МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

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

Лабораторная работа №3.4

по дисциплине: «Дискретная математика»

по теме: **Упорядоченные множества**

Выполнил/а: ст. группы ПВ-231

Чупахина София Александровна

Проверил: Рязанов Юрий Дмитриевич

Белгород, 2024

Вариант 6

$A = \{(a, b) \Big| a_x - b_x < a_y - b_y\}$

Содержание

Задание 1 **3**

Задание 2 **6**

Задание 3 **8**

Задание 4 **12**

Задание 5 **13**

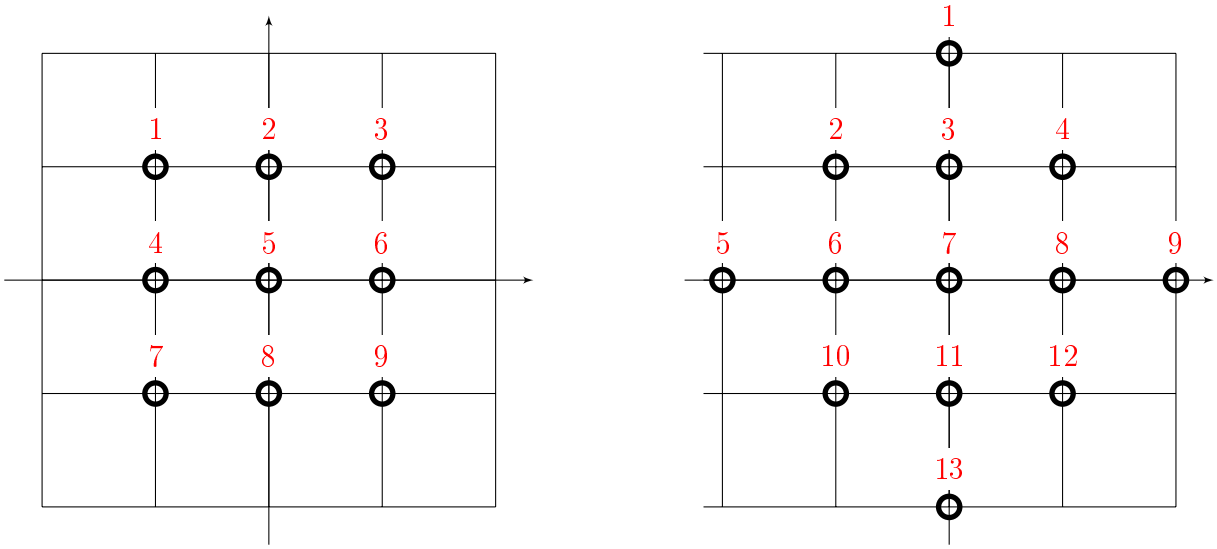
Задание 6 **14**

Вывод **15**

Задание 1

Текст задания: Написать программы, формирующие матрицы отношений в соответствии с вариантом задания (табл. 3), на множествах 1 и 2. Определить свойства отношения. Если отношение не обладает свойством порядка, то выполнить следующий вариант заданий.

Библиотека, разработанная ранее для работы с отношениями, предполагала, что отношения будут строиться на отрезке множества натуральных чисел, то есть на множестве M с элементами — числами от 1 до $|M|$. В нашем же случае отношения будут строиться на множествах точек. Вместо того, чтобы переписывать библиотеку дл работы с произвольным типом, в том числе для структуры «точка», присвоим каждой точке из исходных множеств порядковый номер, и создадим отношения уже на множестве порядковых номеров точек.



