МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ “ЛЬВІВСЬКА ПОЛІТЕХНІКА”

Кафедра інформаційних систем та мереж

ЗВІТ

до лабораторної роботи №3

з дисципліни: **“Теорія прийняття рішень”**

на тему:

**«Дослідження операцій з метризованими бінарними**

**відношеннями та їх властивостей»**

**Виконав:**

студент групи KH-318

Гавриляк В.Т.

**Прийняла:**

асистент Заяць М.М.

Львів-2019

**Мета роботи:**

Вивчення та практичне ознайомлення з основними алгебрами метризованих бінарних відношень та з властивостями і типами метризованих бінарних відношень.

**Теоретична частина**

Апарат бінарних відношень у теорії прийняття рішень є теоретичним підґрунтям для оцінювання переваг альтернатив шляхом попарних порівнянь, оскільки децидентові значно простіше порівняти дві альтернативи ніж багато. Відношення - це твердження, яке відображає взаємний зв’язок між двома чи більшою кількістю об’єктів.

Бінарним відношенням на множині називається підмножина декартового добутку , тобто . Якщо пара (x, y) знаходиться у відношенні , то цей факт позначається як або . Множина А називається носієм відношення . Відношення можна означити і для n об‘єктів. У цьому разі говорять про парне відношення. Бінарне відношення може бути задане або переліком пар, що знаходяться в цьому відношенні, або за допомогою правила, що дозволяє з’ясувати, чи знаходиться пара в заданому відношенні, чи ні.

Звичайні бінарні відношення дозволяють зробити висновок про те, чи краща одна з альтернатив за іншу, але не дозволяють оцінити силу такої переваги.

У багатьох випадках експерт може дати порівняльну кількісну оцінку альтернатив – на скільки чи у скільки разів одна альтернатива краща за іншу. Отже, оскільки результатом експертного опитування можуть бути відношення різних типів, то формалізація поняття "близькості" на множині відношень є важливою проблемою.

Для кількісного оцінювання альтернатив уведено поняття метризованого відношення.

Метризоване відношення , це двійка =, де P – бінарне відношення, а , де – число, що характеризує ступінь переваги альтернативи над альтернативою або ж у разі толерантності – ступінь схожості альтернатив. Метризоване відношення = називається рефлексивним, антирефлексивним, симетричним, асиметричним, антисиметричним, транзитивним, якщо відношення Р має відповідні властивості.

**Варіант 5**

**Завдання.**

1. Робота виконується на базі реалізованих в лабораторних роботах №1,2 базового віртуального об’єкту “ВІДНОШЕННЯ” і його реалізації “ВІДНОШЕННЯ\_МАТР”.
2. Декларувати клас нащадок “ВІДНОШЕННЯ\_МАТР\_МЕТР” від класу “ВІДНОШЕННЯ\_МАТР” для роботи з метризованими відношеннями. Декларація повинна містити додаткові функції отримання та встановлення значень mij (наприклад get\_v та set\_v), перевизначення операцій для роботи з метризованими відношеннями, функцію визначення виду відношення (адитивне, мультиплікативне) та функції перевірки властивостей характерних для метризованих відношень (узгодженість, адитивна чи мультиплікативна транзитивність).
3. Здійснити реалізацію цих функцій. При реалізації операцій з метризованими відношеннями необхідно враховувати вид відношення (адитивне, мультиплікативне) та реалізовувати відповідну алгебру.
4. За допомогою отриманої програми визначити властивості та тип відношень P та Q, та здійснити обчислення основних операцій (об‘єднання, перетин, різниця, композиція) з допомогою відповідної алгебри. Значення відношення P та Q згідно варіанту.
5. Повторити дії з п.4 для відношення P1 та Q1.
6. Оформити звіт про виконану роботу.

