ГБОУ "Президентский ФМЛ № 239"

Автоматизированный поиск страниц в ВКонтакте группы людей, социально связанных друг с другом.

Отчет о годовом проекте по информатике.

Работу выполнил
Ученик 10-1 класса
Заварин Андрей

Санкт-Петербург
2017

1. Постановка задачи.

Программа должна получить из файла список имен и фамилий людей, социально связанных друг с другом (например, учащихся в одном лицее) и также второй список имен и фамилий нескольких людей (возможно, одного человека) из первого списка с указанием id их страниц в Вконтате. Имея возможность отправлять запросы к vk api, программа в результате своей работы должна найти страницы людей из первого списка и вывести эту информацию в выходной файл.

2. Уточнение исходных и выходных данных и ограничений на них

2.1. Исходные данные

На вход программе поступает два файла, имена которых указываются в меню запуска.

2.1.1 Первый файл содержит список имен и фамилий людей, страницы которых необходимо найти. На каждой строке указывается информация об одном человеке в формате: **Фамилия Имя** (через пробел), где **Фамилия**, **Имя** – соответственно фамилия и имя задаваемого человека(строки, состоящие из строчных и заглавных букв русского алфавита).
Гарантируется, что в файле каждый человек указан не более одного раза, также что в нем не указаны два разных человека с одними и теми же фамилией и именем.

2.1.2 Второй файл содержит список людей(возможно, одного человека) из первого списка с указанием id их(его) страниц(ы) в Вконтакте. На каждой строке указывается информация об одном человеке из списка в формате: **Имя Фамилия, id**, где **Имя**, **Фамилия** – соответственно имя и фамилия задаваемого человека(строки, состоящие из строчных и заглавных букв русского алфавита), **id** – натуральное число, равное id его страницы в Вконтакте.
Гарантируется, что каждый человек, указанный в этом файле, указан и в первом, также, что никакой человек не указан в этом файле более одного раза.

2.1.3 Также присутствует возможность задать отношение «эквивалентности» между именами из входных файлов и именами со страничек Вконтакте. Это отношение не имеет всех свойств отношения эквивалентности как математического термина, а именно транзитивности. Под ним подразумевается способ сравнения программой имен из входных файлов и со страничек Вконтакте. По умолчанию, будет проверятся только точное совпадение этих имен(считая «ё» и «е» одной и той же буквой). Но при указании того, некоторые сокращения имен или различные варианты написания будут считаться одним и тем же именем. Например, имя Саша будет считаться эквивалентным имени Александр и Александра, Софья – Софии и т.д.(но при

этом, Александр и Александра не будут считаться одним и тем же именем). Это отношение задается файлом, который может быть пустым, имя которого также указывается в меню запуска. Этот файл содержит информацию об отношении «эквивалентности» в следующем формате: на каждой строке через запятую с пробелом(«**,** ») указывается список имен(строк, состоящих из строчных и заглавных букв русского алфавита). Каждые два имени из этого списка будут считаться эквивалентными.
По умолчанию в папке с программой уже имеется стандартный файл с эквивалентными именами. Хотя информация из него не является необходимой для работы программы, он обязан существовать, хотя и может быть пустым.

Все входные файлы имеют кодировку UTF-8 (без BOM).

2.2. Выходные данные

Программа должна вывести в файл, имя которого задается в меню запуска, соответствие найденных страничек в Вконтакте и людей из входного списка в следующем формате: **Имя Фамилия, Ссылка**, где **Имя**, **Фамилия** – соответственно имя и фамилия задаваемого человека из входного списка(строки, состоящие из строчных и заглавных букв русского алфавита), а **Ссылка** – ссылка на его страницу в Вконтакте в формате **vk.com/id<id>**, где **<id>** – id его страницы в Вконтакте(натуральное число). Формат был выбран таким, так как, если отправить текст файла в сообщениях вк, то ссылки автоматически станут активными и появится возможность смотреть превью профиля, не переходя по ним. Люди из второго входного списка, то есть те, чьи страницы Вконтакте уже были заданы, также должны быть указаны в выходном списке.
Рекомендуется, чтобы этот файл был пустой, т. к. программа пишет результат в конец этого файла.
Кодировка выходного файла – UTF-8.

