федеральное государственное автономное образовательное учреждение
высшего образования
Санкт-Петербургский национальный исследовательский университет
информационных технологий, механики и оптики

Лабораторная работа №6
по дисциплине: «Теория Алгоритмов»
Тема: *«Социальные графы»*

Выполнила ученица 223 группы
Никончук А.П.

Проверил
Сорокин Д.С. _____

Санкт-Петербург,
2015 г.

1. Функция half_file. Проходится по каждой линии файла и записывает в список res одну из двух половин файла, в зависимости от part

```
afile=open('NastyaDmitrievna1.txt', encoding='utf-8')
#afile=open('NastyaLukina.txt', encoding='utf-8')

def half_file(afile,part):
#создает список для заданной (части) таблицы
    res = []
    line=afile.readlines()
    for i in range(len(line)):
        if line[i] == '\n':
            if part==1:
                res = line[1:(i)]
            if part==2:
                res = line[(i+2):]
    return res

half_file1 = half_file(afile,1)
#print(half_file1)
afile.seek(0)
half_file2 = half_file(afile,2)
#print(half_file2)
```

2. Функции one_id и list_of_id выделяют id.

```
def one_id(line,count,side,part):
#выделяет один id, учитывая условия
    if line[count]==' ':
        if part==1:
            if side==1:
                line=line[(count+2):]
            if side==2:
                line=line[:(count-1)]
        if part==2:
            if side==1:
                line=line[(count+1):]
            if side==2:
                line=line[:(count)]
        return line
    count+=1
    return one_id(line,count,side,part)

def list_of_id(res,side,part):
#создаёт список id, для заданной части
    res1=[]
    for line in res:
        res1.append(one_id(line,1,side,part))
    return res1


added1=list_of_id(half_file1,1,1)
part1_ids1=list_of_id(added1,2,1)
#print(part1_ids1)

added2=list_of_id(added1,1,1)
part2_ids1=list_of_id(added2,2,1)
#print(part2_ids1)
```

3. Функция list_of_names. Возвращает словарь, где ключ – id значение, значение – Имя Фамилия.

```
def list_of_names(f=half_file2):
    alist={}
    ids2=list_of_id((list_of_id(f,1,2)),2,2)
    added_n=list_of_id(f,1,2)
    names=list_of_id((list_of_id(f,1,1)),1,1)
    first_names=list_of_id(names,2,1)
    names1=list_of_id(names,1,1)
    second_names=list_of_id(names1,2,1)
    #print(first_names)
    #print(second_names)
    for i in range(len(first_names)):
        alist[ids2[i]]=[first_names[i]+' '+second_names[i]]
    return alist

list_of_names=list_of_names()
#print(len(list_of_names))
#print(list_of_names)
```

4. Создаёт граф, где ключ – это id значение, а значение по ключу – список из id значений, с которыми связан этот человек. Принимает два списка, первый – list1 – это список из id первого столбика первой части файла, второй – list2 – список из id второго столбика первой части файла. Добавляет в качестве ключей все имена из списка list_of_names. Проходится по всем индексам в диапазоне длинны списка list2 (list1 и list2 одного размера) и для каждой пары (парой здесь считаются значения с одинаковым индексом) проверяет по ключу из одного списка в новом (создаваемом) списке есть ли в значениях ключа id значение из другого списка, если нет, то добавляет его. Если у ключа (id значения) не будет связей, в графе значение по его ключу будет пустым.

```
def make_graf(list1,list2):
#создаёт граф из id
    graf={}
    for i in list_of_names:
        if i not in graf:
            graf[i]=[]
    for j in range(len(list2)):
        if list2[j] not in graf[list1[j]]:
            graf[list1[j]].append(list2[j])
        if list1[j] not in graf[list2[j]]:
            graf[list2[j]].append(list1[j])
    return graf

G=make_graf(part1_ids1,part2_ids1)
#print(len(G))
#print(G)
graf_of_id=make_graf(part1_ids1,part2_ids1)
#print(graf_of_id)
```

```
Администратор@TATYANNA /d/Anaconda3 (master)
$ python lab_6.py
{'136120674': ['241258029'], '143897269': ['19550563', '21660247', '23302316', '
25623423', '28967717', '33536820', '61803440', '118912625', '159804367', '160970
469'], '25623423': ['9291723', '19550563', '20255352', '21660247', '21910581', '
23302316', '28967717', '33536820', '39728844', '40252187', '42719145', '44927678
', '56293869', '61803440', '118912625', '143897269', '159804367', '160970469', '
161672134'], '33456137': ['3141809', '7294156', '10058881', '10846475', '1333668
7', '28700311', '32128201', '34691970', '50975821', '63050762', '75007000', '826
53922', '87067616', '91827024', '97151415', '137515619', '140831434', '166823591
', '201496980', '222769462', '239079498', '253191875'], '20255352': ['5780013',
```

