**МИНОБРНАУКИ РОССИИ**

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ОБРАЗОВАНИЯ

**«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ**

**ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В.Г.ШУХОВА»**

**(БГТУ им. В.Г. Шухова)**

Кафедра программного обеспечения вычислительной техники и автоматизированных систем

дисциплина: Дискретная математика

Лабораторная работа №3.4

тема: «Упорядоченные множества»

|  |  |
| --- | --- |
|  | Выполнил: ст. группы ВТ-22  Макаров Даниил Сергеевич  Проверил: Рязанов Ю.Д. |

Белгород

2018

**Цель занятия**: изучить упорядоченные множества, алгоритм топологической сортировки, научиться представлять множества диаграммами Хассе, находить минимальные (максимальные) и наименьшие (наибольшие) элементы упорядоченного множества.

**Вариант 7**

А={(a,b) | ay < by}

**Задания**

Даны множества точек на плоскости М1 (рис. 3.23), М2 (рис. 3.24) и отношение порядка (табл. 3.5). Для определения отношения на множестве точек примем следующие обозначения: ax — абсцисса точки a; ay — ордината a. На рис. 3.23 координаты правой верхней точки считать (1,1). На рис. 3.24 координаты самой верхней точки считать (0,2), а координаты самой правой точки считать (2,0).

Y

X

Рис. 3.24. Множество *М*2

Y

X

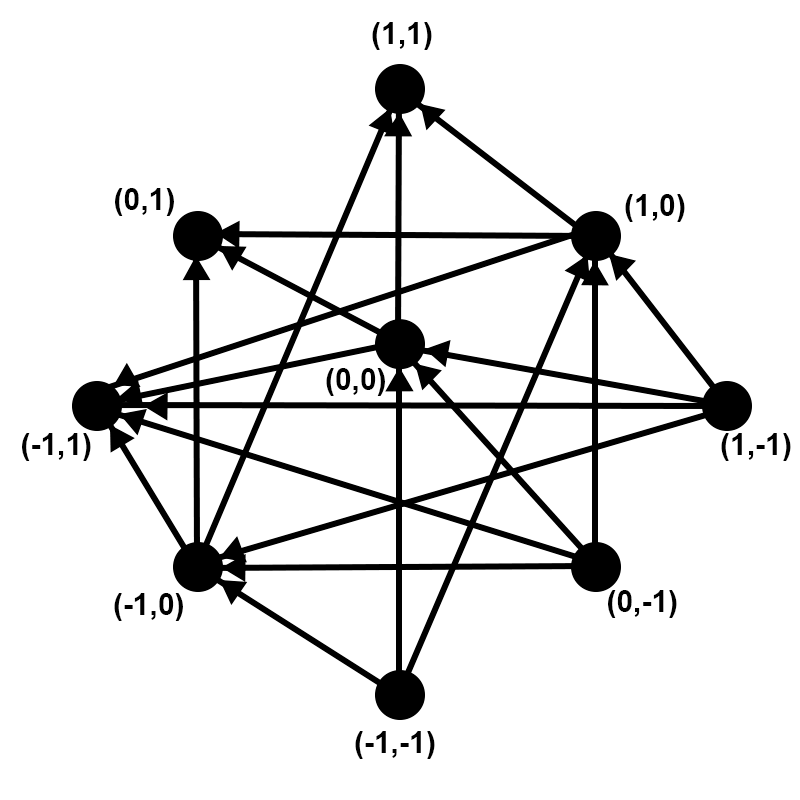
Рис. 3.23. Множество *М*1

1. Написать программы, формирующие матрицы отношения порядка, в соответствии с вариантом задания (табл. 3.5), на множествах М1 и М2.