Также при завершении своей работы программа в интерактивной режиме, позволяющим менять масштаб и перемещать картинку, выводит на экран визуализацию графа, вершины которого соответствуют найденным страницам, а ребро между двумя вершинами присутствует тогда и только тогда, когда эти две странички находятся в друзьях друг у друга.

3. Математическая модель (описание алгоритма)

3.1 Поиск страничек

Предлагается решать задачу следующим образом.
Разберемся сначала, что следует из того, что люди из входного списка «социально связаны» друг с другом. Под этим подразумевается, что эти люди чем-то сплочены вместе(например, учатся в одном лицее), а значит, у каждого человека из этого множество будет достаточно много знакомых их этого же

множества, причем социальный граф знакомства(вершины – люди, ребра между знакомыми людьми) этих людей будет связным и не будет содержать «компонент», мало связанных друг с другом.
Обычно знакомые друг с другом люди находятся в друзьях друг у друга в Вконтакте.
Таким образом, основная идея алгоритма поиска страничек этих людей: идти неким подобием BFS по графу друзей от начальных заданных страничек, переходя только в те странички, которые принадлежат кому-то из заданного списка людей(проверяя принадлежность совпадением имени и фамилии человека), заодно и сохраняя полученную информацию о принадлежности в выходной файл.
У этого подхода есть существенная проблема: при достаточно большом количестве людей во входном списке (несколько сотен человек), начинают попадаться случайные знакомые людей из списка, имеющие такие же имя и фамилию, как и какие-то другие люди из входного списка. Алгоритм никак не учитывает эту ситуацию, и помечает эти странички неправильно. Но то, что эти люди являются случайными знакомыми и не принадлежат той социальной группе людей, странички которых нужно найти, означает, что, скорее всего, у этих людей будет мало знакомых из этой группы, в отличие от членов этой группы.
Поэтому возникает идея ранжировать странички по количеству друзей из заданной социальной группы и рассматривать в первую очередь странички с большим их количеством.

Таким образом, итоговый алгоритм следующий:
Будем в порядке приоритетной очереди обрабатывать странички.
В этой очереди вместе с каждой страничкой будет хранится человек, которому, вероятней всего (среди остальных людей из входного списка), принадлежит эта страничка. Этот человек будет определяться при добавлении очередной странички в очередь.
Изначально в очередь добавляются несколько заданных во входных данных страничек с бесконечным приоритетом (считая, что их немного, и они заданы вручную точно) (и соответственно, информация о людях, владеющих ими).

При обработке очередной страницы делаем следующее:
Сначала сохраним соответствие id страницы и человека, владеющего ей, в выходной файл. Затем загрузим список страничек, находящихся в друзьях у этой, и информацию о них (имена и фамилии, указанные на страничках).
Увеличим у каждой из них счетчик количества друзей, уже помеченных как странички кого-то из входного списка, на 1 и пересчитаем их приоритет, обновив их позицию в приоритетной очереди.
Если очередная страничка уже была поставлена в соответствие кому-то, то пропустим ее.
Если очередная страничка еще не была ни разу рассмотрена в качестве чьего-либо друга, то найдем человека из входного списка, который вероятней всего

владеет этой страничкой, и, если он есть (с ненулевой вероятностью владеющий), то добавим ее в приоритетную очередь, посчитав приоритет. После того, как очередь опустела, мы уже записали найденные соответствия в выходной файл, поэтому остается только визуализировать граф друзей для найденных страничек. Сделаем это с помощью библиотек networkx и matplotlib.

Поиск человека, который скорее всего владеет страничкой, происходит следующим образом:
Просто просматриваем всех людей из списка, для кого еще не найдена(не записано соответствие в результат) страничка, среди них рассматриваем людей только с точно совпадающей фамилией, среди них, если такой есть, берем человека с точно совпадающим именем, иначе с именем, являющимся эквивалентным имени со странички. Если такого человека нет, то говорим об этом.

