



Министерство науки и высшего образования Российской Федерации
Калужский филиал
федерального государственного бюджетного
образовательного учреждения высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(КФ МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ ИУК «Информатика и управление»

**КАФЕДРА ИУК4 «Программное обеспечение ЭВМ,
информационные технологии»**

ЛАБОРАТОРНАЯ РАБОТА

«Замыкание бинарных отношений»

ДИСЦИПЛИНА: «Дискретная математика»

Выполнил: студент гр. ИУК4-31Б


(подпись)

(Суриков Н. С.)
(Ф.И.О.)

Проверил:


(подпись)

(Никитенко У. В.)
(Ф.И.О.)

Дата сдачи (защиты):

Результаты сдачи (защиты):

- Балльная оценка:

- Оценка:

Цель: построение алгоритмов для определения свойств бинарного отношения и замыкания отношений.

Задачи:

1. разработать приложение, определяющее декартово произведение множеств, операции над отношениями, свойства бинарного отношения, заданного на конечном дискретном множестве, замыкание отношения.

Вариант 10

1. Заданы множества $N1$ и $N2$. Вычислить множества:

$$(N1 \times N2) \cap (N2 \times N1);$$

$$(N1 \times N2) \cup (N2 \times N1);$$

$$(N1 \cap N2) \times (N1 \cap N2);$$

$$(N1 \cup N2) \times (N1 \cup N2),$$

где $N1 = \{\text{цифры номера зачетной книжки, три последние}\}$; $N2 = \{\text{цифры даты и номера месяца рождения}\}$.

2. Отношения R , Q и $R \circ Q$ заданы на множестве $X = \{1, 2, 3, 4, 5, 6\}$.

2.1. Описать отношения R , Q и $R \circ Q$ списком пар.

2.2. Найти матрицы отношений R , Q и $R \circ Q$.

2.3. Для каждого из отношений R и Q определить область определения и область значений.

- 2.4. Определить свойства отношений. Затем посредством ответа на запрос выбирается требуемое отношение и требуемое замыкание. Определяется результат выбранного замыкания и выдается на экран с необходимыми пояснениями. Одновременно с результатом на экране должно присутствовать и исходное бинарное отношение.

Указание. Транзитивное замыкание реализовать алгоритмом I для нечётных вариантов, алгоритмом II (Уоршалла) – для чётных вариантов.

Этапы выполнения:

Этап 1 — Выбор средств для решения задачи:

Для решения задачи был выбран язык Python, а так же некоторые модули из numpy и tkinter, так как синтаксис языка достаточно прост, а библиотеки предоставляют готовые решения для быстрой разработки.

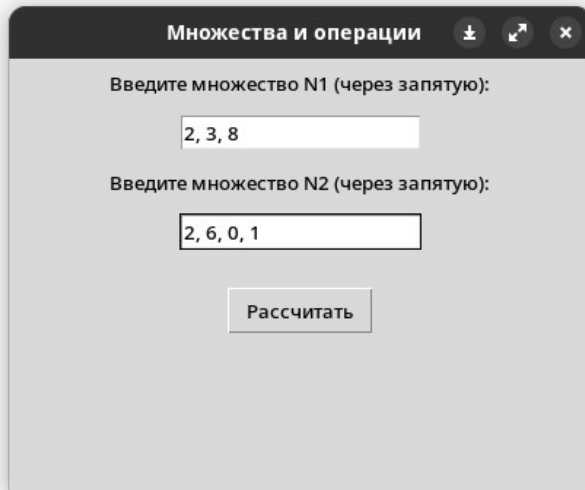
Установка: `pip install numpy`

Этап 2 — Создание консольной версии программы

Этап 3 — Создание графического интерфейса пользователя

Результат работы программы для задания 1:

Ввод:



Множества и операции

Введите множество N1 (через запятую):

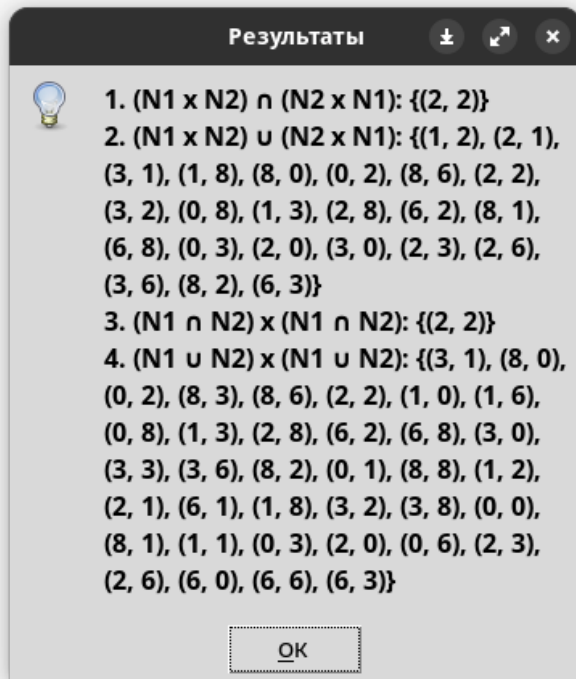
2, 3, 8

Введите множество N2 (через запятую):

