МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІСЬКА ПОЛІТЕХНІКА»

Кафедра ІСМ



**Звіт**

До лабораторної роботи №2

З дисципліни:

«Теорія прийняття рішень»

На тему:

**«Дослідження властивостей та основних типів**

**бінарних відношень»**

Виконав:

Студент групи КН-38

Макара С.В.

Прийняла:

Заяць М.М.

**Львів-2017**

**Мета:** Вивчення та практичне ознайомлення з основними властивостями бінарних відношень та їх типами.

**Завдання:**

Для базового класу або його реалізацій написати функції перевірки властивостей відношення: рефлективності, антирефлексивності, симетричності, асиметричності, антисиметричності, транзитивності, ациклічності.

Для базового класу написати функції перевірки належності відношення до певного типу: толерантності, еквівалентності, квазіпорядку, порядку, строгого порядку, лінійного порядку, строгого лінійного порядку.

Для базового класу або його реалізацій написати функції знаходження симетричної та асиметричної складової, транзитивного замикання, досягальності та взаємної досягальності.

**Текст програми:**

**import** java.util.Scanner;

**public** **class** second {

**static** **int** [][] petetun(**int** [][] A, **int** [][] B)

{

**int** [][] R = **new** **int**[A.length][A.length];

**for**(**int** i = 0;i<A.length;i++)

**for**(**int** j = 0;j<A.length;j++)

R[i][j] = A[i][j]&B[i][j];

**return** R;

}

**static** **int**[][] plus(**int**[][] A, **int**[][] B)

{

**int** [][] R = **new** **int**[A.length][A.length];

**for**(**int** i = 0;i<A.length;i++)

**for**(**int** j = 0;j<A.length;j++)

R[i][j] = A[i][j]|B[i][j];

**return** R;

}

**static** **int**[][] minus(**int**[][] A, **int**[][] B) {

**int** [][] R = **new** **int**[A.length][A.length];

**for**(**int** i = 0;i<A.length;i++)

**for**(**int** j = 0;j<A.length;j++)

**if**(A[i][j]==1&&B[i][j]==0)

R[i][j]=1;

**else** R[i][j]=0;

**return** R;

}

**static** **int**[][] sminus(**int**[][] A, **int**[][] B)

{

**return** *plus*(*minus*(A, B), *minus*(B, A));//(A\B)U(B\A)

}

**static** **int**[][] dopovv(**int**[][] A)

{

**int** [][] R = **new** **int**[A.length][A.length];

**for** (**int** i = 0; i < A.length; i++)

**for** (**int** j = 0; j < A.length; j++)

{

**if** (A[i][j] == 1) R[i][j] = 0;

**else** R[i][j] = 1;

}

**return** R;

}

**static** **int**[][] revers(**int**[][] A)

{

**int** [][] R = **new** **int**[A.length][A.length];

**for** (**int** i = 0; i < A.length; i++)

**for** (**int** j = 0; j < A.length; j++)

R[i][j] = A[j][i];

**return** R;

}

**static** **int**[][] compose(**int**[][] A, **int**[][] B)

{

**int** [][] R = **new** **int**[A.length][A.length];

**for** (**int** i = 0; i < A.length; i++)

**for** (**int** j = 0; j < A[i].length; j++)

{

R[i][j] = 0;

**for** (**int** k = 0; k < A.length; k++)

{

R[i][j] += A[i][k] \* B[k][j];

}

}

**for** (**int** i = 0; i < A.length; i++)

**for** (**int** j = 0; j < B.length; j++)

**if**(R[i][j] > 0) R[i][j] = 1;

**return** R;

}

**static** **int** [][] littleone(**int** [][] A)

{

Scanner read = **new** Scanner(System.***in***);

System.***out***.println("Кількість елементів звуженої матриці:");

**int** con = read.nextInt(), count\_i = -1, count\_j;

**int** [] temp = **new** **int**[con];

**int** [][] R = **new** **int**[con][con];

System.***out***.println("Порядкові номери множин:");

**for**(**int** i = 0;i<con;i++)

{

temp[i] = read.nextInt()-1;

}

**for**(**int** i = 0;i<con;i++)

{

++count\_i;

**if**(temp[count\_i]==i)

{

count\_j = 0;

**for**(**int** j = 0;j<con;j++)

**if**(temp[count\_j] == i)

{

R[count\_i][count\_j] = A[i][j];

count\_j+=1;