Расчет приоритета происходит следующим образом:
Приоритет просто равен количеству уже поставленных в соответствие страничек из друзей данной.
Если у странички имя не совпадает в точности с именем человека, который скорее всего владеет ею (но является эквивалентным), то вычитаем из приоритета определенную константу (на всякий случай, это не принципиально).

Загрузка списка друзей странички и информации о них происходит следующим образом:
В начале работы программа загружает кешированную информацию о некоторых страничках с диска. Если нужная информация была в кеше, то просто возвращаем ее.
Если нет (или если кеш устарел), то загружаем ее из вк, отправляя запрос к vk api, и сохраняем в кеш на диск (и в оперативную память) информацию об этой страничке (имя, фамилию и список id друзей) и об ее друзьях (только имена и фамилии) (в отдельный файл для каждой странички).

Эквивалентность имен определяется согласно правилам, заданным во входном файле (см. **2.1.3**). Также все имена и фамилии сравниваются без учета различия между буквами «ё» и «е» (считая их равными).

3.2 Меню запуска

Помимо самого скрипта поиска страничек, в проекте есть интерактивное меню его запуска.
Поддерживаемые команды:
help – выводит информацию о поддерживаемых командах
help [command] – выводит информацию о команде [command]
exit – завершает работу меню запуска
start – запускаем скрипт с текущими настройками (см. далее).

curSettings – выводит текущие настройки скрипта.
Они имеют следующие поля:
Equal_names_file – имя файла с отношением эквивалентности имен (см. **2.1.3**)
Students_file – имя файла со списком людей, чьи страницы необходимо найти (см. **2.1.1**)
Start_ids_file – имя файла с начальным соответствием людей и страниц в ВК (см. **2.1.2**)
Result_file_name – имя файла для записи результата работы скрипта (см. **2.2**)

set [name] [value] – присваивает полю [name] настроек значение [value]
save [fileName] – сохраняет текущие настройки в файл [fileName]
load [fileName] – загружает сохраненные настройки из файла [fileName]

4. Анализ используемой структуры данных

В программе используются несколько структур данных:

4.1 Приоритетная очередь

Используется для рассмотрения страниц по убыванию приоритета. Асимптотика работы – *O(log n)* для удаления и добавления одной страницы, где *n* – количество страниц в очереди. Реализуется с помощью встроенной в python кучи (heap). Суммарное время работы очереди – *O(N logN)*, где *N* – количество найденных в итоге страниц. Быстрее нельзя, т. к. иначе бы была возможна сортировка на сравнениях, с асимптотикой лучшей, чем эта.

4.2 Структура для хранения отношения эквивалентности имен

Информация об эквивалентности имен хранится в двухмерной хеш-таблице (хеш-таблице хеш-таблиц) d, такой, что если name1 и name2 эквиваленты, существует элемент d[name1][name2] и d[name2][name1]. Каждый запрос выполняется за *O(1)*, построение требует $O(Sum\ n_i^2)$, где n_i – количество имен на в i-той группе эквивалентных имен. Т.к. запросов к структуре сильно больше этой суммы (n_i невелики), то такая асимптотика является оптимальной. Реализуется с помощью встроенных в python структур dict и set.

Для этого не используется система непересекающихся множеств, т. к. это отношение эквивалентности не транзитивно.

4.3 Структура для хранения списка людей, чьи страницы необходимо найти

Т.к. в алгоритме используется только рассмотрение всех элементов списка за раз, при том используется несколько таких рассмотрений, используется массив, реализуемый встроенным в python списком (list). Асимптотика работы – *O(1)* на одно рассмотрение одного элемента. Построение выполняется за *O(n)*, где *n* – количество элементов списка.

5. Выбор метода решения
Оценим асимптотику и эффективность работы алгоритма:

Пусть N – количество людей во входном файле (предполагаем, что количество найденных страничек растет как $O(N)$). c – среднее количество друзей у людей их входного файла.