2, 6, 0, 1

Рассчитать

Вывод:



Результаты

1. $(N1 \times N2) \cap (N2 \times N1): \{(2, 2)\}$

2. $(N1 \times N2) \cup (N2 \times N1): \{(1, 2), (2, 1), (3, 1), (1, 8), (8, 0), (0, 2), (8, 6), (2, 2), (3, 2), (0, 8), (1, 3), (2, 8), (6, 2), (8, 1), (6, 8), (0, 3), (2, 0), (3, 0), (2, 3), (2, 6), (3, 6), (8, 2), (6, 3)\}$

3. $(N1 \cap N2) \times (N1 \cap N2): \{(2, 2)\}$

4. $(N1 \cup N2) \times (N1 \cup N2): \{(3, 1), (8, 0), (0, 2), (8, 3), (8, 6), (2, 2), (1, 0), (1, 6), (0, 8), (1, 3), (2, 8), (6, 2), (6, 8), (3, 0), (3, 3), (3, 6), (8, 2), (0, 1), (8, 8), (1, 2), (2, 1), (6, 1), (1, 8), (3, 2), (3, 8), (0, 0), (8, 1), (1, 1), (0, 3), (2, 0), (0, 6), (2, 3), (2, 6), (6, 0), (6, 6), (6, 3)\}$

OK

Результат работы программы для задания 2:

Отношения и замыкания

Введите множество X
(через запятую, по умолчанию 1,2,3,4,5,6):

Установить множество X

Показать результаты

Выберите отношение:

R

Выберите замыкание:

рефлексивное

Рассчитать замыкание

Результаты

Исходное бинарное отношение R:
{(4, 4), (5, 5), (1, 1), (1, 4), (3, 3), (6, 3),
(3, 6), (2, 2), (6, 6), (2, 5), (4, 1), (5, 2)}

Исходное бинарное отношение Q:
{(6, 2), (3, 1), (6, 1), (5, 1), (4, 2), (6, 4),
(5, 3), (6, 3), (4, 1), (5, 2)}

Композиция $R \circ Q$:
{(6, 2), (1, 2), (3, 4), (2, 1), (3, 1), (6, 1),
(1, 1), (5, 1), (4, 2), (6, 4), (2, 3), (3, 3),
(2, 2), (5, 3), (3, 2), (6, 3), (4, 1), (5, 2)}

Область определения R: {1, 2, 3, 4, 5, 6},
Область значений R: {1, 2, 3, 4, 5, 6}
Область определения Q: {3, 4, 5, 6},
Область значений Q: {1, 2, 3, 4}

Свойства отношения R:
Рефлексивность: True
Симметричность: True
Транзитивность: True

Свойства отношения Q:
Рефлексивность: False
Симметричность: False
Транзитивность: True

OK

Введите множество X
(через запятую, по умолчанию 1,2,3,4,5,6):

Установить множество X

Показать результаты

Выберите отношение:

Q —

Выберите замыкание:

симметричное —

рефлексивное

симметричное

транзитивное

Введите множество X
(через запятую, по умолчанию 1,2,3,4,5,6):

Замыкание



Исходное бинарное отношение:

$\{(6, 2), (3, 1), (6, 1), (5, 1), (4, 2), (6, 4), (5, 3), (6, 3), (4, 1), (5, 2)\}$

Результат замыкания:

$\{(2, 4), (1, 5), (3, 1), (6, 1), (4, 6), (5, 1), (6, 4), (1, 6), (2, 5), (4, 1), (1, 3), (3, 5), (5, 2), (6, 2), (4, 2), (1, 4), (2, 6), (3, 6), (5, 3), (6, 3)\}$

