Министерство науки и высшего образования Российской Федерации

Пензенский государственный университет

Кафедра «Вычислительная техника»

**ОТЧЕТ**

по лабораторной работе №6

по курсу «Логика и основы алгоритмизации в инженерных задачах»

на тему «Унарные и бинарные операции над графами»

Выполнили:

студенты группы 22BВВ1

Митрошин Ю.Е

Коннов А.Д

Приняли:

Акифьев И.В

Юрова О.В

Пенза 2023

**Название**

Унарные и бинарные операции над графами

**Общие сведения.**

Все унарные операции над графами можно объединить в две группы. Первую группу составляют операции, с помощью которых из исходного графа *G*1*,* можно построить граф *G*2 с меньшим числом элементов. В группу входят операции удаления ребра или вершины, отождествления вершин, стягивание ребра. Вторую группу составляют операции, позволяющие строить графы с большим числом элементов. В группу входят операции расщепления вершин, добавления ребра.

*Отождествление вершин.* В графе *G*1 выделяются вершины *и,v.* Определяют окружение *Q*1 вершины *u*,и окружение *Q*2 вершины *v,* вычисляют их объединение *Q* = *Q1* *https://lh7-us.googleusercontent.com/gvIHzL8xddlgnsRYOjdBTmR1vpzUKbQyRqHoBp1yR0qhEB-JKT_nuKxm592aLgrzyGWfeuuSSYGbKh_-MXZytihAgbxB4CF_3_kNwvO0tJi-QrmmFqlpw9nclye_i9jAM7Spxklc3acNODJpYqLJrw Q2.* Затем над графом *G*1 выполняются следующие преобразования:

* из графа *G*1 удаляют вершины *u,* *v (H*1 *= G*1 *- u - v);*
* к графу *Н*1присоединяют новую вершину *z (H*1 *= H*1 *+z);*
* вершину *z* соединяют ребром с каждой из вершин *w*1*https://lh7-us.googleusercontent.com/ipetMDxpiaqr1EC-CNQ7Zw3wISWiPXb9xN9pljp5gMFTIFOKcuTVx1qWjxD4xSgdUdjMLzB7n3--g0kfMAEzbb-luw3Ixar8-WfEsPi7W-sUxuD81SfW3yewrwaDXZflE0SK-hogce1esvkn20p0pgQ*

*(G*2 *= H*1 *+ zwi*, *i =* 1,2,3*,…).*

*Стягивание ребра.* Данная операция является операцией отождествления смежных вершин *и, v* в графе *G*1.

Наиболее важными бинарными операциями являются: объединение, пересечение, декартово произведение и кольцевая сумма.

*Объединение.* Граф *G* называется объединением или наложением графов *G*1 и *G*2, если *VG = V*1*https://lh7-us.googleusercontent.com/DvAmeNrOfUH0JaT_US_ELP31pv9rsVvw_xnAugqlSWUPQD0aTkhsNPGHJNwgZQVNdy-7ixjJsB2xjTMLFS-Lac7F2_p_5XtIghZGPXHipNX0GKDHzIIO7liTPOgWMmqd2FUju_TZRfoWc7xIZCZEFAV*2*; UG = U*1*https://lh7-us.googleusercontent.com/DvAmeNrOfUH0JaT_US_ELP31pv9rsVvw_xnAugqlSWUPQD0aTkhsNPGHJNwgZQVNdy-7ixjJsB2xjTMLFS-Lac7F2_p_5XtIghZGPXHipNX0GKDHzIIO7liTPOgWMmqd2FUju_TZRfoWc7xIZCZEFA U*2 (рис. 1).

Объединение графов *G*1 и *G*2 называется дизъюнктным, если *V*1*https://lh7-us.googleusercontent.com/ww71SYhs20qmT3S0o7HQSX4D1LXNiqF9Arp1umW3XXyLI1zhPqiTJWyO6Skr7rst-1tRmuu7FQqJbvZNJK4hACa1XTCKuv8DJh9OlpzLeDIoTh7xZv7P2FeSBvltq4jdVTD5ct-9R-tVYEsCIWOIgQV*2 *= https://lh7-us.googleusercontent.com/iSPv37raKQWXmHa0Q5v-UX06bhvzw85dMgIxZVs9OQs44Z4OpMGT1qHs70TUOxC1mW8iSVbJoX11PHR7yDf5PqShQNUzL61vcVvvA4SHsIPq4lzhKcmI7atYjOcFCeEbVUDNOfL6JudKunggZfND5g*. При дизъюнктном объединении никакие два из объединяемых графов не должны иметь общих вершин.

