**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ НАЦІОНАЛЬНОМУ УНІВЕРСИТЕТІ “ЛЬВІВСЬКА ПОЛІТЕХНІКА”**

**Кафедра систем штучного інтелекту**

**Лабораторна робота №3**

**з дисципліни «Дискретна математика»**

**Виконав: студент групи КН-110**

**Кравець Данило**

**Викладач: Мельникова Н.І.**

**Львів – 2018 р.**

Варіант 14

Тема: Побудова матриці бінарного відношення

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

Завдання 1

A×(B ∩C ∪ D) = (A× B) ∩ (A×C) ∪ (A× D)

A×(B ∩C ∪ D)--> (АхВ)^(АхС)v(АxD)

(АхВ)^(АхС)v(АxD)=(A× B) ∩ (A×C) ∪ (A× D)

Завдання 2

R ⊂ 2^A×2^B

A ={1, 3}, B ={2, 4}

1. 2^A= {!0}{1}{3}{1,3}
2. 2^B= {!0}{2}{4}{2,4}

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 2^B\2^A | {!0} | {2} | {4} | {2,4} |
| 1 | 1 | 1 | 1 | 0 |
| 3 | 1 | 1 | 1 | 1 |

Завдання 3

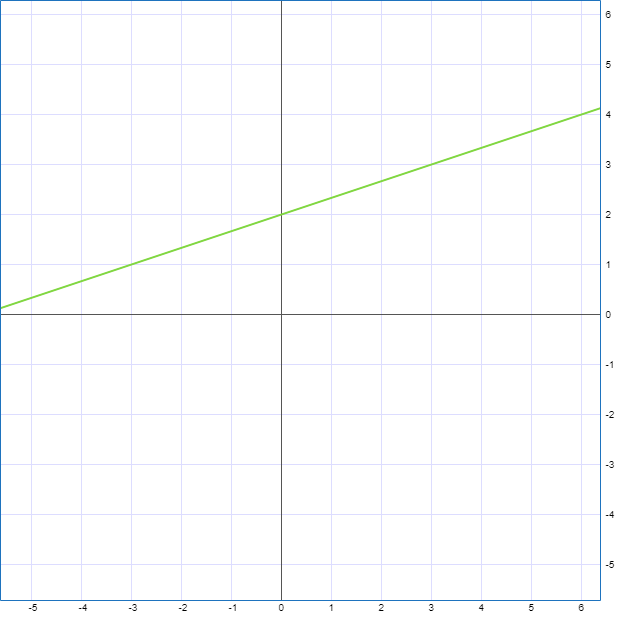
A={(x,y)|(x,y)ЄR^2&|6-3y|=x}

R- множина дійсних чисел

1)6-3y=x 2)6-3y=-x

y=2-(x/3) y=2+(x/3)

За умовою (6-3y)>=0, то графік є таким:



Завдання 4

R ⊂ A× A, де A ={a, b, c, d, е}

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 |
| 0 | 1 | 0 | 1 | 1 |
| 0 | 0 | 0 | 1 | 1 |

1) є рефлексивним тому що головна діагональ = 1

2)є симетричною тому що а(1.3)=а(3.1) і а(2.4)=а(4.2)

3)не є адже а(2.4)=1; а(4.5)=1;а(2.5)=0

4)не є антисиметричним адже а(4.5)=1 і а(5.4)=1

Завдання 5

А={(x,y)|(x,y)ЄR^2&|x|+|y|=4}

У першій чверті y=4-x;(x>0;y>0)

У другій чверті y=4+x;(x<0;y>0)

У третій чверті y=-4-x;(x<0;y<0)

У четвертій y=x-4;(x>0;y<0)

1)функціональне існує лише у точках (х=4;-4) та з одним у.

2)бієктивне існує лише у Y={!0} та з одним х.