Затем в файле C зададим структуру «точка», имеющую два поля с целыми значениями: координаты x и y. Объявив как константы мощности множеств M_1 и M_2 (9 и 13 соответственно), создадим одноименные массивы, хранящие точки в том же порядке, в котором мы пронумеровали их на рисунке выше. Далее функция createForVariant6, принимающая как аргументы указатель на начало такого массива точек и его длину, перебирает все возможные пары точек в этом массиве, и если их координаты отвечают условию, заданному для варианта 6, то в отношение добавляется пара из порядковых номеров этих точек. Результирующее отношение и возвращается функцией. В теле функции main() остается только сохранить результат выполнения функции для массивов M1 и M2 (и их длин) и вывести матрицы полученных отношений на экран.

```
1 #include "ДИСКРЕТКА../../ ЛАБА
    3.1/bin_relations/bin_relation_definition_input_output.c"
2 #include "ДИСКРЕТКА../../ ЛАБА
    3.1/bin_relations/bin_relations_operations.c"
3 #include "ДИСКРЕТКА../../ ЛАБА
    3.1/bin_relations/bin_relations_properties.c"
4 #include <stdbool.h>
5
6 typedef struct {
```

```
    int x;
    int y;
} Point;

#define M1_LEN 9
#define M2_LEN 13

Point M1[M1_LEN] = {
        {-1, 1}, {0, 1}, {1, 1},
        {-1, 0}, {0, 0}, {1, 0},
        {-1, -1}, {0, -1}, {1, -1}
};
Point M2[M2_LEN] = {
        {0, 2},
        {-1, 1}, {0, 1}, {1, 1},
        {-2, 0}, {-1, 0}, {0, 0}, {1, 0},{2, 0},
        {-1, -1}, {0, -1}, {1, -1},
        {0, -2}
};

bin_relation createForVariant6(Point *set, int set_len) {
    bin_relation result = bin_relation_createEmpty(set_len);
    for (int i = 1; i <= set_len; i++) {
        Point first_point = set[i];
        for (int j = 1; j <= set_len; j++) {
            Point second_point = (set[j]);
            if (first_point.x - second_point.x < first_point.y -
   second_point.y) {
                bin_relation_changeValue(&result, i, j, 1);
            }
        }
    }
    return result;
}

int main () {
    bin_relation A1 = createForVariant6(M1, M1_LEN);
    bin_relation A2 = createForVariant6(M2, M2_LEN);

    printf("Matrix of relation A1:\n");
    bin_relation_matrixPrint(A1);
    printf("\nMatrix of relation A2:\n");
    bin_relation_matrixPrint(A2);

    printf("\nProperties of relation A1:\n");
    bin_relation_print_all_properties(A1);
    printf("\nProperties of relation A2:\n");
```

```
53      bin_relation_print_all_properties(A2);
54  }
```

ordered_sets/ordered_sets_main.c

```
Matrix of relation A1:
0 1 0 1 1 1 1 1 1
0 0 0 0 1 0 1 1 0
0 1 0 1 1 1 1 1 1
0 0 0 0 1 0 1 1 0
0 0 0 0 0 0 0 1 0
0 0 0 0 1 0 1 1 0
0 0 0 0 0 0 0 1 0
0 0 0 0 0 0 0 0 0
0 0 0 0 1 0 1 1 0
```

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

```
Matrix of relation A2:
0 1 1 0 1 1 1 1 1 1 1 1 1
0 0 1 0 0 1 1 1 1 1 1 1 1
0 0 0 0 0 0 1 1 0 1 1 1 0
0 1 1 0 1 1 1 1 1 1 1 1 1
0 0 1 0 0 1 1 1 1 1 1 1 1
0 0 0 0 0 0 1 1 0 1 1 1 0
0 0 0 0 0 0 0 1 0 0 1 1 0
0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 1 1 0 1 1 1 0
0 0 0 0 0 0 1 0 0 1 1 0
0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 1 1 0 1 1 1 0
```