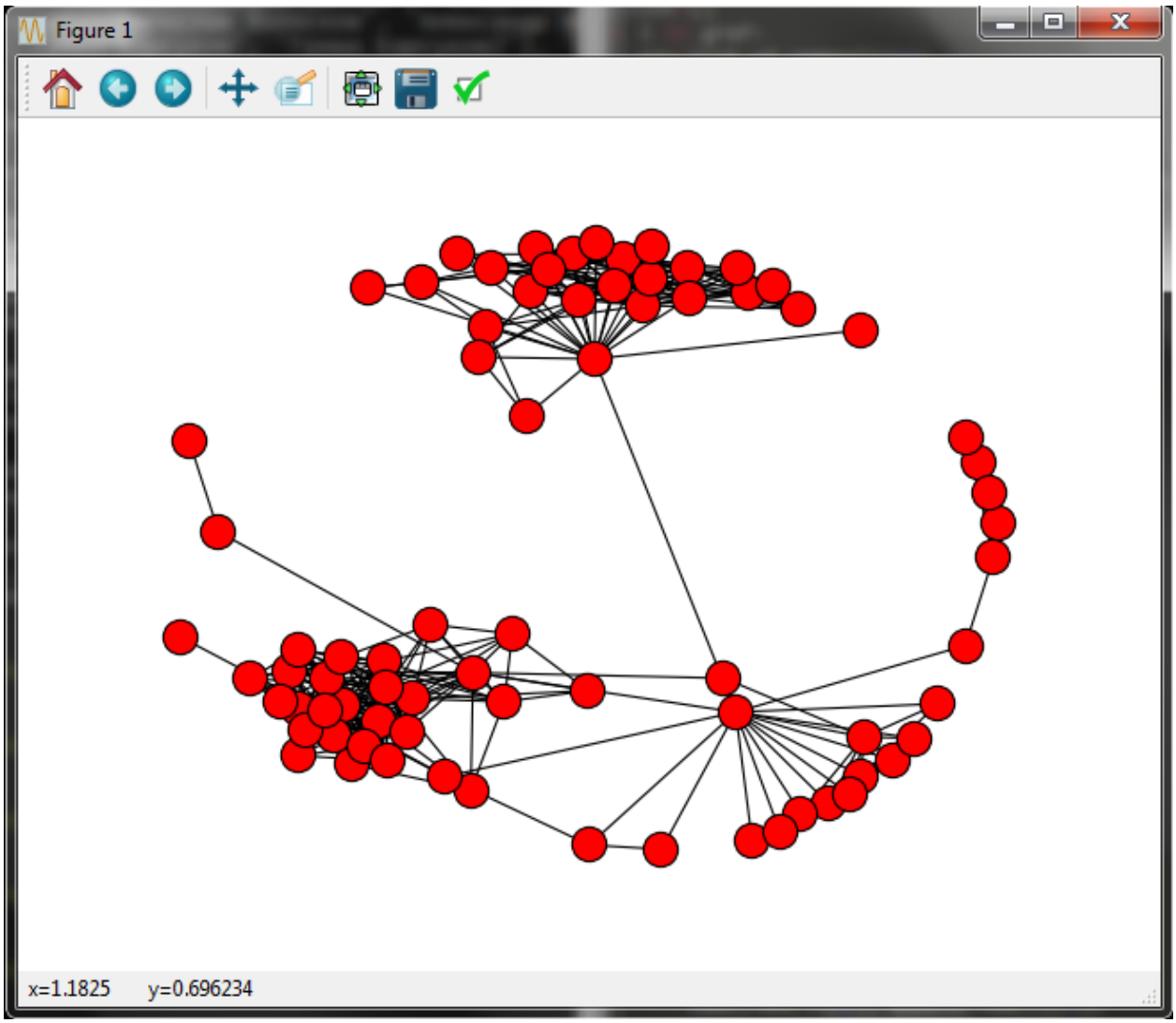
5. Функция make_graf_names принимает граф (graf), построенный с помощью функции make_graf. Для каждого ключа и его значений из graf и с помощью списка list_of_names добавляет в новый граф (new) имя, соответствующее id значению.

```
def make_graf_names(graf):
#создаёт граф имён
    new={}
    keylist=[]
    for i in graf:
        key=list_of_names[i][0]
        new[key]=[]
        if len(graf[i])>0:
            for j in graf[i]:
                #print(list_of_names[j][0])
                new[key].append(list_of_names[j][0])
    #print(len(new))
    return new

graf_of_names=make_graf_names(G)
#print(len(graf_of_names))
#print(graf_of_names)
```

```
Администратор@TATYANNA /d/Anaconda3 (master)
$ python lab_6.py
{'Кристина Эм': ['Екатерина Пономарёва', 'Александра Поганшева', 'Ника Полякова'
, 'Александра Поляничко', 'Дмитрий Коваль', 'Ника Кириллина', 'Даша Осетник', 'Н
аталья Ястребова', 'Владислав Эм', 'Сергей Савинов', 'Оганес Манасян', 'Анастаси
я Черногорова', 'Варя Лукина', 'Алина Сыртланова', 'Алина Филатова', 'Анастасия
Борщ', 'Юлия Ковтун', 'Владислав Эм2', 'Анастасия Сергеевна', 'Карина Калиниченк
о', 'Антон Костенко', 'Юлия Ким'], 'Анастасия Черногорова': ['Виктория Кучаева',
 'Ника Полякова', 'Александра Поляничко', 'Дмитрий Коваль', 'Василина Третьякова
', 'Степан Шибаев', 'Максим Аверин', 'Даша Осетник', 'Кристина Эм', 'Сергей Сави
нов', 'Варя Лукина', 'Алина Сыртланова', 'Анастасия Борщ', 'Юлия Ковтун'], 'Егор
 Васильев': ['Анастасия Чубарь', 'Алина Филатова', 'Роман Палонен'], 'Виталик Ал
пеев': ['Макар Курбатов', 'Игорь Кузнецов', 'Эрик Муртазин', 'Максим Паранин', '
Илья Орлов', 'Лида Сидорук', 'Иван Якушев', 'Анастасия Лукина', 'Аня Никончук',
'Илья Березин', 'Евгений Леонов', 'Иван Васильев', 'Тимофей Волков', 'Владислав
Большаков', 'Семён Малюга', 'Саша Михайлова', 'Милослав Волосков', 'Александр Шу
м', 'Brain Storm'], 'Илья Березин': ['Максим Волков', 'Макар Курбатов', 'Игорь К
```

