowane liczbami od 1 do *n*, to w języku C++ odpowiednie elementy tablicy jednowymiarowej lub wiersze i kolumny tablicy dwuwymiarowej są indeksowane liczbami od 0 do $n - 1$. Wobec powyższego, samo pojęcie wektora oraz macierzy nie będzie tu szerzej omawiane. Natomiast takie pojęcia jak iloczyn skalarny dwóch wektorów, transpozycja macierzy, macierz symetryczna albo algorytm mnożenia macierzy zostaną potraktowane nieco szerzej.

Iloczyn skalarny: operacja przyporządkowująca dwóm wektorom liczbę rzeczywistą. Jeżeli dany jest wektor $a=(a_1, a_2, \ldots, a_n)$ i wektor $b=(b_1, b_2, \ldots, b_n)$, to iloczyn skalarny $a \cdot b$ definiuje się jako sumę wszystkich iloczynów $a_i \cdot b_i$:

$$a \cdot b = \sum_{i=1}^{n} a_i b_i = a_1 b_1 + a_2 b_2 + \ldots + a_n b_n.$$

Macierz kwadratowa: macierz, która ma taką samą liczbę wierszy i kolumn. Macierz A przedstawiona niżej jest przykładem macierzy kwadratowej o rozmiarze 3×3:

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}.$$

Tymczasem pokazane niżej macierze B i C nie są macierzami kwadratowymi, gdyż mają różną liczbę wierszy i kolumn:

$$B = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \\ 0 & 1 & 2 \end{bmatrix}, \qquad C = \begin{bmatrix} 1 & 4 & 7 & 0 \\ 2 & 5 & 8 & 1 \\ 3 & 6 & 9 & 2 \end{bmatrix}.$$

Transpozycja macierzy: przekształcenie macierzy polegające na wzajemnej zamianie (wzajemnym przestawieniu) wierszy z kolumnami. Macierz transponowaną względem macierzy X oznacza się jako X^T. Pokazane wyżej macierze B i C są wzajemnymi transpozycjami: $B = C^T$ oraz $C = B^T$. Dla macierzy A jej transpozycja A^T ma postać:

$$A^T = \begin{bmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{bmatrix}.$$

W wyniku transpozycji elementy na głównej przekątnej macierzy nie ulegają zmianie.

Macierz symetryczna: kwadratowa macierz S o rozmiarze $n{\times}n$ jest symetryczna (względem głównej przekątnej), jeżeli dla każdego $i, j = 1, 2, ..., n$ zachodzi związek $S_{ij} = S_{ji}$:

$$S = S^T = \begin{bmatrix} 1 & 6 & 5 \\ 6 & 2 & 4 \\ 5 & 4 & 3 \end{bmatrix}.$$

Mnożenie macierzy: mnożenie macierzy A przez macierz B jest operacją, której efektem jest macierz wynikowa $C := A{\cdot}B$. Wymaga się przy tym, aby macierz A miała tyle samo kolumn, ile wierszy ma macierz B. Wynikowa macierz C ma tyle wierszy, ile wierszy posiada macierz A i tyle kolumn, ile kolumn ma macierz B. Jeżeli założyć, że A i B są macierzami kwadratowymi o rozmiarze $n{\times}n$, to także macierz C jest kwadratowa i ma identyczny rozmiar. Rozważa się 3 macierze kwadratowe A, B oraz C, każdą o rozmiarze $n{\times}n$. Macierze A i B są dane, zaś macierz C jest nieznana:

$$A = \begin{bmatrix} a_{11} & * & * & * & a_{1n} \\ * & * & * & * & * \\ a_{i1} & a_{i2} & \dots & \dots & a_{in} \\ * & * & * & * & * \\ a_{n1} & * & * & * & a_{nn} \end{bmatrix} \qquad B = \begin{bmatrix} * & * & b_{1j} & * & * \\ * & * & b_{2j} & * & * \\ \dots & \dots & \vdots & \dots & \dots \\ * & * & \vdots & * & * \\ * & * & b_{nj} & * & * \end{bmatrix} \qquad C = \begin{bmatrix} * & * & \vdots & * & * \\ * & * & \vdots & * & * \\ \dots & \dots & c_{ij} & \dots & \dots \\ * & * & \vdots & * & * \\ * & * & \vdots & * & * \end{bmatrix}$$