*Пересечение.* Граф *G* называется пересечением графов *G*1, *G*2,если *VG = V*1*https://lh7-us.googleusercontent.com/ww71SYhs20qmT3S0o7HQSX4D1LXNiqF9Arp1umW3XXyLI1zhPqiTJWyO6Skr7rst-1tRmuu7FQqJbvZNJK4hACa1XTCKuv8DJh9OlpzLeDIoTh7xZv7P2FeSBvltq4jdVTD5ct-9R-tVYEsCIWOIgQV*2и *UG = U*1*https://lh7-us.googleusercontent.com/DvAmeNrOfUH0JaT_US_ELP31pv9rsVvw_xnAugqlSWUPQD0aTkhsNPGHJNwgZQVNdy-7ixjJsB2xjTMLFS-Lac7F2_p_5XtIghZGPXHipNX0GKDHzIIO7liTPOgWMmqd2FUju_TZRfoWc7xIZCZEFAU*2 (риc.2). Операция "пересечения" записывается следующим образом: *G = G*1*https://lh7-us.googleusercontent.com/ww71SYhs20qmT3S0o7HQSX4D1LXNiqF9Arp1umW3XXyLI1zhPqiTJWyO6Skr7rst-1tRmuu7FQqJbvZNJK4hACa1XTCKuv8DJh9OlpzLeDIoTh7xZv7P2FeSBvltq4jdVTD5ct-9R-tVYEsCIWOIgQG*2*.*

*Декартово произведение.* Граф *G* называется декартовым произведением графов *G*1 и *G*2 если *VG* = *V*1https://lh7-us.googleusercontent.com/pfKCYoEXNrHhyxIGRvd6dEWKmuJMxi2J0SERVzhHcGq8HAFTITTJ-GGBbsH4t8199HcV6rLS0BgfcepHsyUZultcZ83PWHKgFEW5fbOLqmrD2LAd4rdynhSvF-lMenWA8Vrw-TEWCtn9L6dgGqoS2g*V*2 —декартово произведение множеств вершин графов *G*1, *G*2, а множество ребер *U*c задается следующим образом: вершины (*zi*, *vk*) и (*zj*, *vl*) смежны в графе *G* тогда и только тогда, когда *zi* = *zj*(*i* = *j*), a *v*k и *vl* смежны в *G*2 или *vk* = *vl*(*k* = *l*), смежны в графе *G*1 (см. рис.3).

*Кольцевая сумма* графов представляет граф, который не имеет изолированных вершин и состоит из ребер, присутствующих либо в первом исходном графе, либо во втором. Кольцевая сумма определяется следующим соотношением: *G* = *G*1 https://lh7-us.googleusercontent.com/S_7iOjrhl3vb8VdhKXYTl659QfjlEsCZ6Ug_y6Miiu1lXR-BNvAszs3EWc9_JiW8auqBVncTUHdzd5f0ctyoNe8t1Es-0blYItlSWDewRbT6L1_4ebd3PFW8T9NR-8y4f4CzuJYYmNKhJecEC423RQ *G*2 (рис.4).

**Задание 1**

1. Сгенерируйте (используя генератор случайных чисел) две матрицы *M*1*, М*2 смежности неориентированных помеченных графов *G*1, *G*2. Выведите сгенерированные матрицы на экран.
2. \* Для указанных графов преобразуйте представление матриц смежности в списки смежности. Выведите полученные списки на экран.

**Задание 2**

1. Для матричной формы представления графов выполните операцию:

а) отождествления вершин

б) стягивания ребра

в) расщепления вершины

Номера выбираемых для выполнения операции вершин ввести с клавиатуры.

Результат выполнения операции выведите на экран.

1. \* Для представления графов в виде списков смежности выполните операцию:

а) отождествления вершин

б) стягивания ребра

в) расщепления вершины

Номера выбираемых для выполнения операции вершин ввести с клавиатуры.

Результат выполнения операции выведите на экран.