6. Визуализация графа с помощью библиотеки Networkx.

```
import networkx as nx
import matplotlib.pyplot as plt

G=nx.Graph()
dic=graf_of_id
for key in dic:
    for neighbour in dic[key]:
        G.add_edge(key,neighbour)

nx.draw(G)
plt.show()
```



7. Распределение связей. Функция get_links. Работу функции можно разбить на несколько этапов:
   1) Для всех ключей в графе добавляет в new пару [<количество связей>, <ключ>]

```
Администратор@TATYANNA /d/Anaconda3 (master)
$ python lab_6.py
[[19, 'Виталик Алпеев'], [22, 'Кристина Эм'], [1, 'Дмитрий Смирнов'], [16, 'Дмит
рий Коваль'], [10, 'Юлия Ковтун'], [9, 'Илона Черногод'], [5, 'Екатерина Пономар
ёва'], [7, 'Екатерина Ошурок'], [19, 'Игорь Кузнецов'], [16, 'Анастасия Лукина']
, [6, 'Наталья Ястребова'], [2, 'Гриша Барсуков3'], [18, 'Иван Якушев'], [13, 'В
аря Лукина'], [9, 'Маша Громова'], [8, 'Оганес Манасян'], [19, 'Аня Никончук'],
[23, 'Макар Курбатов'], [3, 'Илья Макамбаев'], [17, 'Даша Осетник'], [2, 'Кристи
на Кондратьева'], [16, 'Илья Березин'], [12, 'Алина Сыртланова'], [6, 'Владислав
 Эм2'], [7, 'Максим Аверин'], [3, 'Роман Палонен'], [6, 'Ника Кириллина'], [18,
'Тимофей Волков'], [14, 'Максим Волков'], [15, 'Brain Storm'], [16, 'Ника Поляко
ва'], [16, 'Гриша Барсуков2'], [10, 'Эрик Муртазин'], [1, 'Анюта Павлова'], [6,
'Nastya Kuzmina'], [17, 'Алина Филатова'], [9, 'Вера Некрасова'], [20, 'Максим П
аранин'], [3, 'Егор Васильев'], [3, 'Владислав Эм'], [1, 'Гриша Барсуков'], [6,
'Иван Васильев'], [1, 'Карина Калиниченко'], [3, 'Дмитрий Мухин'], [14, 'Анастас
ия Борщ'], [7, 'Елена Юрьева'], [17, 'Евгений Леонов'], [17, 'Семён Малюга'], [1
4, 'Анастасия Черногорова'], [7, 'Александра Поганшева'], [9, 'Ксения Игнатьева'
], [4, 'Юлия Ким'], [3, 'Катерина Орлова'], [6, 'Анастасия Сергеевна'], [2, 'Ана
стасия Чубарь'], [10, 'Анна Ишмаева'], [0, 'Ростик Юденок'], [16, 'Александр Шум
'], [3, 'Антон Костенко'], [14, 'Сергей Савинов'], [7, 'Anya Rogozha'], [6, 'Мар
ия Климова'], [2, 'Павел Алексеев'], [17, 'Александра Поляничко'], [22, 'Лида Си
дорук'], [0, 'Виталий Протопопов'], [9, 'Виктория Кучаева'], [17, 'Милослав Воло
сков'], [6, 'Игорь Стеклов'], [9, 'Степан Шибаев'], [18, 'Илья Орлов'], [4, 'Мар
ия Зеленская'], [3, 'Марина Ляшко'], [8, 'Василина Третьякова'], [6, 'Игорь Стек
лов2'], [0, 'Елизавета Чешина'], [2, 'Александр Павлов'], [14, 'Владислав Больша
ков'], [10, 'Саша Михайлова']]
```