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

Нетрудно заметить, что также для каждого из отношений вызывается функция, выводящая на экран основные и производные свойства этих отношений. В этот раз мы работаем только с матрицами отношений, в то время как доказательство наличия у отношения таких ключевых для данной работы свойств, как транзитивность и антитранзитивность, лишь по матрицам весьма затруднительно. Поэтому прибегнем к помощи ранее написанной функции.

```
Properties of relation A1:
        Simple properties of a relation:
This relation is not reflexive, because the pair (1, 1) doesn't belong it.
This relation is antireflexive.
This relation is not symmetrical, because the pair (1, 2) belongs it and the pair (2, 1) doesn't.
This relation is antisymmetrical.
This relation is transitive.
This relation is not antitransitive, because both pairs (1, 2) and (2, 5) belong it, and also the pair (1, 5) does.
This relation is not full, because none of the pairs (2, 2) and (2, 2) belong it.
        Derived properties of a relation:
This relation has the order property.
This relation has the strict order property.
```

```
Properties of relation A2:
        Simple properties of a relation:
This relation is not reflexive, because the pair (1, 1) doesn't belong it.
This relation is antireflexive.
This relation is not symmetrical, because the pair (1, 2) belongs it and the pair (2, 1) doesn't.
This relation is antisymmetrical.
This relation is transitive.
This relation is not antitransitive, because both pairs (1, 2) and (2, 3) belong it, and also the pair (1, 3) does.
This relation is not full, because none of the pairs (2, 2) and (2, 2) belong it.
        Derived properties of a relation:
This relation has the order property.
This relation has the strict order property.
```

Оба полученных отношения не рефлексивны, но антирефлексивны, не симметричны, но антисимметричны, транзитивны, но не антитразитивны, не полны. Вследствие своей антисимметричности и транзитивности они обладают свойством порядка, а вследствие своей антирефлексивности также свойством строгого порядка.

Поскольку оба отношения обладают свойством порядка, это позволяет нам перейти к следующему заданию.

Задание 2

Текст задания: Написать программы, формирующие матрицы отношения доминирования по матрицам отношения порядка.

Отношение доминирования может быть получено над любым отношением порядка $\leq$. В нем, помимо стандартных свойств отношения порядка, соблюдается условие, что для любых x и y, если $x \leq y$, не существует такого z, что $x \leq z \leq y$, что позволяет впоследствии выстроить между элементами некую иерархию (это отражено в названии).

Также над любым отношением порядка можно получить отношение строгого порядка. Если отношение порядка по определению обладает свойствами антисимметричности и транзитивности, то отношение строгого порядка $<$ обладает также и свойством антирефлексивности.

Значит, его можно получить из отношения порядка, вычтя из него главную диагональ: $<= \leq -I$ (как мы помним, как I обозначается тождественное отношение, $I = \{(x, x) \Big| x \in A\}$).

Отношение доминирования же находится как разность самого отношения строгого порядка и его квадрата: $\triangleleft = < - <^2$

В нашем случае оба полученных отношения уже являются отношениями строгого порядка, но чтобы сделать наш код более универсальным и применимым для других отношений в дальнейшем (если это будет необходимо), позаботимся о том, чтобы функция преобразования отношения порядка в отношение строгого порядка также присутствовала.

```
ДИСКРЕТКАЛАБАДИСКРЕТКАЛАБАДИСКРЕТКАЛАБА
bin_relation createStrictOrderRelation(bin_relation A) {
    for (int i = 1; i <= A.max_value; i++) {
        bin_relation_changeValue(&A, i, i, 0);
    }
    return A;
}

bin_relation createDominationRelation(bin_relation A) {
    bin_relation strict_A = createStrictOrderRelation(A);
    return bin_relation_difference(strict_A,
   bin_relation_degree(strict_A, 2));
}

int main () {
    bin_relation A1 = createForVariant6(M1, M1_LEN);
    bin_relation A2 = createForVariant6(M2, M2_LEN);

    bin_relation dom_A1 = createDominationRelation(A1);
    printf("\nMatrix of domination relation over A1:\n");
    bin_relation_matrixPrint(dom_A1);
    bin_relation dom_A2 = createDominationRelation(A2);
    printf("\nMatrix of domination relation over A2:\n");
    bin_relation_matrixPrint(dom_A2);
}
```

ordered_sets/ordered_sets_main.c

Матрицы полученных отношений доминирования будут иметь следующий вид:

```
Matrix of domination relation over A1:
0 1 0 1 0 1 0 0 1
0 0 0 0 1 0 1 0 0
0 1 0 1 0 1 0 0 1
0 0 0 0 1 0 1 0 0
0 0 0 0 0 0 0 1 0
0 0 0 0 1 0 1 0 0
0 0 0 0 0 0 0 1 0
0 0 0 0 0 0 0 0 0
0 0 0 0 1 0 1 0 0
```