**Задание 3**

1. Для матричной формы представления графов выполните операцию:

а) объединения *G* = *G*1 https://lh7-us.googleusercontent.com/DvAmeNrOfUH0JaT_US_ELP31pv9rsVvw_xnAugqlSWUPQD0aTkhsNPGHJNwgZQVNdy-7ixjJsB2xjTMLFS-Lac7F2_p_5XtIghZGPXHipNX0GKDHzIIO7liTPOgWMmqd2FUju_TZRfoWc7xIZCZEFA *G*2

б) пересечения  *G* = *G*1 https://lh7-us.googleusercontent.com/_6RgyIxcTbp6e57X2QxJikx69xKSkL7mGi6q1kaOSXxnMhYEpj8sNqkBp8Q3aixBwAuCtfVye-JXBk1hbi1q_rSM-kxd8QS63Grg5Dg48SjTeu2JFP5CRpyaO8wO6STydIPdR6gb_Do9-0FmMA2Psw *G*2

в) кольцевой суммы *G* = *G*1 https://lh7-us.googleusercontent.com/S_7iOjrhl3vb8VdhKXYTl659QfjlEsCZ6Ug_y6Miiu1lXR-BNvAszs3EWc9_JiW8auqBVncTUHdzd5f0ctyoNe8t1Es-0blYItlSWDewRbT6L1_4ebd3PFW8T9NR-8y4f4CzuJYYmNKhJecEC423RQ *G*2

Результат выполнения операции выведите на экран.

**Листинг**

import random

size1 = int(input("Введите размер матрицы M1: "))

size2 = int(input("Введите размер матрицы M2: "))

M1 = []

for i in range(size1):

row = []

for j in range(size1):

row.append(0)

M1.append(row)

for i in range(size1):

for j in range(i + 1, size1):

M1[i][j] = M1[j][i] = random.randint(0, 1)

lists = [[] for \_ in range(size1)]

for i in range(size1):

for j in range(size1):

if M1[i][j] == 1:

lists[i].append(j)

M2 = []

for i in range(size2):

row = []

for j in range(size2):

row.append(0)

M2.append(row)

for i in range(size2):

for j in range(i + 1, size2):

M2[i][j] = M2[j][i] = random.randint(0, 1)

print("Матрица смежности для G1: ")

for row in M1:

print(row)

print("\nСписок смежности для M1: ")

for i, verticles in enumerate(lists):

print(f"Вершина {i}: {verticles}")

print("\nМатрица смежности для G2: ")

for row in M2:

print(row)

G\_union = []

size\_union = max(size1, size2)

for i in range(size\_union):

row = []

for j in range(size\_union):

val1 = M1[i][j] if i < size1 and j < size1 else 0

val2 = M2[i][j] if i < size2 and j < size2 else 0

row.append(val1 or val2)

G\_union.append(row)

print("\nМатрица смежности для объединения графов G (G1 U G2): ")

for row in G\_union:

print(row)

G\_intersection = []

size\_intersection = min(size1, size2)

for i in range(size\_intersection):

row = []

for j in range(size\_intersection):

row.append(M1[i][j] and M2[i][j])

G\_intersection.append(row)

print("\nМатрица смежности для пересечения графов G (G1 ? G2): ")

for row in G\_intersection:

print(row)

G\_ring\_sum = []

size\_ring\_sum = max(size1, size2)

for i in range(size\_ring\_sum):

row = []

for j in range(size\_ring\_sum):

if i < size1 and j < size1:

val1 = M1[i][j]

else:

val1 = 0

if i < size2 and j < size2:

val2 = M2[i][j]

else:

val2 = 0

row.append(val1 ^ val2)

G\_ring\_sum.append(row)

print("\nМатрица смежности для кольцевой суммы графов G (G1 ? G2): ")

for row in G\_ring\_sum:

print(row)

dekart = []

size\_product = size1 \* size2

for i in range(size1 \* size2):

row = []

for j in range(size1 \* size2):

row.append(0)

dekart.append(row)

for i in range(size1):

for j in range(size1):

for k in range(size2):

for l in range(size2):

if M1[i][j] == 1 and M2[k][l] == 1:

dekart[i \* size2 + k][j \* size2 + l] = 1

print("\nМатрица смежности для декартова произведения графов G1 и G2:")

for row in dekart:

print(row)

while True:

print("1. Отождествление вершин")

print("2. Стягивание ребра")

print("3. Расщепление вершины")

print("4. Завершить")

choice = int(input("\nВыберите операцию для матрицы M1: "))

if choice == 4:

break

elif choice == 1:

v1 = int(input("Введите номер первой вершины для отождествления: "))

v2 = int(input("Введите номер второй вершины для отождествления: "))

if v1 < size1 and v2 < size1:

size = len(M1)

if v1 < size and v2 < size:

for i in range(size):

if i != v2:

M1[v1][i] = M1[v1][i] or M1[v2][i]

M1[i][v1] = M1[i][v1] or M1[i][v2]

M1[v1][v1] = 1

new\_size = size - 1

new\_matrix = [[0] \* new\_size for \_ in range(new\_size)]

for i in range(size):

if i != v2:

for j in range(size):

if j != v2:

new\_i = i if i < v2 else i - 1

new\_j = j if j < v2 else j - 1

new\_matrix[new\_i][new\_j] = M1[i][j]