2) В диапазоне от 0 до количества ключей в графе, т.к. это максимальное возможное число связей у человека. Проходится по всем парам в new и добавляет в словарь new2 в качестве ключа, если такого там нет, <количество связей>, а в качестве значения <ключ>. Если же он там есть, то только добавляет ключ в значение.

```
{0: ['Виталий Протопопов', 'Елизавета Чешина', 'Ростик Юденок'], 1: ['Анюта Павл
ова', 'Дмитрий Смирнов', 'Гриша Барсуков', 'Карина Калиниченко'], 2: ['Павел Але
ксеев', 'Александр Павлов', 'Анастасия Чубарь', 'Гриша Барсуков3', 'Кристина Кон
дратьева'], 3: ['Роман Палонен', 'Марина Ляшко', 'Антон Костенко', 'Владислав Эм
', 'Илья Макамбаев', 'Катерина Орлова', 'Дмитрий Мухин', 'Егор Васильев'], 4: ['
Мария Зеленская', 'Юлия Ким'], 5: ['Екатерина Пономарёва'], 6: ['Игорь Стеклов',
 'Ника Кириллина', 'Владислав Эм2', 'Анастасия Сергеевна', 'Иван Васильев', 'Мар
ия Климова', 'Игорь Стеклов2', 'Наталья Ястребова', 'Nastya Kuzmina'], 7: ['Алек
сандра Поганшева', 'Anya Rogozha', 'Максим Аверин', 'Екатерина Ошурок', 'Елена Ю
рьева'], 8: ['Оганес Манасян', 'Василина Третьякова'], 9: ['Вера Некрасова', 'Кс
ения Игнатьева', 'Виктория Кучаева', 'Илона Черногод', 'Маша Громова', 'Степан Ш
ибаев'], 10: ['Саша Михайлова', 'Эрик Муртазин', 'Юлия Ковтун', 'Анна Ишмаева'],
 12: ['Алина Сыртланова'], 13: ['Варя Лукина'], 14: ['Сергей Савинов', 'Владисла
в Большаков', 'Анастасия Черногорова', 'Анастасия Борщ', 'Максим Волков'], 15: [
'Brain Storm'], 16: ['Илья Березин', 'Дмитрий Коваль', 'Александр Шум', 'Гриша Б
арсуков2', 'Ника Полякова', 'Анастасия Лукина'], 17: ['Семён Малюга', 'Милослав
Волосков', 'Даша Осетник', 'Евгений Леонов', 'Алина Филатова', 'Александра Полян
ичко'], 18: ['Илья Орлов', 'Тимофей Волков', 'Иван Якушев'], 19: ['Виталик Алпее
в', 'Аня Никончук', 'Игорь Кузнецов'], 20: ['Максим Паранин'], 22: ['Кристина Эм
', 'Лида Сидорук'], 23: ['Макар Курбатов']}
```