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

```
Matrix of domination relation over A2:
0 1 0 0 1 0 0 0 0 0 0 0 0
0 0 1 0 0 1 0 0 1 0 0 0 1
0 0 0 0 0 0 1 0 0 1 0 0 0
0 1 0 0 1 0 0 0 0 0 0 0 0
0 0 1 0 0 1 0 0 1 0 0 0 1
0 0 0 0 0 0 1 0 0 1 0 0 0
0 0 0 0 0 0 0 1 0 0 1 1 0
0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 1 0 0 1 0 0 0
0 0 0 0 0 0 0 1 0 0 1 1 0
0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 1 0 0 1 0 0 0
```

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

Задание 3

Текст задания: Написать программу, реализующую алгоритм топологической сортировки по матрице отношения доминирования.

Принимаясь за реализацию этого алгоритма в программном виде, стоит задуматься: будет ли храниться где-то полученный результат — сортировка элементов множества по уровням? Конечно, можно ограничиться обычным выводом, без сохранения результата. Однако подумав чуть дольше, мы можем провести аналогию между этой задачей и задачей хранения классов эквивалентности в прошлой лабораторной работе. Нам также необходимо разбить множество на непересекающиеся подмножества, при этом такие, что их объединение дает исходное множество. Ведь очевидно, что каждый элемент принадлежит хотя бы какому-то уровню и при этом не

может принадлежать двум уровням сразу.

Таким образом, мы можем скопировать в программу структуру splitting и функции инициализации и вывода для нее без значительных изменений и переименований (разбиение множества на уровни, а не на классы эквивалентности, все еще остается разбиением). При создании структуры требуется параметр max_value, мощность множества, разбиение которого будет храниться. Выделяется память для хранения массива целых чисел длины max_value, и все его элементы инициализируются нулями. В поле display будет храниться указатель на этот массив, а в поле max_value — одноименный входной параметр. При печати используется следующий алгоритм: сначала производится проход по массиву display, в ходе которого находится максимальный номер множества-элемента разбиения, max_set. Затем в теле цикла for перебираются номера таких множеств от 1 до max_set, и для каждого номера совершается проход по массиву display. Если на некоторой его позиции хранится текущий номер, значит, соответствующий элемент исходного множества входит в множество-элемент разбиения. Этот элемент выводится на экран. До прохода по массиву выводится открывающая скобка «{», после прохода по массиву — закрывающая «}», для обособления отдельных множеств-элементов разбиения. Перед основной частью алгоритма и после нее будут выводиться квадратные скобки; квадратные — потому что в нашем случае некорректно называть результат множеством. Порядок множество внутри разбиения имеет значение: в каждой последовательности элементы первого множества относятся к самому низкому, первому уровню, элементы последнего множества — к самому высокому уровню.

Алгоритм же сортировки мы реализуем в функции getLevelsSet, которая и возвращает структуру типа splitting. Вначале создается разбиение l для хранения уровней, массив w_array для хранения суммы столбцов матрицы отношения, задается начальное количество уровней, равное 1, и стартовое значение i = -1. С помощью двух вложенных циклов для каждого столбца матрицы считается сумма элементов и сохраняется на соответствующей позиции массива w_array. Затем инициализируется переменная flag, и ее значение определяется в цикле, после прохода по массиву в котором становится понятно, есть ли в нем хотя бы один неотрицательный элемент. Если есть, запускается основная часть алгоритма: поиск в массиве w_array нулей и замена их на i; поиск i в массиве w_array и вычитание из него строк матрицы с номерами, соответствующими индексам элементов i в массиве; уменьшение i на 1, увеличение количества уровней на 1, изменение значения flag (выполнение алгоритма продолжается, пока flag истинна и в массиве w_array есть хотя бы один неотрицательный элемент).

```
ДИСКРЕТКАЛАБАДИСКРЕТКАЛАБАДИСКРЕТКАЛАБА
typedef struct {
    int *display;
    int max_value;
} splitting;

splitting splitting_createEmpty(int max_value) {
    int *display = malloc(sizeof(int) * max_value);
    for (int cur_el = 0; cur_el < max_value; cur_el++) {
        display[cur_el] = 0;
    }
    return (splitting) {display, max_value};
}