Тогда суммарное время работы очереди – $O(N \log N)$.
Как было уже сказано в анализе структур данных, улучшить эту асимптотику нельзя.

Суммарное время поиска соответствий – $O(N^2 * c)$, т. к. для каждой рассматриваемой странички (их – $O(N * c)$) перебираются $O(N)$ людей из входного файла (имена сравниваются за $O(1)$).
Можно было бы применить эвристику, существенно снижающую количество операций: рассматривать только тех людей, у которых фамилия совпадает с фамилией со странички (если заданной фамилии нет в списке людей, то вовсе ничего не рассматривать, а сразу сказать об этом) с помощью хещ-таблицы за $O(1)$ на получение списка таких людей. Но, во-первых такой подход менее универсален и не позволяет легко изменить способ сравнения имен и фамилий, а во-вторых время работы этой части алгоритма относительно мало по сравнению со временем скачивания информации о страничках из ВК (занимает не больше 10% общего времени работы).
Итоговая асимптотика: $O(N^2 * c)$.

Основное время работы программы приходится на отправку запросов к vk api. Чтобы уменьшить это время и ускорить программу, используется, во-первых локальный кеш страничек в оперативной памяти (не загружается дважды одна и та же информация о страничке), во-вторых кеш страничек на диске (см. **3.1**), в-третьих, если можно, отправляются запросы информации о сразу большом количестве страничек (например, при загрузке списка друзей также загружается и необходимая информация о самих друзьях (имя и фамилия)).
Также можно еще ускорить эту часть программы засчет использования метода execute vk api (позволяющего за раз выполнить несколько запросов) или распараллеливания работы с сетью. При этом возникают проблемы с тем, что ВК начинает не отвечать на запросы из-за слишком высокой их часты (которые, впрочем, частично можно решить (подбором частоты и разными приложениями vk api / IP-адресами клиента)). Это не было реализовано в программе.