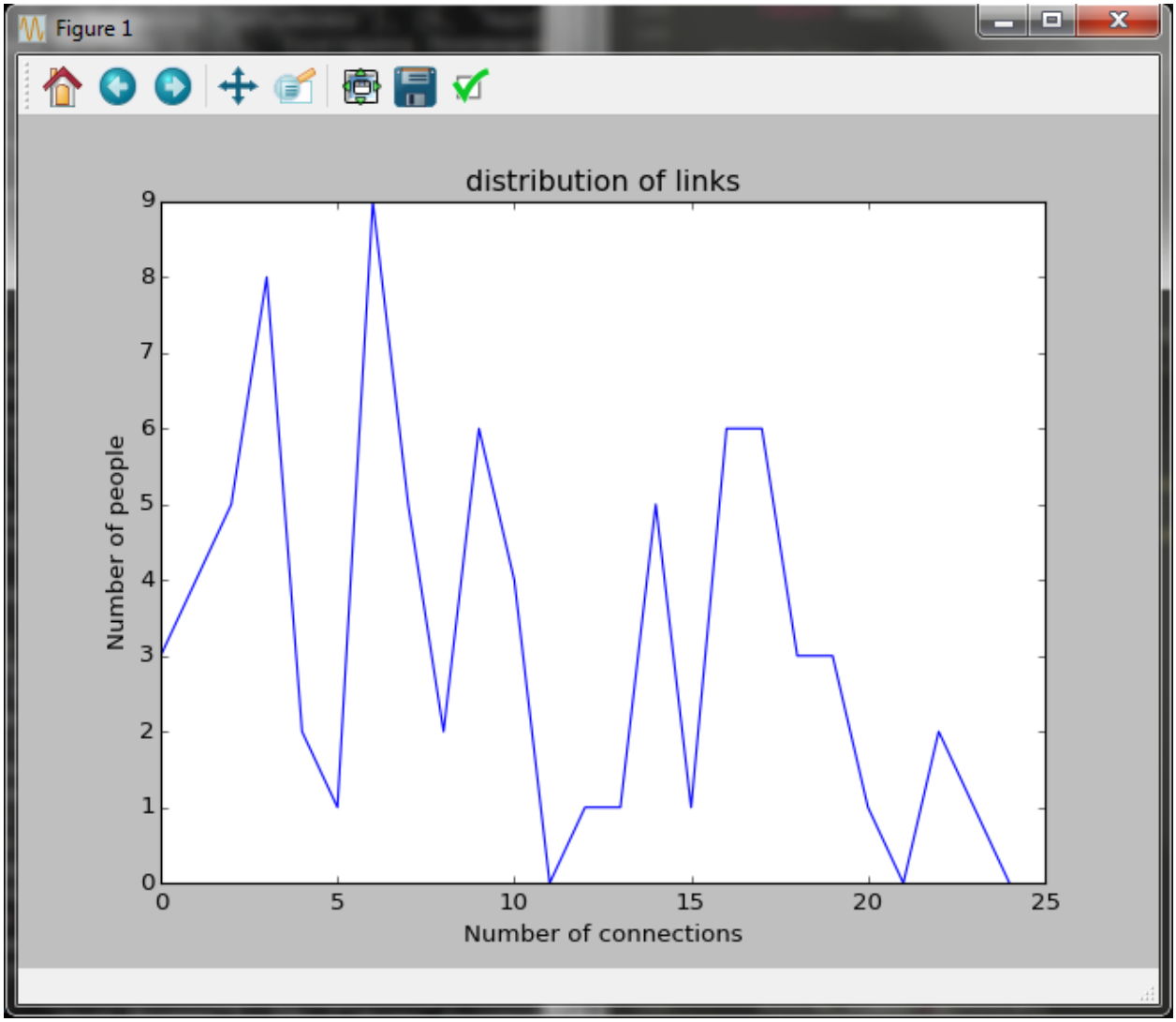
3) Создаёт копию словаря new2 new3, затем заменяет в словаре значение на кол-во элементов в предыдущем значении.

```
{0: [3], 1: [4], 2: [5], 3: [8], 4: [2], 5: [1], 6: [9], 7: [5], 8: [2], 9: [6],
 10: [4], 12: [1], 13: [1], 14: [5], 15: [1], 16: [6], 17: [6], 18: [3], 19: [3]
, 20: [1], 22: [2], 23: [1]}
```

```python
def get_links(graf,n):
#распределение связей
#словарь колличества связей
    new=[]
    new2={}
    for i in graf:
        new.append([len(graf[i]),i])
    #print(new)
    for j in range(len(new)):
        if new[j][0] not in new2:
            new2[new[j][0]]=[new[j][1]]
        else:
            new2[new[j][0]].append(new[j][1])
    if n==0:
        return new2
    if n==1:
        new3=new2
        for k in new3:
            new3[k]=[len(new3[k])]
    return new3

distribution_of_links=get_links(graf_of_names,1)
#distribution_of_links=get_links(graf_of_names,0)
print(distribution_of_links)
```

8. Создаёт график, в котором абсцисса – это количество связей, ордината – количество человек с такой связью.

```
import pylab

def plot_line():
    x=[]
    y=[]
    for i in range(25):
        x.append(i)
        if i in distribution_of_links:
            y.append(distribution_of_links[i][0])
        else:
            y.append(0)
    dx=1
    pylab.plot(x,y)
    pylab.title('distribution of links')
    pylab.ylabel('Number of people')
    pylab.xlabel('Number of connections')