**Код:**

**#include <iostream>**

**#include "iomanip"**

**#include <math.h>**

**using namespace std;**

**int adutuvne(int A[][5]);**

**int multuplikatuvne(int A[][5]);**

**void a\_obiednannia(double A[][5], double B[][5], double C[][5]);**

**void a\_peretun(double A[][5], double B[][5], double C[][5]);**

**void a\_riznucia(double A[][5], double B[][5], double C[][5]);**

**void a\_kompozucia(double A[][5], double B[][5], double C[][5]);**

**void m\_obiednannia(double A[][5], double B[][5], double C[][5]);**

**void m\_peretun(double A[][5], double B[][5], double C[][5]);**

**void m\_riznucia(double A[][5], double B[][5], double C[][5]);**

**void m\_kompozucia(double A[][5], double B[][5], double C[][5]);**

**double A[5][5] =**

**{**

**0,0,0,0,0,**

**1,0,0,0,0,**

**5,4,0,2,3,**

**3,2,0,0,1,**

**2,1,0,0,0**

**};**

**double B[5][5] =**

**{**

**0,0,0,0,0,**

**5,0,0,0,0,**

**0,0,0,0,0,**

**4,0,0,0,0,**

**7,2,0,3,0**

**};**

**double P1[5][5] =**

**{**

**1,0,0,0,7,**

**0,1,0,0,0,**

**0,0,1,0,0,**

**0,6,0,1,0,**

**0,0,0,0,1**

**};**

**double P2[5][5]**

**{**

**1,0,0,0,0,**

**0,1,2,0,8,**

**0,0,1,0,4,**

**1,1,2,1,8,**

**0,0,0,0,1**

**};**

**double res[5][5];**

**class VIDNOSHENNIA**

**{**

**public:**

**int adutuvne(double A[][5])**

**{**

**for (int i = 0; i < 5; i++)**

**for (int j = 0; j < 5; j++)**

**{**

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

**{**

**cout << "Вiдношення адитивне :" << endl;**

**return 1;**

**}**

**} return 0;**

**}**

**int multuplikatuvne(double A[][5])**

**{**

**for (int i = 0; i < 5; i++)**

**for (int j = 0; j < 5; j++)**

**{**

**if (A[i][i] == 1)**

**{**

**cout << "Вiдношення мультиплiкативне :" << endl;**

**return 1;**

**}**

**} return 0;**

**} public:**

**};**

**class VIDNOSHENNIA\_MATR : public virtual VIDNOSHENNIA**

**{**

**public:**

**virtual void a\_obiednannia(double A[][5], double B[][5], double C[][5])**

**{**

**for (int i = 0; i < 5; i++)**

**{**

**for (int j = 0; j < 5; j++)**

**{**

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

**{**

**C[i][j] = (A[i][j] + B[i][j]) / 2;**

**}**

**else {**

**C[i][j] = A[i][j] + B[i][j];**

**}**

**}**

**}**

**for (int i = 0; i < 5; i++)**

**{**

**for (int j = 0; j < 5; j++)**

**{**

**cout << setw(4) << C[i][j];**

**}**

**cout << endl;**

**}**

**}**

**virtual void a\_peretun(double A[][5], double B[][5], double C[][5])**

**{**

**for (int i = 0; i < 5; i++)**

**{**

**for (int j = 0; j < 5; j++)**

**{**

**if (A[i][j] && B[i][j])**

**{**

**C[i][j] = (A[i][j] + B[i][j]) / 2;**

**}**

**else {**

**C[i][j] = 0;**

**}**

**}**

**}**

**for (int i = 0; i < 5; i++)**

**{**

**for (int j = 0; j < 5; j++)**

**{**

**cout << setw(4) << C[i][j];**

**}**

**cout << endl;**

**}**

**}**

**virtual void a\_riznucia(double A[][5], double B[][5], double C[][5])**

**{**

**for (int i = 0; i < 5; i++)**

**{**

**for (int j = 0; j < 5; j++)**

**{**

**{**

**if (A[i][j] && !B[i][j]) {**

**C[i][j] = A[i][j];**

**}**

**else if (A[i][j] && B[i][j])**

**{**

**C[i][j] = 0;**

**}**

**else**

**{**

**C[i][j] = 0;**

**}**

**}**

**}**

**}**

**for (int i = 0; i < 5; i++)**

**{**

**for (int j = 0; j < 5; j++)**

**{**

**cout << setw(4) << C[i][j];**

**}**

**cout << endl;**

**}**

**}**

**virtual void a\_kompozucia(double A[][5], double B[][5], double C[][5])**

**{**

**int i, j, k, count = 0;**

**for (i = 0; i < 5; ++i) {**

**for (j = 0; j < 5; ++j)**

**{**

**C[i][j] = 0;**

**}**

**for (i = 0; i < 5; ++i) {**

**for (j = 0; j < 5; ++j) {**

**for (k = 0; k < 