```

```
15 void splitting_print(splitting s) {
16     printf("[");
17     int max_set = 0;
18     for (int cur_x = 1; cur_x <= s.max_value; cur_x ++) {
19         if (s.display[cur_x-1] > max_set) {
20             max_set = s.display[cur_x-1];
21         }
22     }
23     for (int cur_set = 1; cur_set <= max_set; cur_set++) {
24         printf("{");
25         for (int cur_y = 1; cur_y <= s.max_value; cur_y++) {
26             if (s.display[cur_y-1] == cur_set) {
27                 printf("%d, ", cur_y);
28             }
29         }
30         printf("\b\b}, ");
31     }
32     printf("\b\b]");
33 }
34
35 splitting getTopologicalSorting (bin_relation A) {
36     splitting l = splitting_createEmpty(A.max_value);
37     int levels_amount = 1;
38     int w_array[A.max_value];
39     int i = -1;
40
41     for (int cur_col_ind = 1; cur_col_ind <= A.max_value;
       cur_col_ind++) {
42         int cur_sum = 0;
43         for (int cur_row_ind = 1; cur_row_ind <= A.max_value;
       cur_row_ind++) {
44             cur_sum += bin_relation_getValue(A, cur_row_ind,
       cur_col_ind);
45         }
46         w_array[cur_col_ind-1] = cur_sum;
47     }
48
49     bool flag = false;
50     for (int w_el_ind = 0; w_el_ind < A.max_value; w_el_ind++) {
51         flag += w_array[w_el_ind] >= 0;
52     }
53
54     while (flag) {
55         for (int w_el_ind = 0; w_el_ind < A.max_value; w_el_ind++) {
56             if (w_array[w_el_ind] == 0) {
57                 l.display[w_el_ind] = levels_amount;
58                 w_array[w_el_ind] = i;
```

```
            }
        }

        for (int w_el_ind = 0; w_el_ind < A.max_value; w_el_ind++) {
            if (w_array[w_el_ind] == i) {
                for (int cur_col_ind = 1; cur_col_ind <= A.max_value;
   cur_col_ind++) {
                    w_array[cur_col_ind-1] -= bin_relation_getValue(A,
   w_el_ind+1, cur_col_ind);
                }
            }
        }

        i--;
        levels_amount++;
        flag = false;
        for (int w_el_ind = 0; w_el_ind < A.max_value; w_el_ind++) {
            flag += w_array[w_el_ind] >= 0;
        }
    }
    return l;
}

int main () {
    bin_relation A1 = createForVariant6(M1, M1_LEN);
    bin_relation A2 = createForVariant6(M2, M2_LEN);

    bin_relation dom_A1 = createDominationRelation(A1);
    printf("\nMatrix of domination relation over A1:\n");
    bin_relation_matrixPrint(dom_A1);
    bin_relation dom_A2 = createDominationRelation(A2);
    printf("\nMatrix of domination relation over A2:\n");
    bin_relation_matrixPrint(dom_A2);

    printf("\n");
    splitting L1 =
   getTopologicalSorting(createDominationRelation(dom_A1));
    splitting L2 =
   getTopologicalSorting(createDominationRelation(dom_A2));
    splitting_print(L1);
    printf("\n");
    splitting_print(L2);
}
```

ordered_sets/ordered_sets_main.c

На экран будут выведены следующие последовательности множеств.