6. Комментированный листинг

Рекомендуется читать листинг в файле программы с помощью IDE

```
# -!- coding: utf-8 -!-

#импортирование библиотек
import requests
import json

import time
import Queue
import datetime
import os
import networkx as nx
import matplotlib.pyplot as plt
import sys
import heapq

#ключ доступа для прав на просмотр друзей от имени странички
#не обязятелен
access_token = ''

#класс, содержащий информацию об ученике
class Student:
        def __init__(self, firstName, secondName, brd_year, brd, cl, gender):
                self.firstName = firstName      #имя
                self.secondName = secondName    #фамилия
                self.vkId = ''                  #id странички вк

                #эта информация не используется
                self.brd_year = brd_year        #год рождения
                self.brd = brd                  #число и месяц
                self.cl = cl                    #класс
                self.otherVkIds = []            #вспомогательный массив(не
используется)
                self.gender = gender            #пол

        def __init__(self):
                self.firstName = ''
                self.secondName = ''
                self.brd_year = ''
                self.brd = ''
                self.cl = ''
                self.vkId = ''
                self.otherVkIds = []
                self.gender = ''

#класс, содержащий информацию о страничке в вк
class VkVert:
        def __init__(self, data = None, friends = None, was = False, cnt = 0,
id = 0):
                self.data = data          #информация о страничке(словарь,
который возращает vk api)
                self.friends = friends    #список id друзей
                self.cnt = cnt            #вспомогательный счетчик(количество
уже отмеченных друзей)
                self.was = was            #вспомогательный влажок(посещена ли
вершина)
                self.id = id              #id странички

#загрузка списка учеников и информации о них
```

```
def loadStudents(fileName):
        #чтение текста из файла
        f = open(fileName)
        lines = map(lambda x: x.decode('utf-8'), f.readlines())
        f.close()

        #парсинг текста
        students = []
        for line in lines:
                data = line.strip().split(', ')

                #создрание объекта класса Student
                st = Student()

                #парсинг инфомрации про конкретно этого ученика
                st.firstName = data[0].split()[1]
                st.secondName = data[0].split()[0]

                #в первоначальной версии была эта информация об учениках, хотя
она и не использовалась при поиске
                #st.gender = data[1]
                #st.brd_year = data[3].split('-')[0]
                #st.brd = data[3].split('-')[1:]
                #st.cl = data[4]

                #добавление ученика в список
                students.append(st)

        return students

#возращает индекс ученика в списке, имеющего заданные имя и фамилию
def getStIdByName(students, firstName, secondName):
        #перебор учеников
        for i in range(0, len(students)):
                #условие сравнения
                if students[i].firstName == firstName and
students[i].secondName == secondName:
                        return i

        return None


#чтение из файла начальных id страничек и имен людей, владеющих ими, и
загрузка данных этих страничек из вк
def loadStartProfilies(fileName, verts, students):
        #чтение текста из файла
        f = open(fileName)
        lines = f.readlines()
        f.close()

        #построение списка id
        ids = [int(line.split(', ')[1]) for line in lines]
        #построение словаря имен
        names = {int(line.split(', ')[1]):line.split(', ')[0].decode('utf-8')
for line in lines}

        #запрос к vk api для получения данных страничек
        params = {'user_ids' : ','.join(map(str, ids)), 'v' : '5.52', 'fields'
: 'first_name,last_name,bdate,schools', 'access_token' : access_token}
        r = requests.get("https://api.vk.com/method/users.get", params =
params)
        time.sleep(0.5)
        print r.status_code
```

```
        #парсинг ответа из json в словарь
        data = json.loads(r.text)

        #создание соответствующих объектов в словаре страничек
        for user in data[u'response']:
                if user['id'] in verts:
                        continue

                verts[user['id']] = VkVert(data = user.copy(), friends = None,
was = False, id = user['id'])
                firstName, secondName = names[user['id']].split()

        return ids, names

#загружает данные странички из вк с диска(сохраняются туда, чтобы при каждом
запуске заново всё не загружать)
def loadSavedUserData(id, user_data_folder):
        try:
                f = open(user_data_folder + '/' + str(id) + '.txt', 'r')
                data = json.load(f)
                f.close()
                return data
        except:
                return None

#сохраняет данные странички
def saveUserData(user_data_folder, vert, id):
        f = open(user_data_folder + '/' + str(id) + '.txt', 'w')
        data = {'data' : vert.data, 'friends' : vert.friends}
        f.write(json.dumps(data))
        f.close()

#заргужает заданные данные о странички из вк, записывает их в словарь
страничек, и опционально сохраняет на диск
def loadUserData(verts, id, loadFriends = False, loadData = False, reload =
False, user_data_folder = 'vk', save_data = True):