Należy znaleźć macierz C jako iloczyn macierzy A i B:

$$C := A{\cdot}B\,.$$

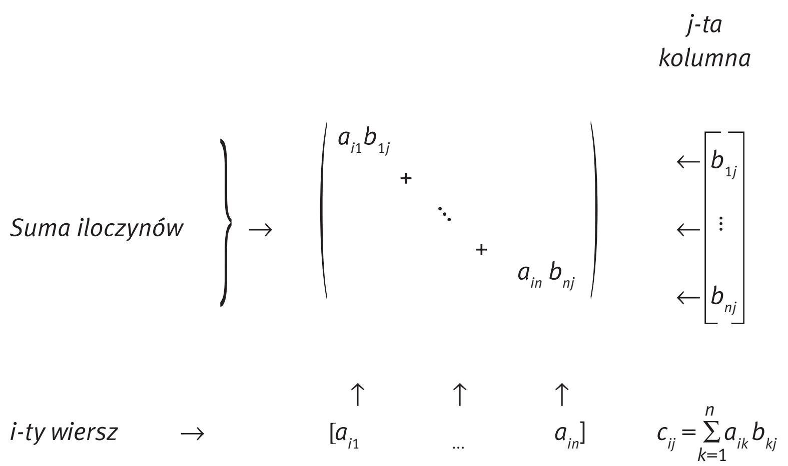
Pojedynczy element znajdujący się w i-tym wierszu oraz w j-tej kolumnie macierzy C (oznaczony jako c_{ij}) liczy się, jako iloczyn skalarny i-tego wiersza macierzy A i j-tej kolumny macierzy B. Dla podkreślenia tego faktu, operację tę można zapisać w następujący sposób:

$$\begin{array}{cc} & \begin{bmatrix} * & * & b_{1j} & * & * \\ * & * & b_{2j} & * & * \\ \dots & \dots & \vdots & \dots & \dots \\ * & * & \vdots & * & * \\ * & * & b_{nj} & * & * \end{bmatrix} \\ \begin{bmatrix} * & * & * & * & * \\ * & * & * & * & * \\ a_{i1} & a_{i2} & \dots & \dots & a_{in} \\ * & * & * & * & * \\ * & * & * & * & * \end{bmatrix} & \begin{bmatrix} * & * & \vdots & * & * \\ * & * & \vdots & * & * \\ \dots & \dots & c_{ij} & \dots & \dots \\ * & * & \vdots & * & * \\ * & * & \vdots & * & * \end{bmatrix} \end{array}$$

Dla znalezienia wszystkich elementów macierzy C należy znaleźć wszystkie iloczyny skalarne odpowiednich wierszy macierzy A oraz odpowiednich kolumn macierzy B. Funkcja służąca do znajdowania elementów macierzy C ma następującą postać:

```
void mnozenie_macierzy(int A[SIZE][SIZE], int B[SIZE][SIZE], int C[SIZE][SIZE], int n)
{  int i,j;
// C := A*B
   for (i=0; i<n; i++)
      for (j=0; j<n; j++)
      { C[i][j]=iloczyn_skalarny(i-ty wiersz macierzy A, j-ta kolumna macierzy B);
      }
}
```

Iloczyn skalarny *i*-tego wiersza macierzy *A* i *j*-tej kolumny macierzy *B* można obrazowo przedstawić w postaci następującego schematu:



Iloczyn ten jest sumą *n* iloczynów $a_{ik}b_{kj}$. Ostatecznie algorytm mnożenia macierzy ma następującą postać:

```
void mnozenie_macierzy(int A[SIZE][SIZE], int B[SIZE][SIZE], int C[SIZE][SIZE], int n)
{  int i,j;
//mnożenie macierzy: C := A*B
   for (i=0; i<n; i++)
      for (j=0; j<n; j++)
      { //iloczyn skalarny i-tego wiersza macierzy A
        //oraz j-tej kolumny macierzy B
        C[i][j]=0;
        for (k=0;k<n;k++) C[i][j] += A[i][k] * B[k][j];
      }
}
```