OK

Листинг программы:

task1.py

```
1  import tkinter as tk
2  from tkinter import messagebox
3
4
5  def decart(s1, s2):
6      """Возвращает декартово произведение двух множеств."""
7      return set((a, b) for a in s1 for b in s2)
8
9
10 def calculate_results():
11     """Выполняет расчеты и отображает результаты."""
12     try:
13         # Получаем значения из полей ввода
14         n1_values = entry_n1.get()
15         n2_values = entry_n2.get()
16
17         # Преобразуем строки в множества
18         N1 = set(map(int, n1_values.split(",")))
19         N2 = set(map(int, n2_values.split(",")))
20
21         N1_x_N2 = decart(N1, N2)
22         N2_x_N1 = decart(N2, N1)
23
24         inter = N1_x_N2 & N2_x_N1
25         union = N1_x_N2 | N2_x_N1
26
27         N1_inter_N2 = N1 & N2
28         inter_dec = decart(N1_inter_N2, N1_inter_N2)
29
30         N1_union_N2 = N1 | N2
31         union_dec = decart(N1_union_N2, N1_union_N2)
32
33         results = (
34             f"1. (N1 x N2) ∩ (N2 x N1): {inter}\n"
35             f"2. (N1 x N2) ∪ (N2 x N1): {union}\n"
36             f"3. (N1 ∩ N2) x (N1 ∩ N2): {inter_dec}\n"
```

```

37         f"4. (N1 ∪ N2) × (N1 ∪ N2): {union_dec}"
38     )
39
40     messagebox.showinfo("Результаты", results)
41
42     except ValueError:
43         messagebox.showerror(
44             "Ошибка", "Введите корректные числа, разделенные запятыми."
45         )
46
47
48 root = tk.Tk()
49 root.title("Множества и операции")
50 root.geometry("400x300")
51
52 label_n1 = tk.Label(root, text="Введите множество N1 (через запятую):")
53 label_n1.pack(pady=5)
54
55 entry_n1 = tk.Entry(root)
56 entry_n1.pack(pady=5)
57
58 label_n2 = tk.Label(root, text="Введите множество N2 (через запятую):")
59 label_n2.pack(pady=5)
60
61 entry_n2 = tk.Entry(root)
62 entry_n2.pack(pady=5)
63
64 calculate_button = tk.Button(root, text="Рассчитать",
65 command=calculate_results)
66 calculate_button.pack(pady=20)
67
68 root.mainloop()

```

task2.py

```

1 import tkinter as tk
2 from tkinter import messagebox
3 import numpy as np
4
5 default_X = X = {1, 2, 3, 4, 5, 6}
6
7

```

```

8 def relation_R(X):
9     """Возвращает бинарное отношение R для множества X."""
10    return {(m, n) for m in X for n in X if m % 3 == n % 3}
11
12
13 def relation_Q(X):
14     """Возвращает бинарное отношение Q для множества X."""
15    return {(m, n) for m in X for n in X if (m - n) >= 2}
16
17
18 def composition_R_Q(R, Q):
19     """Возвращает композицию отношений R и Q."""
20    return {(m, p) for m, n1 in R for n2, p in Q if n1 == n2}
21
22
23 def relation_matrix(X, relation):
24     """Возвращает матрицу отношения для заданного множества X и отношения."""
25    matrix = np.zeros((len(X), len(X)), dtype=int)
26    elements = list(X)
27    for m, n in relation:
28        matrix[elements.index(m)][elements.index(n)] = 1
29    return matrix
30
31
32 def domain_and_range(relation):
33     """Возвращает область определения и область значений для отношения."""
34    domain = {m for m, n in relation}
35    range_ = {n for m, n in relation}
36    return domain, range_
37
38
39 def is_reflexive(X, relation):
40     """Проверяет, является ли отношение рефлексивным."""
41    return all((x, x) in relation for x in X)
42
43
44 def is_symmetric(relation):
45     """Проверяет, является ли отношение симметричным."""
46    return all((n, m) in relation for m, n in relation)
47
48
49 def is_transitive(relation):

```



```

50     """Проверяет, является ли отношение транзитивным."""
51     return all(
52         (m, p) in relation for m, n1 in relation for n2, p in relation if n1
53         == n2
54     )
55
56 def reflexive_closure(X, relation):
57     """Возвращает рефлексивное замыкание отношения."""
58     return relation | {(x, x) for x in X}
59
60
61 def symmetric_closure(relation):
62     """Возвращает симметричное замыкание отношения."""
63     return relation | {(n, m) for m, n in relation}
64
65
66 def transitive_closure(X, relation):
67     """Возвращает транзитивное замыкание отношения."""
68     matrix = relation_matrix(X, relation)
69     elements = list(X)
70     n = len(X)
71
72     for k in range(n):
73         for i in range(n):
74             for j in range(n):
75                 matrix[i][j] = matrix[i][j] or (matrix[i][k] and matrix[k]
76 [j])
77
78     closure_relation = set()
79     for i in range(n):
80         for j in range(n):
81             if matrix[i][j]:
82                 closure_relation.add((elements[i], elements[j]))
83     return closure_relation
84
85 def show_results():
86     """Отображает результаты отношений и их свойства."""
87     global X
88     if not X:
89         messagebox.showerror(