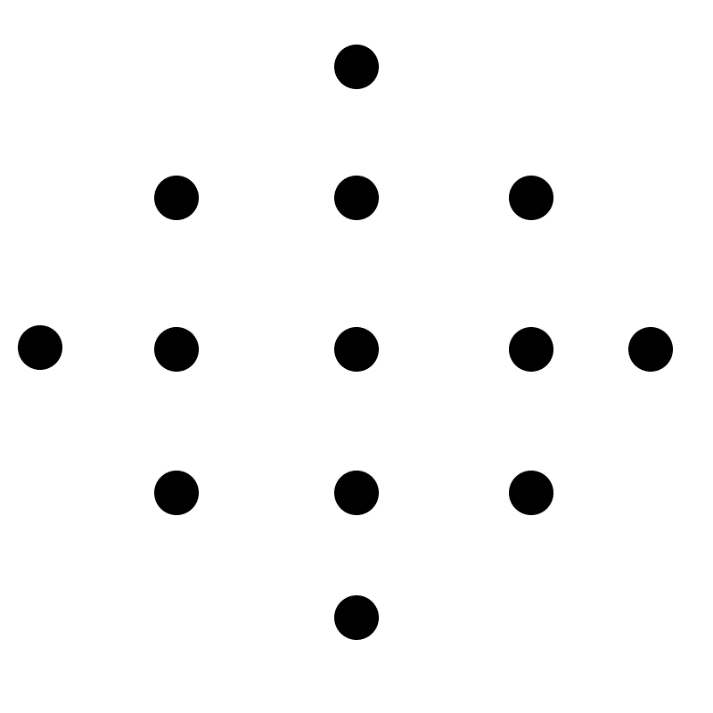
2. Написать программы, формирующие матрицы отношения доминирования по матрицам отношения порядка.

3. Написать программу, реализующую алгоритм топологической сортировки по матрице отношения доминирования.

4. Изобразить диаграмму Хассе отношения доминирования на множествах М1 и М2.



*Диаграмма Хассе отношения доминирования на множестве М1*

****

*Диаграмма Хассе отношения доминирования на множестве М2*

5. Найти минимальные и максимальные элементы множеств М1 и М2.

|  |  |  |
| --- | --- | --- |
|  | Максимальные элементы | Минимальныe элементы |
| М1 |  |  |
| М2 |  |  |

6. Найти, если существуют, наименьший и наибольший элементы множеств М1 и М2.

|  |  |  |
| --- | --- | --- |
|  | Наибольший элемент | Наименьший элемент |
| М1 |  |  |
| М2 |  |  |

Матрица отношения на множестве М1:

0 1 1 0 1 1 0 1 1

0 0 1 0 0 1 0 0 1

0 0 0 0 0 0 0 0 0

0 1 1 0 1 1 0 1 1

0 0 1 0 0 1 0 0 1

0 0 0 0 0 0 0 0 0

0 1 1 0 1 1 0 1 1

0 0 1 0 0 1 0 0 1

0 0 0 0 0 0 0 0 0

Матрица отношения доминирования по матрице отношения порядка на множестве М1:

0 1 0 0 1 0 0 1 0

0 0 1 0 0 1 0 0 1

0 0 0 0 0 0 0 0 0

0 1 1 0 1 0 0 1 0

0 0 1 0 0 1 0 0 1

0 0 0 0 0 0 0 0 0

0 1 1 0 1 1 0 1 0

0 0 1 0 0 1 0 0 1

0 0 0 0 0 0 0 0 0

Матрица отношения на множестве М2:

0 1 1 0 1 1 0 1 1 1 0 1 1

0 0 1 0 0 1 0 0 1 0 0 0 1

0 0 0 0 0 0 0 0 0 0 0 0 1

0 1 1 0 1 1 0 1 1 1 0 1 1

0 0 1 0 0 1 0 0 1 0 0 0 1

0 0 0 0 0 0 0 0 0 0 0 0 1

0 1 1 0 1 1 0 1 1 1 0 1 1

0 0 1 0 0 1 0 0 1 0 0 0 1

0 0 0 0 0 0 0 0 0 0 0 0 1

0 0 1 0 0 1 0 0 1 0 0 0 1

1 1 1 1 1 1 1 1 1 1 0 1 1

0 0 1 0 0 1 0 0 1 0 0 0 1

0 0 0 0 0 0 0 0 0 0 0 0 0

Матрица отношения доминирования по матрице отношения порядка на множестве М2:

0 1 0 0 1 0 0 1 0 1 0 1 0

0 0 1 0 0 1 0 0 1 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 1

0 1 1 0 1 0 0 1 0 1 0 1 0

0 0 1 0 0 1 0 0 1 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 1

0 1 1 0 1 1 0 1 0 1 0 1 0

0 0 1 0 0 1 0 0 1 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 1

0 0 1 0 0 1 0 0 1 0 0 0 1

1 1 1 1 1 1 1 1 1 1 0 1 0

0 0 1 0 0 1 0 0 1 0 0 0 1

0 0 0 0 0 0 0 0 0 0 0 0 0

#include <stdio.h>

#include <stdlib.h>

#include <locale.h>

struct dot{

int x;

int y;

};

void create\_rel(struct dot\* M, int n, int \*\* A) {

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

for (int j=0; j<n; j++)