        #если в словаре ничего нет, создаем пустой объект
        if not id in verts:
                verts[id] = VkVert(data = None, friends = None, was = False,
id = id)

        #если нужно загрузить список друзей
        if loadFriends:
                #вывод на экран текста 'Load friends of' и id странички через
пробел
                print 'Load friends of', id

                #пытаемся загрузить данные, пока не получится(вк позволяет
получать некоторую информацию о друзьях(имя, фамилию, день рождения и школу)
сразу при загрузки списка друзей за один запрос)
                params = {'user_id' : str(id), 'v' : '5.52', 'fields' :
'first_name,last_name,bdate,schools', 'access_token' : access_token}
                while True:
                        try:
                                r =
requests.get("https://api.vk.com/method/friends.get", params = params, timeout
= 5)
                                time.sleep(0.5)
                                print r.status_code
                                break
                        except KeyboardInterrupt:
```

```
                                raise
                        except:
                                print ':('

                #парсинг ответа из json в словарь
                data = json.loads(r.text)

                #если что-то пошло не так, выводим на экран ответ
                if not 'response' in data:
                        print data
                        verts[id].friends = []
                        if save_data and user_data_folder != None:
                                        saveUserData(user_data_folder,
verts[id], id)
                        return

                #заполняем список друзей и создаем для них объекты в словаре
страничек с полученной инфомарцией
                verts[id].friends = []
                for user in data['response']['items']:
                        verts[id].friends.append(user['id'])

                        #если такой странички нет в словаре, добавляем
                        if not user['id'] in verts:
                                verts[user['id']] = VkVert(data = user.copy(),
friends = None, was = False, id = user['id'])

                                if save_data and user_data_folder != None:
                                        saveUserData(user_data_folder,
verts[user['id']], user['id'])

        #если нужно заргрузить информацию о страничке
        if loadData:
                print 'Load data of', id

                #пытаемся загрузить данные, пока не получится
                params = {'user_ids' : str(id), 'v' : '5.52', 'fields' :
'first_name,last_name,bdate,schools', 'access_token' : access_token}
                while True:
                        try:
                                r =
requests.get("https://api.vk.com/method/users.get", params = params, timeout =
5)
                                time.sleep(0.5)
                                print r.status_code
                                break
                        except KeyboardInterrupt:
                                raise
                        except:
                                print ':('

                #парсинг ответа из json в словарь
                data = json.loads(r.text)

                #если что-то пошло не так, выводим на экран ответ
                if not 'response' in data:
                        print data
                        verts[id].data = None
                        return

                #сохраняем информацию
                verts[id].data = data['response'][0].copy()
```

```
        #обновляем кеш, если нужно
        if save_data and user_data_folder != None:
                saveUserData(user_data_folder, verts[id], id)

#загружает информацию о страничках сразу нескольких пользователей за один
запрос
def loadUsersData(verts, ids, reload = False, user_data_folder = 'vk',
save_data = True):
        if len(ids) == 0:
                return

        print 'Load data of ' + ', '.join(map(str, ids))

        #пытаемся загрузить данные, пока не получится
        params = {'user_ids' : ','.join(map(str, ids)), 'v' : '5.52', 'fields'
: 'first_name,last_name,bdate,schools', 'access_token' : access_token}
        while True:
                try:
                        r =
requests.get("https://api.vk.com/method/users.get", params = params, timeout =
5)
                        time.sleep(0.5)
                        print r.status_code
                        break
                except KeyboardInterrupt:
                        raise
                except:
                        print ':('

        #парсинг ответа из json в словарь
        data = json.loads(r.text)

        #если что-то пошло не так, выводим на экран ответ
        if not 'response' in data:
                print data
                return

        #записываем данные в словарь
        for user in data['response']:
                id = user['id']

                if not id in verts:
                        verts[id] = VkVert(data = None, friends = None, was =
False, id = id)

                verts[id].data = user.copy()

                #сохраняем на диск, если нужно
                if save_data and user_data_folder != None:
                        saveUserData(user_data_folder, verts[id], id)

#словарь из имени в множество эквивалентынх ему имен
#не используется СНМ, так как отношение не транзитивное
#например имя Саша эквиваленто Александре и Александру, но они не эквиваленты
друг другу
eqNames = {}