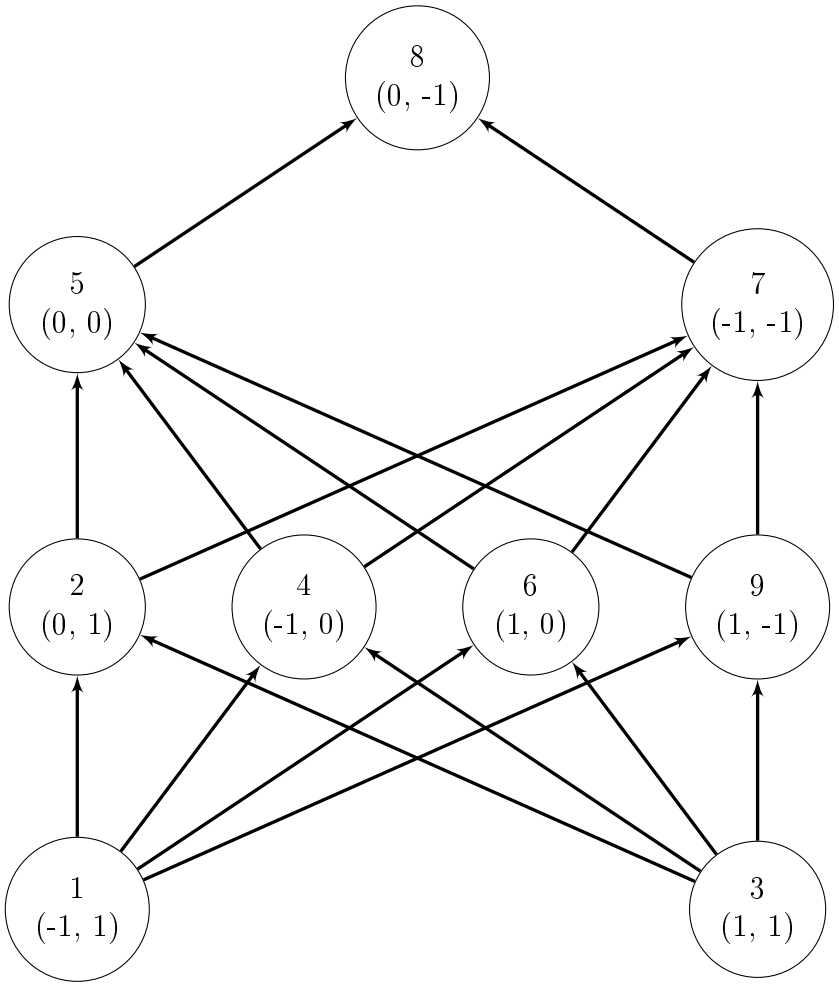
```
[{1, 3}, {2, 4, 6, 9}, {5, 7}, {8}]
[{1, 4}, {2, 5}, {3, 6, 9, 13}, {7, 10}, {8, 11, 12}]
```