    pylab.show()
    return '\0'

print(plot_line())
```



9. Поиск в ширину (BFS)

```
def enqueue(G,i):
    G.append(i)

def dequeue(G):
    return G[1:]

def for_number(number):
    number+=1
    return number

def for_path(d,finish):
    path=[]
    d2={}
    finish_number=d[finish]
    for i in d:
        d2[d[i]]=i
    finish_number=finish_number//10
    while finish_number>=1:
        if finish_number in d2:
            path.append(d2[finish_number])
            finish_number=finish_number//10
        else:
            finish_number=finish_number//10
    path.append('')
    path=path[::-1]
    path.append(finish)
    return '/'.join(path)

def BFS(G,start,finish):
    Q=[]
    d={}
    number=1
    count=len(G)
    enqueue(Q,start)
    start=Q[0]
    d[start]=number
    while Q:
        if count<0 or G[finish]==[]:
            return 'not path'
        start=Q[0]
        number=for_number(number)
        for i in G[start]:
            if i!=finish:
                enqueue(Q,i)
                number2=int(str(d[start])+str(number))
                number=for_number(number)
                if i not in d:
                    d[i]=number2
                count-=1
            else:
                number2=int(str(d[start])+str(number))
                d[i]=number2
                count-=1
                return for_path(d,finish)
        Q=Q[1:]

print(BFS(graf_of_names,'Аня Никончук','Екатерина Ошурок'))
```