Завдання 2.1

#include <stdio.h>

#include <math.h>

int main(){

int size;

printf("Enter size of arrays: ");

scanf("%d", &size);

printf("Enter array A\n");

int A[size];

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

printf("A[%d] = ", i);

scanf("%d", &A[i]);

}

int B[size];

printf("\nEnter array B\n");

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

printf("B[%d] = ", i);

scanf("%d", &B[i]);

}

printf("\nA = ");

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

printf("%d ", A[i]);

}

printf("\n");

printf("B = ");

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

printf("%d ", B[i]);

}

printf("\n");

/\* Generate the binary matrix \*/

int matr[size][size];

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

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

if(A[r]>2\*(B[c]))

matr[r][c] = 1;

else

matr[r][c] = 0;

}

}

/\* Print generated binary matrix \*/

printf("\n");

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

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

printf("%d ", matr[r][c]);

if(c==size-1)

printf("\n");

}

}

/\* Generate the inverse matrix \*/

int antimatr[size][size];

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

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

antimatr[r][c] = matr[c][r];

}

}

/\* Check for 0 \*/

int diagonal7 = 0;

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

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

if(matr[r][c]== 1)

diagonal7 = 1;

}

}

/\* Check for reflexivity \*/

printf("\n");

int diagonal1=1;

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

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

diagonal1=0;

}

if(diagonal1==1)

printf("This relation is reflexive.\n");

else if(diagonal1==0)

printf("This relation is not reflexive.\n");

/\* Check for antireflexivity \*/

int diagonal2=0;

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

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

diagonal2=1;

}

if(diagonal2==0)

printf("This relation is antireflexive.\n");

else if(diagonal2==1)

printf("This relation is not antireflexive.\n");

/\* Check for symmetry \*/

int diagonal3=1;

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

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

if((r!=c) && (antimatr[r][c]!=matr[r][c]))

diagonal3 = 0;

}

}

if(diagonal3==1)

printf("This relation is symmetric\n");

else if(diagonal3==0)

printf("This relation is not symmetric\n");

/\* Check for antisymmetry \*/

int diagonal4=0;

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

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

if((r!=c) && (antimatr[r][c]==matr[r][c]))

diagonal4 = 1;

}

}

if(diagonal4==0)

printf("This relation is antisymmetric\n");

else if(diagonal4==1)

printf("This relation is not antisymmetric\n");

/\* Check for transitivity \*/

int Asize = pow(size, 3);

int Apoint[Asize];

int ai=0;

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

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

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

if((i!=j && i!=k && j!=k) && (matr[i][j] == 0 || matr[j][k] == 0))

Apoint[ai]=1;

else if((i!=j && i!=k && j!=k) && (matr[i][j] == 1 && matr[j][k] == 1 && matr[i][k] == 1))

Apoint[ai]=1;

else if ((i!=j && i!=k && j!=k) && (matr[i][j] == 1 && matr[j][k] == 1 && matr[i][k] == 0))

Apoint[ai]=0;

else

Apoint[ai]=1;

ai++;

}

}

}

int diagonal5 = 1;

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

if(Apoint[i]==0)

diagonal5=0;

}

if(diagonal7 == 0)

diagonal5 = 0;

if(diagonal5==1)

printf("This relation is transitive\n");

else if(diagonal5==0)

printf("This relation is not transitive\n");

/\* Check for antitransitivity \*/

int Bsize = pow(size, 3);

int Bpoint[Bsize];

int bi=0;

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

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

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

if((i!=j && i!=k && j!=k) && (matr[i][j] == 0 || matr[j][k] == 0))

Bpoint[bi]=1;

else if((i!=j && i!=k && j!=k) && (matr[i][j] == 1 && matr[j][k] == 1 && matr[i][k] == 1))

Bpoint[bi]=0;

else if ((i!=j && i!=k && j!=k) && (matr[i][j] == 1 && matr[j][k] == 1 && matr[i][k] == 0))

Bpoint[bi]=1;

else

Bpoint[bi]=1;

bi++;

}

}

}

int diagonal6 = 1;

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

if(Bpoint[i]==0)

diagonal6=0;

}

if(diagonal7 == 0 || diagonal5==1)

diagonal6 = 0;

if(diagonal6==1)

printf("This relation is antitransitive\n");

else if(diagonal6==0)

printf("This relation is not antitransitive\n");

printf("\n");

}