5; ++k)**

**{**

**if (A[i][k] > 0 && B[k][j] > 0)**

**{**

**C[i][j] += (A[i][k] + B[k][j]);**

**count++;**

**}**

**// cout << "count = " << count << endl;**

**}**

**if (count > 0)**

**{**

**C[i][j] = C[i][j] / count;**

**count = 0;**

**}**

**}**

**}**

**for (i = 0; i < 5; ++i) {**

**for (j = 0; j < 5; ++j)**

**{**

**cout << setw(6) << C[i][j];**

**}**

**cout << endl;**

**}**

**}**

**}**

**virtual void m\_obiednannia(double A[][5], double B[][5], double C[][5])**

**{**

**for (int i = 0; i < 5; ++i) {**

**for (int j = 0; j < 5; ++j) {**

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

**{**

**C[i][j] = sqrt(A[i][j] \* B[i][j]);**

**}**

**else {**

**C[i][j] = A[i][j] + B[i][j];**

**}**

**}**

**}**

**for (int i = 0; i < 5; ++i) {**

**for (int j = 0; j < 5; ++j)**

**{**

**cout << setw(10) << C[i][j];**

**}**

**cout << endl;**

**}**

**}**

**virtual void m\_peretun(double A[][5], double B[][5], double C[][5])**

**{**

**for (int i = 0; i < 5; ++i) {**

**for (int j = 0; j < 5; ++j) {**

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

**{**

**C[i][j] = sqrt(A[i][j] \* B[i][j]);**

**}**

**else**

**C[i][j] = 0;**

**}**

**}**

**for (int i = 0; i < 5; ++i) {**

**for (int j = 0; j < 5; ++j)**

**{**

**cout << setw(10) << C[i][j];**

**}**

**cout << endl;**

**}**

**}**

**virtual void m\_riznucia(double A[][5], double B[][5], double C[][5])**

**{**

**for (int i = 0; i < 5; ++i) {**

**for (int j = 0; j < 5; ++j) {**

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

**{**

**C[i][j] = A[i][j];**

**}**

**else**

**C[i][j] = 0;**

**}**

**}**

**for (int i = 0; i < 5; ++i) {**

**for (int j = 0; j < 5; ++j)**

**{**

**cout << setw(10) << C[i][j];**

**}**

**cout << endl;**

**}**

**}**

**virtual void m\_kompozucia(double A[][5], double B[][5], double C[][5])**

**{**

**int i, j, k;**

**double count = 0;**

**for (i = 0; i < 5; ++i) {**

**for (j = 0; j < 5; ++j)**

**{**

**C[i][j] = 0;**

**}**

**for (i = 0; i < 5; ++i) {**

**for (j = 0; j < 5; ++j) {**

**for (k = 0; k < 5; ++k)**

**{**

**if (A[i][k] > 0 && B[k][j] > 0)**

**{**

**C[i][j] += (A[i][k] \* B[k][j]);**

**count++;**

**}**

**}**

**if (count > 0)**

**{**

**C[i][j] = pow(C[i][j], 1 / count);**

**count = 0;**

**}**

**}**

**for (i = 0; i < 5; ++i) {**

**for (j = 0; j < 5; ++j)**

**{**

**cout << setw(8) << C[i][j];**

**}**

**cout << endl;**

**}**

**}**

**}**

**};**

**};**

**int main(int argc, const char \* argv[]) {**

**setlocale(LC\_ALL, "UKR");**

**VIDNOSHENNIA v;**

**VIDNOSHENNIA\_MATR a, m;**

**v.adutuvne(A);**

**v.adutuvne(B);**

**if (v.adutuvne(A) && v.adutuvne(B)) {**

**cout << "Kompozucia\_A" << endl;**

**a.a\_kompozucia(A, B, res);**

**cout << "Riznucia\_A" << endl;**

**a.a\_riznucia(A, B, res);**

**cout << endl << "Obiednannia\_A " << endl;**

**a.a\_obiednannia(A, B, res);**

**cout << "Peretun\_A" << endl;**

**a.a\_peretun(A, B, res);**

**}**

**v.multuplikatuvne(P1);**

**v.multuplikatuvne(P2);**

**if (v.multuplikatuvne(P1) && v.multuplikatuvne(P2))**

**{**

**cout << "Kompozucia\_M " << endl;**

**m.m\_kompozucia(P1, P2, res);**

**cout << endl << "Obiednannia\_M " << endl;**

**m.m\_obiednannia(P1, P2, res);**

**cout << "Peretun\_M" << endl;**

**m.a\_peretun(P1, P2, res);**

**cout << "Riznucia\_M" << endl;**

**m.a\_riznucia(P1, P2, res);**

**}**

**return 0;**

**}**

**Скріншоти:**

**Висновок:** в ході цієї роботи я вивчив та практично ознайомився з основними алгебрами метризованих бінарних відношень та з властивостями і типами метризованих бінарних відношень.