**МИНОБРНАУКИ РОССИИ**

ФЕДЕРАЛЬНОЕ ГОСУДАРСТЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕНЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В.Г.Шухова»**

**(БГТУ им. В.Г.Шухова)**

Кафедра программного обеспечения вычислительной техники и автоматизированных систем

Дисциплина: Дискретная математика

Лабораторная работа № 4

Тема: «Упорядоченные множества»

Выполнил:

Студент группы ВТ-22

Богатырев В.В.

Белгород 2018

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

**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).

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

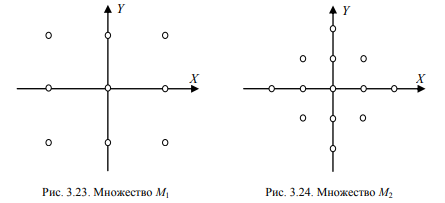
2. Написать программы, формирующие матрицы отношения доминирования по матрицам отношения порядка.

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

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

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

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



#include <stdio.h>

#include <stdlib.h>

typedef struct point{

int x;

int y;

}point;

void WriteMatr(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");

}

printf("\n");

}

void getmem(int \*\*\*a, int n)

{

\*a = calloc(n,sizeof(int\*));

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

(\*a)[i] = calloc(n,sizeof(int));

}

void GetMatr(int \*\*a, point \*M, int n)

{

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

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

a[i][j] = (M[i].x <= M[j].y) && (M[j].x == M[i].y);

}

void M1Build(point \*a)

{

int k = 0;

for (int i = -1; i < 2; i++)

for (int j = -1; j < 2; j++)

{

a[k].x = i;

a[k].y = j;

k++;

}

}

void M2Build(point \*a)

{

int k = 0;

for (int i = -2; i < 3; i++)

for (int j = abs(i) - 2; j <= abs(abs(i) -2); j++)

{

a[k].x = i;

a[k].y = j;

k++;

}

}

void dominational(int \*\*a, int \*\*b, int n)

{

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

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

if (a[i][j] == 1)

{

int t,z;

for (z = i+1, t = 0; (z < j && t == 0); z++)

t = a[i][z] && a[z][j];

b[i][j] = !t;

}

}

int check\_otr(int \*a, int n)

{

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

if (a[i] >= 0)

return 0;

return 1;

}

void sort(int \*\*a, int n)

{

int \*w = calloc(n,sizeof(int));

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

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

w[i] += a[j][i];

int k = 0;

while (!check\_otr(w,n))

{

printf("%i level > ", k);

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

if(w[i] == 0)

{

printf("%i ",i+1);

w[i] = -(k + 1);

}

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

if (w[i] == -(k+1))

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

w[j] -= a[i][j];

k++;

printf("\n");

}

}

int main()

{

printf("M1 :\n");

int \*\*a, \*\*a1;

point \*M1 = calloc(9,sizeof(point));

M1Build(M1);

getmem(&a,9);

getmem(&a1,9);

GetMatr(a,M1,9);

dominational(a,a1,9);

WriteMatr(a1,9);

sort(a1,9);

printf("\nM2 :\n");

int \*\*b, \*\*b1;

point \*M2 = calloc(13,sizeof(point));

M2Build(M2);

getmem(&b,13);

getmem(&b1,13);

GetMatr(b,M2,13);

dominational(b,b1,13);

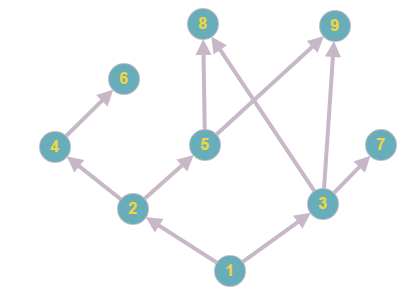
WriteMatr(b1,13);

sort(b1,13);

}

|  |  |
| --- | --- |
| M1 | M2 |
|  |  |
|  |  |
|  |  |

**M1:**



**Для M2:**