```

```

90         "Ошибка", "Пожалуйста, задайте множество X перед получением
результатов."
91     )
92     return
93
94     R = relation_R(X)
95     Q = relation_Q(X)
96     R_composed_Q = composition_R_Q(R, Q)
97
98     results = f"Исходное бинарное отношение R:\n{R}\n"
99     results += f"Исходное бинарное отношение Q:\n{Q}\n"
100    results += f"Композиция R∘Q:\n{R_composed_Q}\n\n"
101
102    domain_R, range_R = domain_and_range(R)
103    domain_Q, range_Q = domain_and_range(Q)
104
105    results += f"Область определения R: {domain_R},\nОбласть значений R:
{range_R}\n"
106    results += f"Область определения Q: {domain_Q},\nОбласть значений Q:
{range_Q}\n\n"
107
108    results += f"Свойства отношения R:\n"
109    results += f"Рефлексивность: {is_reflexive(X, R)}\n"
110    results += f"Симметричность: {is_symmetric(R)}\n"
111    results += f"Транзитивность: {is_transitive(R)}\n\n"
112
113    results += f"Свойства отношения Q:\n"
114    results += f"Рефлексивность: {is_reflexive(X, Q)}\n"
115    results += f"Симметричность: {is_symmetric(Q)}\n"
116    results += f"Транзитивность: {is_transitive(Q)}\n"
117
118    messagebox.showinfo("Результаты", results)
119
120
121    def calculate_closure():
122        """Рассчитывает замыкание выбранного отношения."""
123        closure_type = closure_var.get()
124        relation_choice = relation_var.get()
125
126        if not X:
127            messagebox.showerror(
128                "Ошибка", "Пожалуйста, задайте множество X перед расчетом
замыкания."

```

```

129         )
130         return
131
132     R = relation_R(X)
133     Q = relation_Q(X)
134     R_composed_Q = composition_R_Q(R, Q)
135
136     if relation_choice == "R":
137         selected_relation = R
138     elif relation_choice == "Q":
139         selected_relation = Q
140     elif relation_choice == "R∘Q":
141         selected_relation = R_composed_Q
142     else:
143         messagebox.showerror("Ошибка", "Неверный выбор отношения!")
144         return
145
146     if closure_type == "рефлексивное":
147         closure = reflexive_closure(X, selected_relation)
148     elif closure_type == "симметричное":
149         closure = symmetric_closure(selected_relation)
150     elif closure_type == "транзитивное":
151         closure = transitive_closure(X, selected_relation)
152     else:
153         messagebox.showerror("Ошибка", "Неверный выбор замыкания!")
154         return
155
156     result_message = f"Исходное бинарное отношение:\n{selected_relation}\n"
157     result_message += f"Результат замыкания:\n{closure}"
158
159     messagebox.showinfo("Замыкание", result_message)
160
161
162 def set_custom_set():
163     """Устанавливает пользовательское множество X."""
164     global X
165     X = default_X
166     input_value = entry_X.get()
167     if input_value:
168         try:
169             X = set(map(int, input_value.split(",")))
170         except ValueError:

```

```
171         messagebox.showerror(
172             "Ошибка", "Введите корректные числа, разделенные запятыми."
173         )
174         return
175
176
177 root = tk.Tk()
178 root.title("Отношения и замыкания")
179 root.geometry("400x400")
180
181 label_X = tk.Label(
182     root, text="Введите множество X\n(через запятую, по умолчанию
183     1, 2, 3, 4, 5, 6):"
184 )
185 label_X.pack(pady=10)
186
187 entry_X = tk.Entry(root)
188 entry_X.pack(pady=10)
189
190 set_button = tk.Button(root, text="Установить множество X",
191     command=set_custom_set)
192 set_button.pack(pady=10)
193
194 results_button = tk.Button(root, text="Показать результаты",
195     command=show_results)
196 results_button.pack(pady=10)
197
198 relation_var = tk.StringVar(value="R")
199 relation_label = tk.Label(root, text="Выберите отношение:")
200 relation_label.pack()
201 relation_options = ["R", "Q", "R∘Q"]
202 relation_menu = tk.OptionMenu(root, relation_var, *relation_options)
203 relation_menu.pack()
204
205 closure_var = tk.StringVar(value="рефлексивное")
206 closure_label = tk.Label(root, text="Выберите замыкание:")
207 closure_label.pack()
208 closure_options = ["рефлексивное", "симметричное", "транзитивное"]
209 closure_menu = tk.OptionMenu(root, closure_var, *closure_options)
210 closure_menu.pack()
211
212 closure_button = tk.Button(root, text="Рассчитать замыкание",
213     command=calculate_closure)
```

```
210 closure_button.pack(pady=10)
211
212 root.mainloop()
213
```

Вывод: В ходе работы были изучены алгоритмы для определения свойств бинарного отношения и замыкания отношений. Разработаны программы для их расчета и вывода.