

МІНІСТЕРСТВО ОСВІТИ І НАУКИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

Інститут комп'ютерних наук та інформаційних технологій

Кафедра систем штучного інтелекту



Лабораторна робота №3
з курсу “Дискретна математика ”

Виконав:
ст. гр. КН-110
Чорній Юрій

Викладач:
Мельникова Н.І.

Львів – 2018

Побудова матриці бінарного відношення

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

Завдання:

Варіант № 15

1. Чи є вірною рівність: $(A \times (B \cap C)) \cap ((A \cap B) \times C) = (A \times C) \cap (B \times B)$?

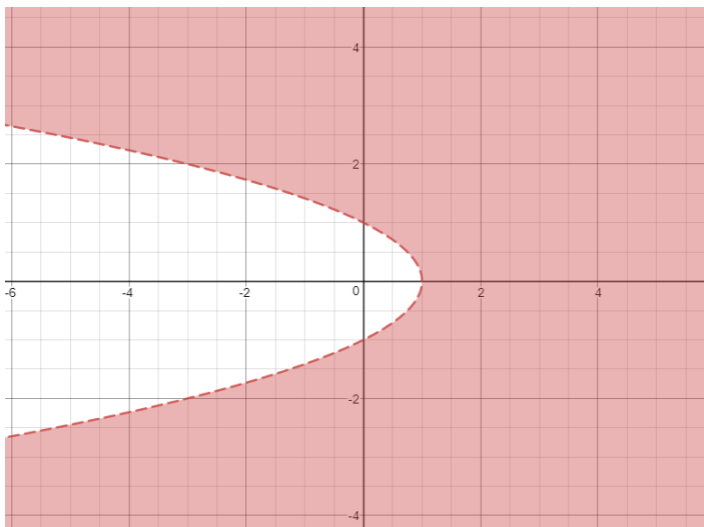
$$(Ax(B \cap C) \cap ((A \cap B)x C) = ((Ax B) \cap (Ax C)) \cap ((Ax C) \cap (Bx C) = (Ax B) \cap (Ax C) \cap (Bx C)$$

Ні, невірна.

2. Знайти матрицю відношення $R \subset M \times 2^M$, де $M = \{1, 2, 3\}$: $11 R = \{(x, y) \mid x \in M \text{ \& } y \subset M \text{ \& } y \leq x\}$.

$M \backslash 2^M$	$\{\emptyset\}$	$\{1\}$	$\{2\}$	$\{3\}$	$\{1, 2\}$	$\{1, 3\}$	$\{2, 3\}$	$\{1, 2, 3\}$
1	1	1	1	1	0	0	0	0
2	1	1	1	1	1	1	1	0
3	1	1	1	1	1	1	1	1

3. Зобразити відношення графічно $\alpha = \{(x, y) \mid (x, y) \in R^2 \text{ \& } x + y^2 - 1 > 0\}$; де R - множина дійсних чисел.

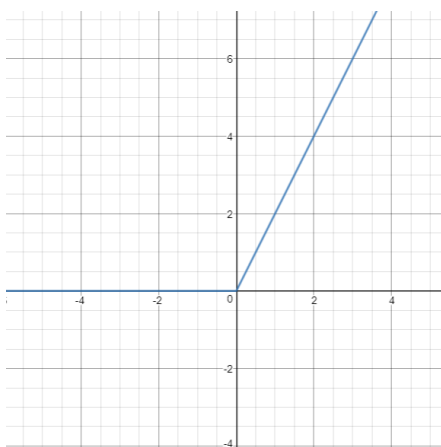


4. Навести приклад бінарного відношення $R \subset A \times A$, де $A = \{a, b, c, d, e\}$, яке є антирефлексивне, несиметричне, транзитивне, та побудувати його матрицю .

$$\alpha = \{(x, y) | (x, y) \in A^2 \text{ \& } x * y < x^2\}$$

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 \end{pmatrix}$$

5. Визначити множину (якщо це можливо), на якій дане відношення є: а) функціональним; б) бієктивним: $\alpha = \{(x, y) | (x, y) \in R^2 \text{ \& } y = x + |x|\}$.



А) $D: x \in \mathbb{R}, E: y \in [0; \infty)$

Б) $D: x \in [0; \infty), E: y \in [0; \infty)$

Код програми

```
#include <stdio.h>
#include <malloc.h>
#include <stdbool.h>
#include <math.h>

void numsInArr(int* arr, int size)
{
    for(int i = 0; i < size; i++)
    {
        scanf("%d", (arr+i));
    }
}
```

```

void printArrays(int*arr, int size)
{
    printf("\n");
    for(int i = 0; i < size; i++)
    {
        printf("%d\t",*(arr+i));
    }
    printf("\n");
}

void binary(int* arr1, int* arr2, int* binary, int size1, int size2)
{
    int counter = 0;
    for(int i = 0; i < size1; i++)
    {
        for(int k = 0; k < size2; k++)
        {
            int help1 = *(arr1+i);
            int help2 = *(arr2+k);
            if(*(arr1+i) + *(arr2+k) + 1 > 3)
            {

                *(binary + counter * 4 ) = 1;
                int help = *(binary + counter * 4 );
                counter++;
            }
            else
            {
                *(binary + counter *4) = 0;
            }
        }
    }
}