До цикла добавляем в очередь наш start и добавляем его уникальный номер в словарь d. На каждом этапе цикла проверяем не нашли ли мы наш finish, если нет, то берём из очереди первый элемент (с индексом 0), добавляем всех его потомков в очередь Q, и добавляем для каждого его уникальный номер в словарь, укорачиваем очередь на 1 элемент. Если мы всё-таки нашли наш finish, добавляем его уникальный номер в d и возвращаем значение функции for_path. Также есть счётчик count, равный длине графа, для остановки цикла на случай, если связи не будет.

Перед тем как перейти к описанию for_path следует рассказать об уникальных номерах. Start всегда будет присвоен номер 1 (т.к. number будет равен 1) , остальным, в порядке очереди будет присваиваться номер родителя + число number, после каждого присваивания number увеличивается на 1. Таким образом, можно найти предка для каждого ключа, кроме start. Функция for_path принимает словарь d и finish. Эта функция добавляет в свой новый список path finish. Далее каждый раз укорачивает номер и ищет совпадения в списке d, если есть, добавляет, пока не дойдёт до start.

Список d и результат работы функции:

```
{'Илья Березин': 112, 'Максим Паранин': 15, 'Максим Волков': 12, 'Игорь Кузнецов
': 14, 'Анастасия Лукина': 111, 'Иван Васильев': 114, 'Аня Никончук': 1, 'Макар
Курбатов': 13, 'Brain Storm': 120, 'Лида Сидорук': 17, 'Виталик Алпеев': 18, 'Эр
ик Муртазин': 1224, 'Александр Шум': 119, 'Милослав Волосков': 118, 'Тимофей Вол
ков': 115, 'Екатерина Ошурок': 1337, 'Семён Малюга': 117, 'Владислав Большаков':
 116, 'Гриша Барсуков2': 110, 'Евгений Леонов': 113, 'Илья Орлов': 16, 'Иван Яку
шев': 19}
```

```
Администратор@TATYANNA /d/Anaconda3 (master)
$ python lab_6.py
/Аня Никончук/Макар Курбатов/Екатерина Ошурок
```

10. Объединение и пересечение графов.

```python
g1={2:[6,7,9],6:[2],7:[2,9],9:[2,7]}
g2={1:[8,9,4,5],4:[1,5],5:[1,4,9],8:[1],9:[1,5]}

def union_grafs(graf1,graf2):
#объединяет графы
    for i in graf2:
        if i not in graf1:
            graf1[i]=graf2[i]
        else:
            for j in graf2[i]:
                if j not in graf1[i]:
                    graf1[i].append(j)
    return graf1

print(union_grafs(g1,g2))

def crossing_grafs(graf1,graf2):
#возвращает граф с пересечением графов
    new={}
    for i in graf2:
        if i in graf1:
            new[i]=[graf1[i],graf2[i]]
    return new

print('\n')
print(crossing_grafs(g1,g2))
```

```
Администратор@TATYANNA /d/Anaconda3 (master)
$ python lab_6.py
{1: [8, 9, 4, 5], 2: [6, 7, 9], 4: [1, 5], 5: [1, 4, 9], 6: [2], 7: [2, 9], 8: [
1], 9: [2, 7, 1, 5]}


{8: [[1], [1]], 1: [[8, 9, 4, 5], [8, 9, 4, 5]], 4: [[1, 5], [1, 5]], 5: [[1, 4,
 9], [1, 4, 9]], 9: [[2, 7, 1, 5], [1, 5]]}
```