A[i][j]=(M[i].y<M[j].y);

}

int check\_anti\_sym (int \*\*R, int N) {

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

for (int j=0; j<N; j++)

if((i != j) && (R[i][j]\*R[j][i]))

return 0;

return 1;

}

int check\_transitive (int \*\*R, int N) {

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

for (int k=0; k<N; k++)

if (R[i][k])

for (int j=0; j<N; j++)

if (R[k][j] && (R[i][j] == 0))

return 0;

return 1;

}

int check\_order(int \*\*R, int N) {

int tmp1, tmp2;

tmp1 = check\_anti\_sym(R,N);

tmp2 = check\_transitive(R, N);

return (tmp1\*tmp2);

}

int check (int \*\*A, int x, int y) {

for (int i= x+1; i<y; i++)

if (A[x][i] && A[i][y])

return 1;

return 0;

}

void create\_domin\_rel (int \*\*A, int N) {

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

for (int j=0; j<N; j++)

if (A[i][j])

if (i == j)

A[i][j] = 0;

else

if (check (A, i, j))

A[i][j] = 0;

}

void print\_matr (int \*\*A, int N) {

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

for (int j=0; j<N; j++)

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

printf ("\n");

}

}

void mem\_free (int \*\*A, int N) {

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

free (A[i]);

free (A);

}

void print\_a (int \*\*A, int N) {

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

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

printf ("\n\n\n");

}

int check\_nut (int \*A, int N) {

int i;

for(i=0;i<N && A[i]<0;i++);

return (i<N);

}

void matr\_subs\_mass (int \*\*A, int \*B, int N, int i) {

for (int j=0; j<N; j++)

B[j] = B[j] - A[i][j];

}

void create\_top\_sort (int \*\*A, int N, struct dot \*M) {

int i, k=0;

int \*B = (int\*)calloc(N,sizeof(int));

for (i=0; i<N; i++) {

B[i] = 0;

for (int j=0; j<N; j++)

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

}

do {

printf("%i lvl: ", k);

for (i=0; i<N; i++)

if (B[i] == 0) {

printf("(%i, %i)", M[i].x, M[i].y);

B[i] = -k-1;

}

printf("\n");

for (i=0; i<N; i++)

if (B[i] == -k-1)

matr\_subs\_mass(A, B, N, i);

k++;

}

while (check\_nut(B,N) && (k<N));

}

int main(){

int n1=9, n2=13;

struct dot \*M1 = ((struct dot\*) calloc(n1, sizeof(struct dot)));

struct dot \*M2 = ((struct dot\*) calloc(n2, sizeof(struct dot)));

int XY[3]={-1, 0, 1};

int k=0;

for (int i=0; i<3; i++)

for (int j=0; j<3; j++){

M1[k].x = M2[k].x = XY[i];

M1[k].y = M2[k].y = XY[j];

k++;

}

M2[9].x -= 2; M2[9].y = 0;

M2[10].x = 0; M2[10].y -= 2;

M2[11].x = 2; M2[11].y = 0;

M2[12].x = 0; M2[12].y = 2;

int\*\* A1 = (int\*\*) calloc(n1, sizeof(int\*));

for (int i=0; i<n1; i++)

A1[i] = (int\*) calloc(n1, sizeof(int));

printf("Relationship matrix on set M1:\n\n");

create\_rel (M1, n1, A1);

print\_matr (A1, n1);

if (check\_order(A1, n1)) {

printf("check\_order\n");

create\_domin\_rel (A1, n1);

printf("\nThe dominance relation matrix in the matrix of an check\_order relation on the set M1:\n\n");

print\_matr(A1, n1);

printf("\n");

create\_top\_sort (A1, n1, M1);

}

else{

printf("Not check\_order\n");

}

int\*\* A2 = (int\*\*) calloc (n2, sizeof(int\*));

for (int i=0; i<n2; i++)

A2[i] = (int\*) calloc (n2, sizeof(int));

create\_rel (M2, n2, A2);

printf ("\nRelationship matrix on set M2:\n\n");

print\_matr(A2,n2);

if (check\_order(A2,n2)) {

create\_domin\_rel(A2,n2);

printf("\nThe dominance relation matrix in the matrix of an check\_order relation on the set M2:\n\n");

print\_matr(A2,n2);

printf("\n");

create\_top\_sort(A2,n2,M2);

}

else{

printf("Not check\_order\n");

}

mem\_free(A1,n1);

mem\_free(A2,n2);

getchar();

return 0;

}