#загружает информацию об эквивалентности имен из файла
def loadEqNames(fileName):
        global eqNames

        #чтение текста из файла
        f = open(fileName, 'r')
```

```
        lines = f.readlines()
        f.close()

        #записанные через запятую в одной строчке имена считаются попарно
эквивалентными
        #создание словаря
        for line in lines:
                line = line.decode('utf-8').strip()

                lst = None
                for s1 in line.split(', '):
                        for s2 in line.split(', '):
                                if s1 != s2:
                                        if not s1 in eqNames:
                                                eqNames[s1] = set()
                                        eqNames[s1].add(s2)

#возращает насколько похожи имена
#2 -- полностью совпадают, 1 -- являются эквивалентыми, но не совпадают, 0 --
в остальных случаях
def getEqNames(name1, name2):
        #меняем ё на е, т.к. встречаются разные варианты написания
        name1 = name1.replace(u'ё', u'e')
        name2 = name2.replace(u'ё', u'e')

        #если полностью совпадают
        if name1 == name2:
                return 2
        #если являются эквивалентными
        if name1 in eqNames and name2 in eqNames[name1]:
                return 1

        return 0

#возращает, совпадают ли фамилии
def isEqSecNames(name1, name2):
        #меняем ё на е, т.к. встречаются разные варианты написания
        name1 = name1.replace(u'ё', u'e')
        name2 = name2.replace(u'ё', u'e')
        return name1 == name2


#возращает наиболее вероятного владельца странички из списка учеников(а
насколько точно он им является), либо None, если такого нет
def getStudentIdByVK(students, verts, user):
        if user.data == None:
                return None

        #насколько точно он им является
        pr = 0

        #увеличиваем приоритет на количество друзей странички, уже помеченных
как странички кого-то из спика учеников
        cnt = user.cnt
        pr += cnt

        school = None

        #если есть информация о школе и ученик учится в 239 -- ничего не
делаем
        if 'schools' in user.data:
                for sch in user.data['schools']:
                        school = sch
```

```
                        if sch['id'] == '12':
                                #pr += 7
                                break

        #номер лучшего ученика и степень похожести его имени
        #best_k = 0 гарантирует, что имя хотя бы является эквивалентным
        best = -1
        best_k = 0

        #ищем самого похожего по имени и фамилии ученика среди тех, у кого еще
не определена страничка
        for id in range(0, len(students)):
                st = students[id]

                #если страничка этого ученика уже найдена, пропускаем его
                if st.vkId != '':
                        continue

                #фамилии обязательно должны совпасть
                if isEqSecNames(st.secondName, user.data['last_name']):
                        #смотрим, насколько совпадают имена
                        k = getEqNames(st.firstName, user.data['first_name'])
                        if k > best_k:
                                best_k = k
                                best = id

        #если никого не нашли, возращает пустой кортеж
        if best == -1:
                return (pr, None, school)
        else:
                #если имя не полностью совпадает, уменьшаем приоритет, но не
делаем его слишком малельнким
                if best_k == 1:
                        pr = max(min(pr, 5), pr - 5)
                return (pr, best, school)


#сохраняем в файл с результатом имя ученика и id его странички в удобном
формате ссылки на нее(если отправить текст файла в сообщениях вк, то ссылки
автоматически станут активными и появится возможность смотреть превью профиля,
не переходя по ним)
def saveResForStudent(vert, st, fileName):
        #если известен день рождения, получим его(но он не будет
использоваться в дальнейшем)
        bdate = ''
        if 'bdate' in vert.data:
                bdate = vert.data['bdate']

        #сохраняем запись в конец файла
        f = open(fileName, 'a')
        f.write(st.firstName.encode('utf-8') + ' ' +
st.secondName.encode('utf-8') + ', vk.com/id' + str(vert.id) + '\n')
        f.close()

#загружает сохраненную на диск инфомацию о страничках
def loadSavedData(verts, user_data_folder):
        i = 0
        files = os.listdir(user_data_folder)

        #для каждого файла из папки
        for i in range(len(files)):
                print str(i) + '/' + str(len(files))
```

```
                file = files[i]

                #id странички -- его имя
                id = int(file.split('.')[0])

                #читаем из него данные
                f = open(user_data_folder + '/' + file, 'r')
                data = json.load(f)
                f.close()

                #создаем соответствующий объект в словаре
                verts[id] = VkVert(data = None, friends = None, was = False)

                verts[id].data = data['data']
                verts[id].friends = data['friends']
                verts[id].id = id


def solve():

        #читаем настройки с именами файлов, указанных в меню запуска
        f = open("settings.txt", "r")
        settings = json.load(f)
        f.close()

        #загружаем списки эквивалентных имен
        loadEqNames(settings['Equal_names_file'])

        unknown = {}    #вспомогательный словарь