Задание 4

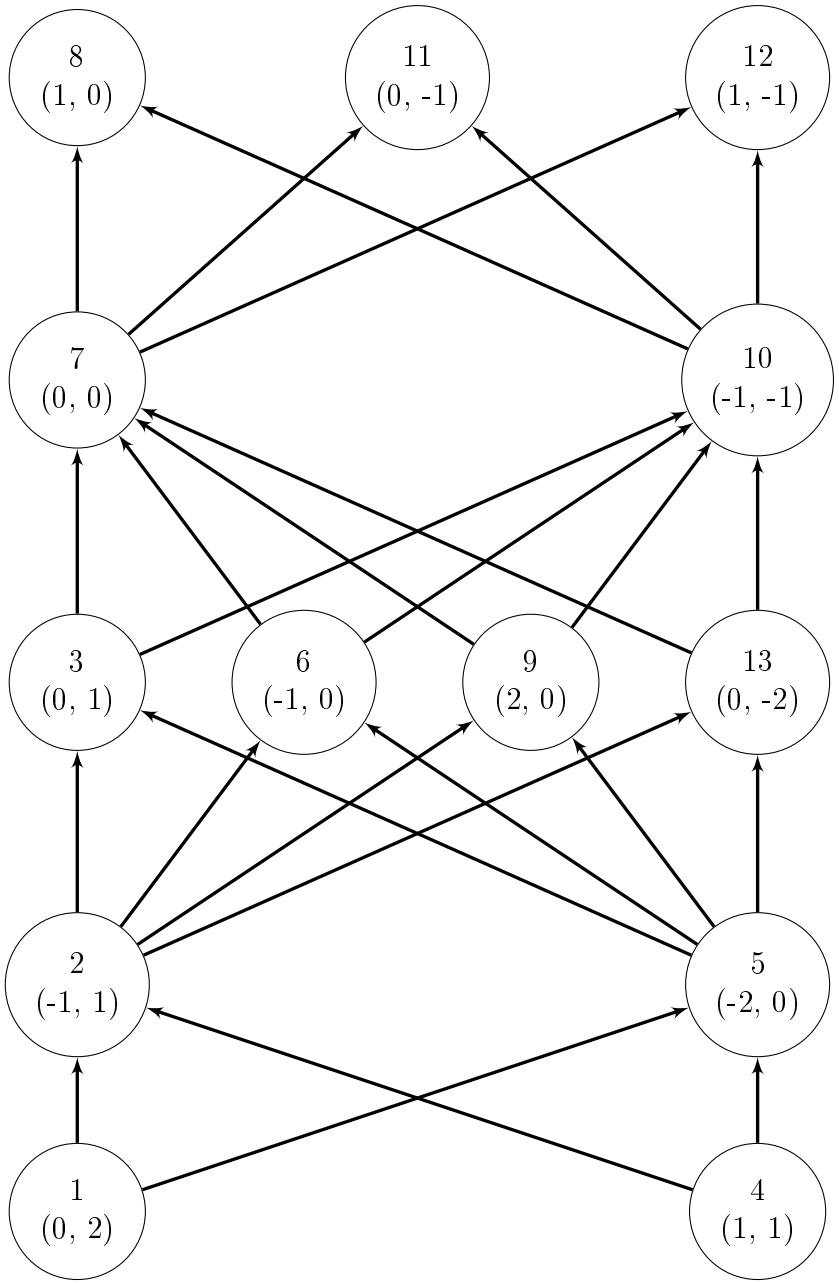
Текст задания: Изобразить диаграмму Хассе отношения доминирования на множествах M_1 и M_2.

Опираясь на полученные топологические сортировки, мы можем разбить элементы по уровням, а сверяясь с матрицами соответствующих отношений доминирования, провести дуги от элементов нижних уровней к элементам верхних.

Диаграмма Хассе на отношении доминирования над множеством M_1 будет выглядеть следующим образом:



А диаграмма Хассе на отношении доминирования над множеством M_2 — следующим образом:



Задание 5

Текст задания: Найти минимальные и максимальные элементы множеств M_1 и M_2.

Минимальными элементами в каждом отношении порядка будут те элементы, которые принадлежат наиболее низкому уровню в соответствующем отношении доминирования; максимальными — все те, которые принадлежат наиболее высокому уровню.

Проанализировав диаграмму отношения доминирования на множестве M_1, составленную

в задании 4, мы можем увидеть, что для него минимальными элементами будут элементы 1 и 3, соответствующие точкам $(-1, 1)$ и $(1, 1)$. Максимальный же элемент будет только один — 8, соответствующий точке $(0, -1)$. Это можно проверить, проанализировав соответствующие *столбцы* матрицы исходного отношения для минимальных элементов и *строки* для максимальных. Анализируя *столбец* элемента x, мы анализируем все пары вида (a, x). Если столбец содержит хотя бы одну единицу (кроме как в строке, соответствующей этому же элементу, если само по себе отношение не обладает свойством строгого порядка), это значит, что для элемента x существует элемент a, меньший его. Если же столбец не содержит единиц, значит, любой другой элемент больше x либо не сравним с ним. И наоборот, анализируя *строку* элемента x, мы анализируем все пары вида (x, a). Если столбец содержит хотя бы одну единицу, это значит, что для элемента x существует элемент a, больший его; иначе остальные элементы либо меньше x, либо не сравнимы с ним. Эти свойства соответствуют условиям, которым должны обладать минимальные и максимальные элементы. И если мы просмотрим строки максимальных элементов и столбцы минимальных, мы увидим, что единиц они и впрямь не содержат. Значит, они были выбраны верно.

