**Міністерство освіти і науки України**

**Національному університеті "Львівська**

**Політехніка"**

**Кафедра систем штучного інтелекту**

**Лабораторна робота № 3**

з дисципліни

**Виконав:**

студент групи КН-114

Сиротюк Владислав

**Викладач:**

Мельникова Н.І.

Львів - 2019р.

**Лабораторна робота № 3.**

**Тема:** Побудова матриці бінарного відношення

**Мета роботи:** набуття практичних вмінь та навичок при побудові матриць бінарних відношень та визначені їх типів.

**ТЕОРЕТИЧНІ ВІДОМОСТІ ТА ПРИКЛАДИ РОЗВ’ЯЗАННЯ ЗАДАЧ**

Декартів добуток множин А і В (позначається A× B) – це множина всіх упорядкованих пар елементів (a,b), де a ∈ A, b∈ B.

При цьому вважається, що (a1,b1) = (a2,b2) тоді і тільки тоді, коли a1 = a2 , b1 = b2. Потужність декартова добутку дорівнює B .× A = B ×A

Приклад. Довести тотожність (A×B)∩(C×D)=(A∩C) × (B∩D).

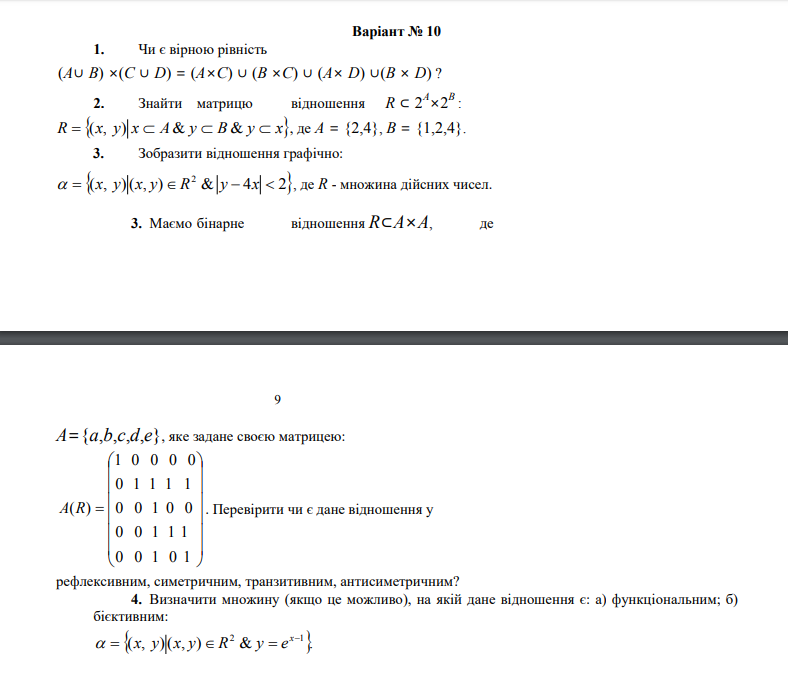
Розв’язання. Нехай (x , y)∈(A× B) ∩ (C × D) ⇔ (x, y)∈(A× B) & (x, y)∈ (C × D) ⇔ (x ∈ A& y ∈ B) & (x ∈C & y ∈ D) ⇔ (x ∈ A& x ∈C) & (y ∈ B & y ∈ D) ⇔ (x ∈ A∩ C) & (y ∈ B ∩ D) ⇔ (x, y)∈(A∩C)×(B ∩ D) .

Бінарним відношенням R називається підмножина декартового добутку A×B ( тобто R ⊂ A×B ). Якщо пара (a,b) належить відношенню R , то пишуть (a, b)∈R , або aRb .

Областю визначення бінарного відношення R ⊂ X ×Y називається множина Rδ }R∈y (x, y)∃x { = , а областю значень – множина Rρ }R∈x (x, y)∃y { = (∃- існує ).