}

}

}

**return** R;

}

**static** **int** [][] doubleOne(**int** [][] A)

{

**return** *revers*(*dopovv*(A));

}

**public** **static** **int** [][] inputing(**int** n)

{

**int** [][] A = **new** **int**[n][n];

Scanner scn = **new** Scanner(System.***in***);

**for**(**int** i = 0;i<n;i++)

{

System.***out***.println("Введіть "+ (i+1) + " рядок:");

**for**(**int** j = 0;j<n;j++)

A[i][j] = scn.nextInt();

}

**return** A;

}

**public** **static** **int** reflect(**int** [][] A)

{

**int** r = 0, j2 = 0;

**boolean** re = **true**, are = **true**;

**int** [][] E = **new** **int**[A.length][A.length];

**for**(**int** i = 0;i<E.length;i++)

{

**for**(**int** j = 0;j<E.length;j++)

**if**(j==j2&&i==j2) E[i][j] = 1;

**else** E[i][j] = 0;

j2++;

}

j2 = 0;

**for**(**int** i = 0;i<E.length;i++)

{

**for**(**int** j = 0;j<E.length;j++)

**if**(j==j2&&i==j2)

**if**(E[i][j] != A[i][j]) re = **false**;

j2++;

}

A = *petetun*(A, E);

**for**(**int** i = 0;i<E.length;i++)

**for**(**int** j = 0;j<E.length;j++)

**if**(A[i][j] != 0)

are = **false**;

**if**(are) r = 2;

**if**(re) r = 1;

**return** r;

}

**public** **static** **int** simetr(**int** [][] A)

{

**int** s = 0, j2 = 0;

**boolean** su = **true**, asu = **true**, antsu1 = **false**, antsu2 = **true**;

**int** [][] Ap = *petetun*(A, *revers*(A));

**for**(**int** i = 0;i<A.length;i++)

**for**(**int** j = 0;j<A.length;j++)

**if**(A[i][j] != A[j][i])

su = **false**;

**for**(**int** i = 0;i<Ap.length;i++)

**for**(**int** j = 0;j<Ap.length;j++)

**if**(Ap[i][j] != 0)

asu = **false**;

**for**(**int** i = 0;i<Ap.length;i++)

{

**for**(**int** j = 0;j<Ap.length;j++)

**if**(j==j2&&i==j2){

**if**(Ap[i][j]==1)

antsu1 = **true**;}

**else**

**if**(Ap[i][j]==1)

antsu2 = **false**;

j2++;

}

**if**(su) s = 1;

**if**(asu) s = 2;

**if**(antsu1&&antsu2) s = 3;

**return** s;

}

**public** **static** **int** tranz(**int** [][] A)

{

**int** tr = 0;

**boolean** tra = **true**;

**int**[][] pA = **new** **int** [A.length][A.length];

pA = *compose*(A, A);

**for**(**int** i = 0;i<A.length;i++)

**for**(**int** j = 0;j<A.length;j++)

**if**(A[i][j] == 0 && pA[i][j]==1) tra = **false**;

**if**(tra) tr = 1;

**return** tr;

}

**public** **static** **int** looping(**int** [][] A)

{

**int** lo = 0;

**int** [][] pA = *petetun*(*compose*(A, *compose*(A, A)), *revers*(A));

**boolean** lop = **true**;

**for**(**int** i = 0;i<pA.length;i++)

**for**(**int** j = 0;j<pA.length;j++)

**if**(pA[i][j]!=0)

lop = **false**;

**if**(lop) lo = 1;

**return** lo;

}

**public** **static** **int** [][] simetric(**int** [][] A)

{

**return** *petetun*(A, *revers*(A));

}

**public** **static** **int** [][] asimetric(**int** [][] A)

{

**return** *minus*(A, *petetun*(A, *revers*(A)));

}

**public** **static** **int** [][] transzam(**int** [][] A)

{

**int** [][] A2 = *plus*(A, *compose*(A, A));

**if**(*tranz*(A2)!=1) A2 = *plus*(A2, *compose*(A, A2));

**else** **if**(*tranz*(A2)!=1) A2 = *plus*(A2, *compose*(A, A2));

**else** **if**(*tranz*(A2)!=1) A2 = *plus*(A2, *compose*(A, A2));

**else** **if**(*tranz*(A2)!=1) A2 = *plus*(A2, *compose*(A, A2));

**else** **if**(*tranz*(A2)!=1) A2 = *plus*(A2, *compose*(A, A2));

**return** A2;

}

**public** **static** **int** [][] taken(**int** [][] A)

{

**int** j2 = 0;

**int** [][] E = **new** **int**[A.length][A.length];

**int** [][] tA = **new** **int**[A.length][A.length];

**for**(**int** i = 0;i<E.length;i++)

{

**for**(**int** j = 0;j<E.length;j++)

**if**(j==j2&&i==j2) E[i][j] = 1;

**else** E[i][j] = 0;

j2++;