```

```

        int help = *(binary + counter * 4 );
        counter++;
    }
}

printf("\n\n");
counter = 0;
for(int i = 0; i < size1;i++)
{
    for(int k = 0; k < size2; k++)
    {
        int help1 = *(binary + counter * 4);
        printf("%d\t",*(binary + counter * 4));
        counter++;
    }
    printf("\n");
}

}

bool rexability(int* binary,int size2)
{

    bool rexability = true;
    for(int i = 0; i < size2; i++)
    {

        rexability = *(&(*(binary+i*4))+i*4) == 1;
        if(!rexability)
            return rexability;
    }
}

```

```

        return rexeability;
    }
    bool antirexeability(int* binary,int size2)
    {
        bool antirexeability = true;
        for(int i = 0; i < size2; i++)
        {
            antirexeability = *(&(*(binary+i*4))+i*4)== 0;
            if(!antirexeability)
                return antirexeability;
        }
        return antirexeability;
    }
    bool symetric(int* binary, int size1, int size2)
    {
        bool symetric = true;
        int tmparr[size1][size2];
        int count = 0;
        int squaresize;
        if(size1 != size2)
        {
            if(size1>size2)
                squaresize = size2;
            else
                squaresize = size1;
        }
        else
            squaresize = size1;
    }

```

```

    for(int i = 0; i < size1; i++)
    {
        for(int k = 0; k < size2; k++)
        {
            tmparr[i][k] = *(binary + count*4);
            count++;
        }
    }
    for(int i = 0; i < squaresize; i++)
    {
        for(int k = 0; k < squaresize; k++)
        {
            symetric = tmparr[i][k] == tmparr[k][i];
            if(!symetric)
            {
                return symetric;
            }
        }
    }
    return symetric;
}

bool antisymmetric(int* binary, int size1, int size2)
{
    bool antisymmetric = false;
    int tmparr[size1][size2];
    int count = 0;

```



```

int squaresize;
if(size1 != size2)
{
    if(size1>size2)
        squaresize = size2;
    else
        squaresize = size1;
}
else
    squaresize = size1;
for(int i = 0; i < size1; i++)
{
    for(int k = 0; k < size2;k++)
    {
        tmparr[i][k] = *(binary + count*4);
    }
}
for(int i = 0; i < squaresize; i++)
{
    for(int k = 0; k < squaresize; k++)
    {
        antisymmetric = tmparr[i][k] * tmparr[k][i] == 0;
        if(!antisymmetric)
        {
            return antisymmetric;
        }
    }
}
}

```

```

        return antisymetric;
    }

    bool transitiv(int* binary, int size1, int size2)
    {
        bool transitiv = transitiv;
        int tmparr[size1][size2];
        int count = 0;
        for(int i = 0; i < size1; i++)
        {
            for(int k = 0; k < size2; k++)
            {
                tmparr[i][k] = *(binary + count*4);

            }
        }
        for(int i = 0; i < size1; i++)
        {
            for(int k = 0; k < size2; k++)
            {
                for(int j = 0; j < size2; j++)
                {
                    transitiv = tmparr[i][k] == tmparr[k][j] == tmparr[i][j];
                    if(!transitiv)
                        return transitiv;
                }
            }
        }
        return transitiv;
    }

```

```

}

int main(int argc, char* argv[])
{
    int firstsize, secondsize;

    printf("Write a size of first matrix:\t");
    scanf("%d",&firstsize);
    printf("Write a size of second matrix:\t");
    scanf("%d",&secondsize);

    int* firstarr = malloc(firstsize * 4);
    int* secondarr = malloc(secondsize * 4);
    int* binaryarr = (int*)malloc(firstsize * secondsize * 4);
    printf("Put the numbers in first matrix\n");
    numsInArr(firstarr,firstsize);
    printf("Put the numbers in second matrix\n");
    numsInArr(secondarr,secondsize);
    printArrays(firstarr,firstsize);
    printArrays(secondarr,secondsize);

    binary(firstarr,secondarr,binaryarr,firstsize,secondsize);
    if(rexability(binaryarr,secondsize))
        printf("\nRexability");
    else if(antirexability(binaryarr, secondsize))
        printf("\nAntirexability");
    else
        printf("\nNot Rexability");

    if(symetric(binaryarr, firstsize,secondsize))
        printf("\nSymetric");

```

```
else
    printf("\nNot symetric");
if(antisymmetric(binaryarr, firstsize,secondsize))
    printf("\nAntisymmetric");

if(transitiv)
    printf("\nTransitiv");
else
    printf("Not transitiv");
free(firstarr);
free(secondarr);
free(binaryarr);
return 0;
}
```

Висновок я набув практичних вмінь та навичок при побудові матриць бінарних відношень та визначені їх типів.