Для скінчених множин бінарне відношення R ⊂ A×B зручно задавати за допомогою матриці відношення Rm×n = (rij ) , де A , а=m B .=n

**Завдання №1**

****

**Розв`язання**

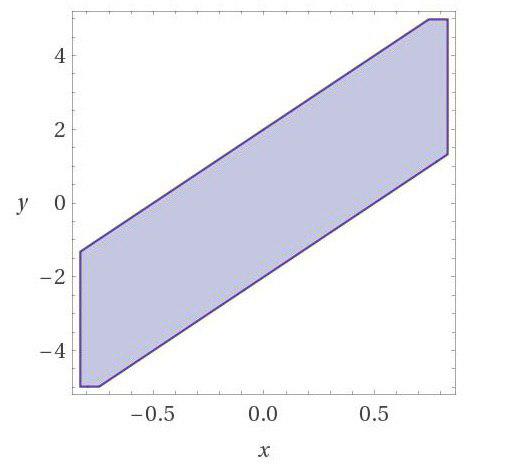
1. Нехай(x,y) ∈ (A x C) ∪ (B x C) ∪ (A x D) ∪ (B x D) 🡨🡪(x,y) ∈ (A x C) ∈ ∈(B x C) ∈ (A x D) ∈ (B x D) 🡨🡪(x ∈ A & y ∈ C) & (x ∈ B & y ∈ C) & (x ∈ A & y ∈D ) & (x ∈ B & y ∈ D) 🡨🡪 (x ∈ A & x ∈ B & x ∈ A & x ∈ B) & (y ∈ C & y ∈ C & y ∈ D & y ∈ D) 🡨🡪(x∈A ∪B) & (y∈C ∪ D).Отже,рівність вірна.

m = |2^A| = 2^2 = 4;

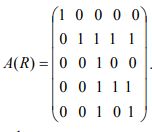
n = |2^B| = 2^3 = 8;

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | {} | {1} | {2} | {4} | {1,2} | {1,4} | {2,4} | {1,2,4} |
| {} | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| {2} | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| {4} | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| {2,4} | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 |

1. |y-4x|< 2 => y-4x> -2

y – 4x < 2 

**4.**

****

Дане відношення:

-Рефлексивне(бо діагональ заповнена одиницями);

-Симетричне(бо при транспортуванні вигляд матриці не міняється,симетричною вважається будь-яка діагональна матриця;

-Транзитивна();

-Не є антисиметричною(бо у антисиметричних матриць всі елементи діагоналі = 0)

**5.**

**1)На множині дійсних чисел ,відношення є функціональним,бо кожному x відповідає максимум 1 y,тому відношення функціональне на всій R;**

**2)На множині дійсних чисел,якщо f(x1) = f(x2) => x1 = x2,отже воно є бієктивним.**

**Завдання №2**

**C:\Users\ACER\Desktop\Screenshot_3.png**

**Хід виконання програми**

#include <iostream>

#include <sstream>

using namespace std;

int firstSize;

int secondSize;

int\* a;

int\* b;

int\*\* binarnaMatr;

void sortArray(int\* array, int size\_arr) {

int temp;

for (int j = 0; j < size\_arr - 1; j++) {

for (int i = 0; i < size\_arr - 1; i++) {

if (array[i] > array[i + 1]) {

temp = array[i];

array[i] = array[i + 1];

array[i + 1] = temp;

}

}

}

}

bool checkRefleks(int\*\* matr, int ryad, int col) {

if (ryad != col) {

return false;

}

for (int i = 0; i < ryad && i < col;i++) {

if (matr[i][i] == 0) {

return false;

}

}

return true;

}

bool checkAntirefleks(int\*\* matr, int ryad, int col) {

if (ryad != col) {

return false;

}

for (int i = 0; i < ryad && i < col;i++) {

if (matr[i][i] == 1) {

return false;

}

}

return true;

}

bool checkSymmetry(int\*\* matr, int ryad, int col) {

if (ryad != col) {

return false;

}

for (int i = 0; i < ryad;i++) {

for (int j = 0; j < ryad; j++) {

if (matr[i][j] != matr[j][i]) {

return false;

}

}

}

return true;

}

bool checkAntisymmetry(int\*\* matr, int ryad, int col) {

if (ryad != col) {

return false;