}

tA = *plus*(E, A);

**return** tA;

}

**public** **static** **void** outputing(**int** [][] A)

{

**for**(**int** i = 0;i<A.length;i++)

{

**for**(**int** j = 0;j<A.length;j++)

System.***out***.print(A[i][j] + " ");

System.***out***.println();

}

}

**public** **static** **void** scan(**int** [][] A)

{

**int** refl = *reflect*(A), sim = *simetr*(A), tran = *tranz*(A), lop = *looping*(A);

System.***out***.println("Властивості відношення:");

**if**(refl == 1) System.***out***.println("рефлективне");

**if**(refl == 2) System.***out***.println("антирефлективне");

**if**(refl == 0) System.***out***.println("ні рефлективне, ні антирефлективне");

**if**(sim == 1) System.***out***.println("симетричне");

**if**(sim == 2) System.***out***.println("асиметричне");

**if**(sim == 3) System.***out***.println("антисиметричне");

**if**(sim == 0) System.***out***.println("ні симетричне, ні асиметричне, ні антисиметричне");

**if**(tran == 1) System.***out***.println("транзитивне");

**if**(tran == 0) System.***out***.println("не транзитивне");

**if**(lop == 1) System.***out***.println("ациклічне");

**if**(lop == 0) System.***out***.println("не ациклічне");

System.***out***.println("Належності до типів:");

**if**(refl == 1 && sim == 1) System.***out***.println("толерантності");

**if**(refl == 1 && sim == 1 && tran == 1) System.***out***.println("еквівалентності");

**if**(refl == 1 && tran == 1) System.***out***.println("квазіпорядку");

**if**(refl == 1 && sim == 3 && tran == 1) System.***out***.println("порядку");

**if**(sim == 2 && tran == 1) System.***out***.println("стогого порядку");

**if**(refl == 1 && sim == 3 && tran == 1) System.***out***.println("лінійного порядку");

**if**(tran == 0) {

System.***out***.println("Транзитивне замикання:");

*outputing*(*transzam*(A));

}

System.***out***.println("Симетрична складова:");

*outputing*(*simetric*(A));

System.***out***.println("Асиметрична складова:");

*outputing*(*asimetric*(A));

**if**(tran == 0){

System.***out***.println("Досягальність:");

*outputing*(*taken*(*transzam*(A)));

System.***out***.println("Взаємна досягальність:");

*outputing*(*petetun*(*taken*(*transzam*(A)), *revers*(*taken*(*transzam*(A)))));

}

**else**{

System.***out***.println("Досягальність:");

*outputing*(*taken*(A));

System.***out***.println("Взаємна досягальність:");

*outputing*(*petetun*(*taken*(A), *revers*(*taken*(A))));

}

}

**public** **static** **void** main(String[] args) {

Scanner scn = **new** Scanner(System.***in***);

**int** num = 0;

**do**{

System.***out***.println("Розмірність матриці:");

num = scn.nextInt();

} **while**(num<=1 || num > 10);

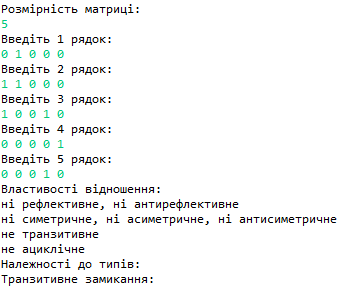
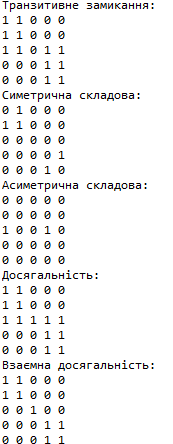
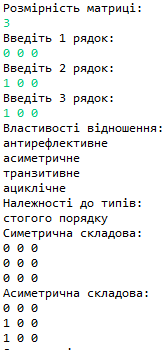
**int** [][] A = *inputing*(num);

*scan*(A);

}

}

**Приклади роботи програми**

****

*Рис.1. Результати тестових прикладів*

**Висновки:** На даній лабораторній роботі мною було вивчено основні властивості бінарних відношень та їх типів. Набуто практичних навиків з реалізації автоматичного знаходження типу та функцій відношень.