**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**НАЦІОНАЛЬНОМУ УНІВЕРСИТЕТІ “ЛЬВІВСЬКА**

**ПОЛІТЕХНІКА”**

**Кафедра систем штучного інтелекту**

**Лабораторна робота № 3**

з дисципліни

«Дискретна математика»

**Виконав:**

студент групи КН-108

Левицький Богдан

**Викладач:**

Бойко Н.І.

Львів - 2018 р.

**Побудова матриці бінарного відношення**

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

**Завдання:**

**Варіант № 15**

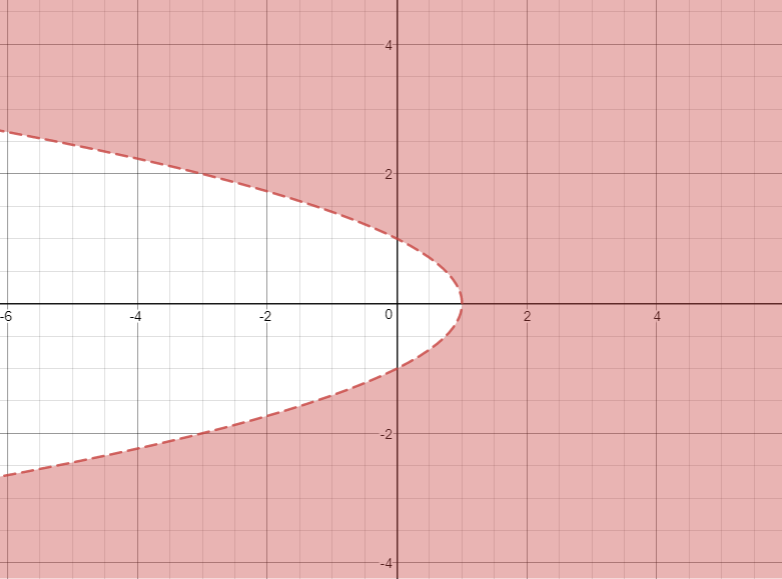
1. Чи є вірною рівність: (A×(B ∩C)) ∩ ((A∩ B) ×C) = (A×C) ∩ (B × B)?

Ні, невірна.

2. Знайти матрицю відношення R ⊂ M × , де M = {1,2,3}: 11 R = {(x, y) x ∈ M & y ⊂ M & y ≤ x}.

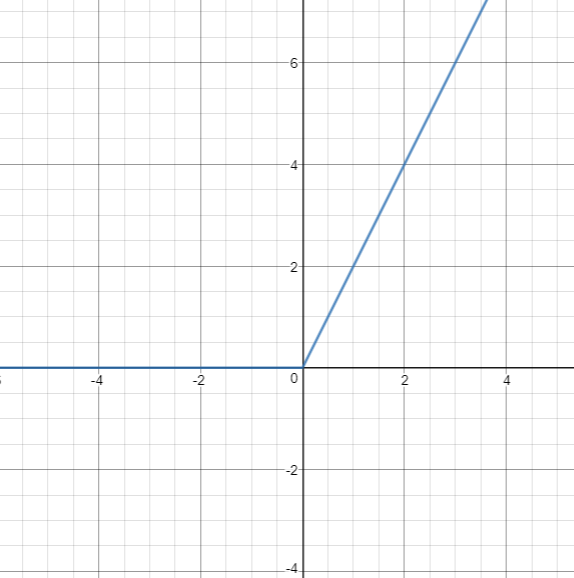
|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| M 2M | {Ø} | {1} | {2} | {3} | {1,2} | {1,3} | {2,3} | {1,2,3} |
| 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| 2 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| 3 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

3. Зобразити відношення графічно:, де R - множина дійсних чисел.



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

5. Визначити множину (якщо це можливо), на якій дане відношення є: а) функціональним; б) бієктивним: α = {(x, y)|(x, y)∈ & y = x + |x|}.



А)

Б)

**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 rexebility(int\* binary,int size2)

{

bool rexebility = true;

for(int i = 0; i < size2; i++)

{

rexebility = \*(&(\*(binary+i\*4))+i\*4) == 1;

if(!rexebility)

return rexebility;

}

return rexebility;

}

bool antirexebility(int\* binary,int size2)

{

bool antirexebility = true;

for(int i = 0; i < size2; i++)

{

antirexebility = \*(&(\*(binary+i\*4))+i\*4)== 0;

if(!antirexebility)

return antirexebility;

}

return antirexebility;

}

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 antisymetric(int\* binary, int size1, int size2)

{

bool antisymetric = 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++)

{

antisymetric = tmparr[i][k] \* tmparr[k][i] == 0;

if(!antisymetric)

{

return antisymetric;

}

}

}

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(rexebility(binaryarr,secondsize))

printf("\nRexebility");

else if(antirexebility(binaryarr, secondsize))

printf("\nAntirexebility");

else

printf("\nNot Rexebility");

if(symetric(binaryarr, firstsize,secondsize))

printf("\nSymetric");

else

printf("\nNot symetric");

if(antisymetric(binaryarr, firstsize,secondsize))

printf("\nAntisymetric");

if(transitiv)

printf("\nTransitiv");

else

printf("Not transitiv");

free(firstarr);

free(secondarr);

free(binaryarr);

return 0;

}