}

for (int i = 0; i < ryad;i++) {

for (int j = 0; j < ryad; j++) {

if (i != j && matr[i][j] == matr[j][i]) {

return false;

}

}

}

return true;

}

bool checkTrans(int\*\* matr, int ryad, int col) {

if (ryad != col) {

return false;

}

bool temp = false;

for (int i = 0; i < ryad;i++) {

for (int j = 0; j < ryad; j++) {

for (int m = 0; m < ryad;m++) {

if (matr[i][j] == 1 && matr[j][m] == 1) {

temp = true;

if (matr[i][m] != 1) {

return false;

}

}

}

}

}

if (temp) {

return true;

}

else {

return false;

}

}

bool checkAntitrans(int\*\* matr, int ryad, int col) {

if (ryad != col) {

return false;

}

bool temp = false;

for (int i = 0; i < ryad;i++) {

for (int j = 0; j < ryad; j++) {

for (int m = 0; m < ryad;m++) {

if (matr[i][j] == 1 && matr[j][m] == 1) {

temp = true;

if (matr[i][m] != 0) {

return false;

}

}

}

}

}

if (temp) {

return true;

}

else {

return false;

}

}

int main() {

cout << "size1 = ";

cin >> firstSize;

cout << "size2 = ";

cin >> secondSize;

cout << endl;

a = new int[firstSize];

b = new int[secondSize];

binarnaMatr = new int\* [firstSize];

for (int i = 0; i < firstSize;i++) {

binarnaMatr[i] = new int[secondSize];

}

cout << "a:" << endl;

int number;

string line;

int size = 0;

cout << "[" << size << "] = ";

while (getline(cin, line))

{

stringstream ss(line);

if (ss >> number)

{

if (ss.eof())

{

bool present = 0;

for (int i = 0; i < size; i++) {

if (a[i] == number) present = 1;

}

if (!present) {

a[size] = number;

size++;

if (size == firstSize) break;

cout << "[" << size << "] = ";

}

}

}

}

cout << "b:" << endl;

size = 0;

cout << "[" << size << "] = ";

while (getline(cin, line))

{

stringstream ss(line);

if (ss >> number)

{

if (ss.eof())

{

bool present = 0;

for (int i = 0; i < size; i++) {

if (b[i] == number) present = 1;

}

if (!present) {

b[size] = number;

size++;

if (size == secondSize) break;

cout << "[" << size << "] = ";

}

}

}

}

cout << endl;

sortArray(a, firstSize);

sortArray(b, secondSize);

cout << "a:";

for (int i = 0; i < firstSize;i++) {

cout << a[i] << " ";

}

cout << endl << endl;

cout << "b:";

for (int i = 0; i < secondSize;i++) {

cout << b[i] << " ";

}

cout << endl << endl;

for (int row = 0; row < firstSize;row++) {

for (int column = 0; column < secondSize; column++) {

if (a[row] < 2 \* b[column] + 1) {

binarnaMatr[row][column] = 1;

}

else {

binarnaMatr[row][column] = 0;

}

}

}

cout << "Matr:" << endl;

for (int i = 0; i < firstSize;i++) {

for (int j = 0; j < secondSize;j++) {

cout << binarnaMatr[i][j] << " ";

}

cout << endl;

}

cout << endl;

if (checkRefleks(binarnaMatr, firstSize, secondSize)) {

cout << "matr is reflexive " << endl;

}

if (checkAntirefleks(binarnaMatr, firstSize, secondSize)) {

cout << "matr is antireflexive" << endl;

}

if (checkSymmetry(binarnaMatr, firstSize, secondSize)) {

cout << "matr is symmetry" << endl;

}

if (checkAntisymmetry(binarnaMatr, firstSize, secondSize)) {

cout << "matr is antisymmetry" << endl;

}

if (checkTrans(binarnaMatr, firstSize, secondSize)) {

cout << "matr is transitive" << endl;

}

if (checkAntitrans(binarnaMatr, firstSize, secondSize)) {

cout << "matr is antitransitive" << endl;

}

return 0;

}