M1 = new\_matrix

elif choice == 2:

v1 = int(input("Введите номер первой вершины: "))

v2 = int(input("Введите номер второй вершины: "))

if v1 < size1 and v2 < size1:

size = len(M1)

if v1 < size and v2 < size and M1[v1][v2] == 1:

for i in range(size):

if i != v2:

M1[v1][i] = M1[v1][i] or M1[v2][i]

M1[i][v1] = M1[i][v1] or M1[i][v2]

new\_size = size - 1

new\_matrix = [[0] \* new\_size for \_ in range(new\_size)]

for i in range(size):

if i != v2:

for j in range(size):

if j != v2:

new\_i = i if i < v2 else i - 1

new\_j = j if j < v2 else j - 1

new\_matrix[new\_i][new\_j] = M1[i][j]

for i in range(new\_size):

new\_matrix[i][i] = 0

M1 = new\_matrix

else:

print("Такого ребра не существует")

continue

elif choice == 3:

v1 = int(input("Введите номер вершины для расщепления: "))

if v1 < size1:

size = len(M1)

size += 1

new\_matrix = [[0] \* size for \_ in range(size)]

for i in range(size - 1):

for j in range(size - 1):

new\_matrix[i][j] = M1[i][j]

for i in range(size - 1):

if M1[i][v1] == 1:

new\_matrix[i][size - 1] = 1

new\_matrix[size - 1][i] = 1

M1 = new\_matrix

print("Итоговая матрица M1: ")

for row in M1:

print(row)

while True:

print("1. Отождествление вершин")

print("2. Стягивание ребра")

print("3. Расщепление вершин")

print("4. Завершить")

choice = int(input("\nВыберите операцию для списка смежности M1:"))

if choice == 4:

break

elif choice == 1:

v1 = int(input("Введите номер первой вершины для отождествления: "))

v2 = int(input("Введите номер второй вершины для отождествления: "))

if v1 < size1 and v2 < size1:

new\_adj\_list = list(set(lists[v1] + lists[v2]))

if v2 not in lists[v1]:

new\_adj\_list.append(v1)

if M1[v1][v2] != 1:

new\_adj\_list = [x for x in new\_adj\_list if x != v1]

lists[v1] = new\_adj\_list

for vertex in lists:

if v2 in vertex:

vertex.remove(v2)

for index, item in enumerate(vertex):

if item == v2:

vertex[index] = v1

lists.pop(v2)

size1 -= 1

elif choice == 2:

v1 = int(input("Введите номер первой вершины, соединенной ребром: "))

v2 = int(input("Введите номер второй вершины, соединенной ребром: "))

if v1 < size1 and v2 < size1:

new\_adj\_list = list(set(lists[v1] + lists[v2]))

if M1[v1][v2] == 1:

new\_adj\_list = [x for x in new\_adj\_list if x != v1]

lists[v1] = new\_adj\_list

for vertex in lists:

if v2 in vertex:

vertex.remove(v2)

for index, item in enumerate(vertex):

if item == v2:

vertex[index] = v1

lists.pop(v2)

size1 -= 1

elif choice == 3:

v1 = int(input("Введите номер вершины для расщепления: "))

if v1 < size1:

new\_vertex = size1

lists.append([v1])

for vertex in lists[v1]:

lists[new\_vertex].append(vertex)

lists[vertex].append(new\_vertex)

lists[vertex].remove(v1)

lists[v1].append(new\_vertex)