11. Кластеры. Кластером считается список из людей, которые знакомы между собой. Первая функция cluster создаёт один кластер, вторая clusters создаёт список из кластеров.

```
def cluster(graf,vertex,x):
    claster=[]
    count=0
    claster.append(vertex)
    index_x=graf[vertex].index(x)
    claster.append(graf[vertex][index_x])
    for i in list_of_names:
        for j in claster:
            #print(j)
            #print(list_of_names[i][0],'  :  ',graf[j])
            if list_of_names[i][0] in graf[j]:
                count+=1
        if count==len(claster):
            claster.append(list_of_names[i][0])
        count=0
    return claster

def clusters(graf):
    list1=[]
    for i in list_of_names:
        for j in graf[list_of_names[i][0]]:
            #print(list_of_names[i][0],j)
            list2=cluster(graf,list_of_names[i][0],j)
            list1.append(list2)
    for j in list1:
        if len(j)<=2:
            list1.remove(j)
    return list1,len(list1)

print(cluster(graf_of_names,'Аня Никончук','Анастасия Лукина'))
#print(clusters(graf_of_names))
```

Результат работы функции cluster для 'Аня Никончук' и 'Анастасия Лукина' :

```
Администратор@TATYANNA /d/Anaconda3 (master)
$ python lab_6.py
['Аня Никончук', 'Анастасия Лукина', 'Максим Волков', 'Максим Паранин', 'Макар К
урбатов', 'Тимофей Волков', 'Игорь Кузнецов', 'Илья Березин', 'Иван Якушев', 'Br
ain Storm', 'Илья Орлов', 'Лида Сидорук']
```

Кусочек результата работы функции clusters, в конце написано количество кластеров:

```
 Савинов', 'Анастасия Черногорова', 'Дмитрий Коваль', 'Кристина Эм', 'Даша Осетн
ик', 'Александра Поляничко', 'Варя Лукина', 'Ника Полякова'], ['Анастасия Борщ',
 'Оганес Манасян', 'Дмитрий Коваль', 'Кристина Эм', 'Алина Сыртланова', 'Даша Ос
етник'], ['Анастасия Борщ', 'Анастасия Черногорова', 'Дмитрий Коваль', 'Кристина
 Эм', 'Алина Сыртланова', 'Даша Осетник', 'Александра Поляничко', 'Варя Лукина',
 'Ника Полякова'], ['Анастасия Борщ', 'Варя Лукина', 'Анастасия Черногорова', 'Д
митрий Коваль', 'Кристина Эм', 'Алина Сыртланова', 'Даша Осетник', 'Александра П
оляничко', 'Ника Полякова'], ['Анастасия Борщ', 'Алина Сыртланова', 'Анастасия Ч
ерногорова', 'Дмитрий Коваль', 'Кристина Эм', 'Даша Осетник', 'Александра Поляни
чко', 'Варя Лукина', 'Ника Полякова'], ['Анастасия Борщ', 'Юлия Ковтун', 'Анаста
сия Черногорова', 'Кристина Эм', 'Алина Сыртланова', 'Даша Осетник', 'Александра
 Поляничко', 'Варя Лукина'], ['Анастасия Борщ', 'Владислав Эм2', 'Дмитрий Коваль
', 'Кристина Эм', 'Александра Поляничко', 'Ника Полякова'], ['Анастасия Борщ', '
Анастасия Сергеевна', 'Юлия Ковтун', 'Кристина Эм', 'Даша Осетник', 'Александра
Поляничко'], ['Анастасия Борщ', 'Юлия Ким', 'Кристина Эм', 'Варя Лукина', 'Ника
Полякова'], ['Дмитрий Мухин', 'Анна Ишмаева', 'Алина Филатова', 'Вера Некрасова'
], ['Дмитрий Мухин', 'Вера Некрасова', 'Алина Филатова', 'Анна Ишмаева'], ['Дмит
рий Мухин', 'Алина Филатова', 'Анна Ишмаева', 'Вера Некрасова'], ['Илья Макамбае
в', 'Вера Некрасова', 'Алина Филатова'], ['Илья Макамбаев', 'Гриша Барсуков2', '
Алина Филатова'], ['Илья Макамбаев', 'Алина Филатова', 'Вера Некрасова']], 741)
```