Перейдя к отношению на множестве M_2, мы можем сказать, что минимальными элементами для него будут элементы 1 и 4, соответствующие точкам $(0, 2)$ и $(1, 1)$. Максимальных же элементов будет три — 8, 11 и 12, соответствующие точкам $(1, 0)$, $(0, -1)$ и $(1, -1)$. Точно так же *столбцы* минимальных элементов и *строки* максимальных в исходной матрице содержат только нули.

Задание 6

Текст задания: Найти, если существуют, наименьший и наибольший элементы множеств M_1 и M_2.

Если некоторое отношение доминирования имеет на своем наиболее низком уровне только один элемент, то он и будет наименьшим в соответствующем отношении порядка. И наоборот, если на наиболее высоком уровне отношения доминирования находится только один элемент, то он и будет наибольшим в соответствующем отношении порядка.

Согласно наблюдениям в задании 5, на нижнем уровне диаграммы отношения на множестве M_1 находятся два элемента. Все они являются минимальными, но ни один из них нельзя назвать наименьшим, потому что между собой они несравнимы. Это заключение можно подтвердить тем похожим, но все-таки отличающимся анализом матрицы исходного отношения. В *строках* матрицы, соответствующих этим элементам, содержится больше одного нуля — то есть существуют элементы, с которыми они несравнимы, кроме самих себя (это не могут быть элементы, которые меньше данных, потому что в задании 5 мы выяснили: элементы первого уровня минимальны). На верхнем же уровне диаграммы находится только один элемент, 8 (точка $(0, -1)$), и его можно назвать наибольшим. Это подтверждается, если мы проанализируем *столбец* матрицы, соответствующий этому элементу: он содержит только 1 ноль, знак того, что элемент не сравним сам с собой.

Произведем аналогичные рассуждения для отношения на множестве M_2. На нижнем уровне диаграммы отношения доминирования находятся два элемента. Они также являются лишь минимальными, но не наименьшими, и в соответствующих *строках* матрицы исходного

отношения больше одного нуля, что показывает, что существуют элементы, с которыми они не сравнимы. На верхнем же уровне диаграммы находятся три элемента, максимальных, но не наибольших. Это подтверждается анализом *столбцов* матрицы, соответствующих этим элементам: они содержат больше одного нуля, что значит, что существуют элементы, с которыми они несравнимы.

Подведем итоги: в отношении на множестве M_1 нет наименьшего элемента, но есть наибольший элемент: 8 точка $(0, -1)$ с порядковым номером 8. В отношении на множестве M_2 не существует ни наименьшего, ни наибольшего элемента.

Вывод

Если некоторое отношение обладает свойством порядка или зависящими от него свойствами строго или нестрого, линейного строгого или нестрогого порядка, то между его элементами возникают более глубокие логические связи и свойства. Среди элементов исходного множества для такого отношения всегда существуют максимальные и минимальные. Необязательно, но возможно существование наибольшего и наименьшего элемента, причем наибольший и наименьший элемент в множестве может быть только один. Также над исходным отношением порядка всегда можно построить отношение строгого порядка, а следом — и отношение доминирования. Для отношения доминирования, в свою очередь, элементы можно разбить на уровни — такое разбиение называется топологической сортировкой; ее получение помогает в дальнейшем быстро находит минимальные и максимальные, наибольший и наименьший (при их наличии) элементы для исходного отношения порядка. В ходе лабораторной работы научились создавать для отношения порядка отношение доминирования, запрограммировали алгоритм получения топологической сортировки по отношению доминирования, научились строить диаграммы Хассе для отношений доминирования и искать максимальные и минимальные, наибольший и наименьший элементы.