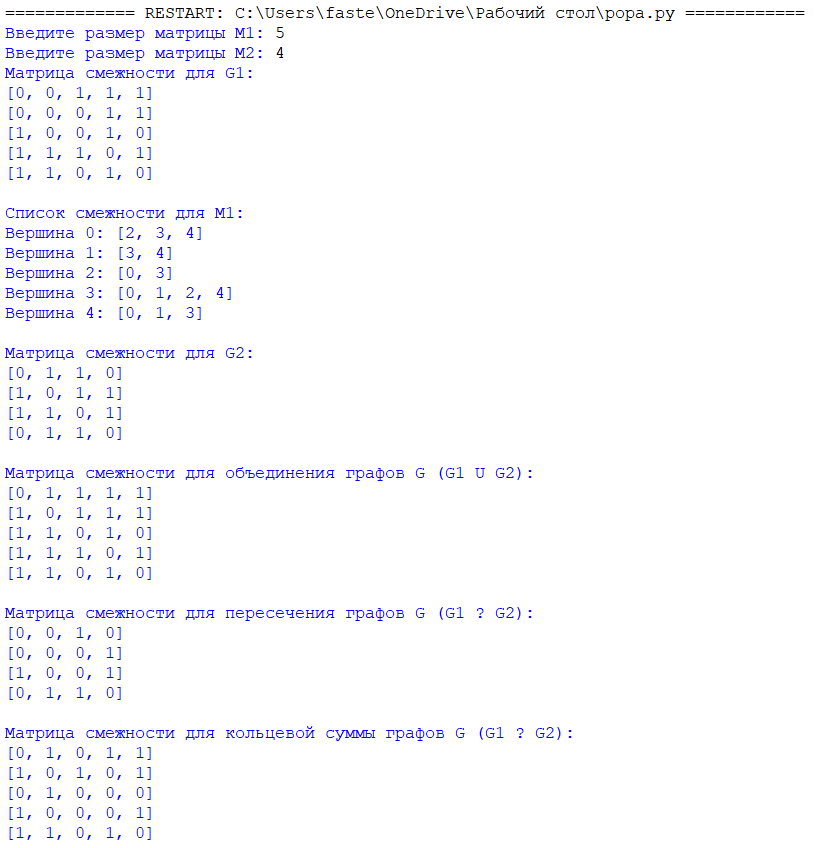
size1 += 1

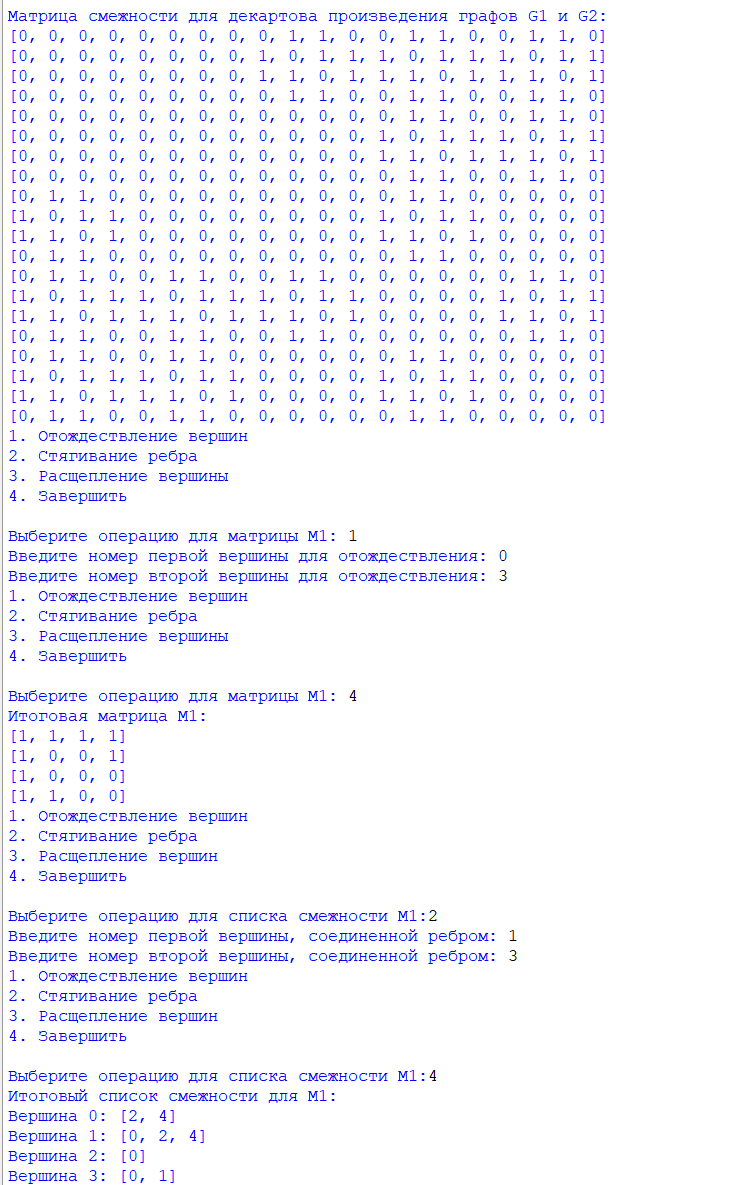
print("Итоговый список смежности для M1:")

for i, vertices in enumerate(lists):

print(f"Вершина {i}: {vertices}")

**Результат работы программы:**

****

****

**Вывод:** в ходе лабораторной работы мы изучили унарные и бинарные операции над графами.