        verts = {}      #словарь из id страницы в ее объект

        #загружаем сохраненные данные
        loadSavedData(verts, 'vk')

        #загржаем список учеников и сортируем его по фамилии
        students = loadStudents(settings['Students_file'])
        students.sort(key = lambda st: st.secondName)

        #загружаем начальные страницы
        ids, names = loadStartProfilies(settings['Start_ids_file'], verts,
students)

        #приоритетная очередь страничек, которые должны быть расмотрены
        #каждый раз будет расматриваться страница с наивысшим приоритетом
        #в очереди хранятся кортежи (приоритет, id страницы, номер ученика в
списке)
        q = []

        #добавляем стартовые странички с бесконечным приоритетом
        for id in ids:
                name = names[id]
                fName = name.split()[0]
                sName = name.split()[1]

                st_id = getStIdByName(students, fName, sName)

                heapq.heappush(q, (-1e9, id, st_id))

        #пока что-то есть в очереди
        while len(q) > 0:
                #вынимаем элемент с максимальным приоритетом
                el = heapq.heappop(q)
```

```
                #куча на минимум и в ней хрантся отрицательные приоритеты
                #восстанавливаем приоритет
                el = (-el[0], el[1], el[2])

                id = el[1]
                st_id = el[2]

                #если страничка уже была расмотрена (после этого добавления
она была добавлена еще раз с большим приоритетом), то пропускаем ее
                if verts[id].was:
                        continue
                #если этому ученику принадлежит с большей вероятностью какая-
то другая страничка, и это уже было установлено, то пропускаем ее
                if students[st_id].vkId != '':
                        continue
                #если приоритет слишком маленький, то тоже пропускаем
страничку
                if el[0] < 1:
                        continue

                #выводим текущую страничку
                print str(el) + ', ' + str(len(q))

                #сохраняем соответствие страницы и ученика
                saveResForStudent(verts[id], students[st_id],
settings['Result_file_name'])

                #запоминаем, что этот ученик и эта страничка уже поставлены в
соответствие
                verts[id].was = True
                students[st_id].vkId = str(id)

                #если неизвестен список друзей, загрузим его
                if verts[id].friends == None:
                        loadUserData(verts, id, loadFriends = True)

                #список страниц, информацию с которых нужно скачать
                toLoad = []

                #добавляем туда друзей, информации которых еще нет
                for fr_id in verts[id].friends:
                        if not fr_id in verts:
                                verts[fr_id] = VkVert(data = None, friends =
None, was = False, id = fr_id)
                                toLoad.append(fr_id)

                #если что-то нужно загрузить, загружаем
                if len(toLoad) > 0:
                        loadUsersData(verts, toLoad)

                #рассматриваем каждого друга
                for fr_id in verts[id].friends:
                        #если он уже обработан, пропускаем
                        if fr_id in verts and verts[fr_id].was:
                                continue

                        #увеличиваем счетчик количества уже посещенных друзей
этого друга на 1(мы посетили текущую страничку)
                        user = verts[fr_id]
                        user.cnt += 1

                        #смотрим, чья скорее всего эта страничка
                        st = getStudentIdByVK(students, verts, user)
```

```
                        if st == None:
                                continue

                        #сохраняем вспомогательную информацию
                        if st[1] == None:
                                if user.cnt > 5:
                                        if not fr_id in unknown:
                                                unknown[fr_id] = 0

                                        unknown[fr_id] = max(unknown[fr_id],
user.cnt)
                                continue

                        #добавляем страничку в очередь (с отрицательным
приоритетом, т.к. куча на минимум)
                        heapq.heappush(q, (-st[0], fr_id, st[1]))



        f = open('notfound.txt', 'w')

        for st in students:
                if st.vkId == '':
                        f.write((st.firstName + u' ' + st.secondName +
u'\n').encode('utf-8'))

        f.close()

        #*** далее: выводим на экран граф друзей и завершаем работу      ***

        #создаем пустой граф
        G = nx.Graph()
        #строим граф
        for it in verts.items():
                v = it[1]

                #если страничка не принадлежит ученику из списка, пропускаем
ее
                if not v.was or v.friends == None:
                        continue

                #смотрим на друзей и добавляем ребра к ученикам из списка
                for fr_id in v.friends:
                        if not fr_id in verts or not verts[fr_id].was:
                                continue
                        G.add_edge(v.id, fr_id)

        #рисуем граф
        nx.draw(G, node_size=80)
        plt.show()
```

7. Пример работы программы

Пример 1.
Поиск страничек учеников 10-1 класса:
Входные данные:

students.txt:

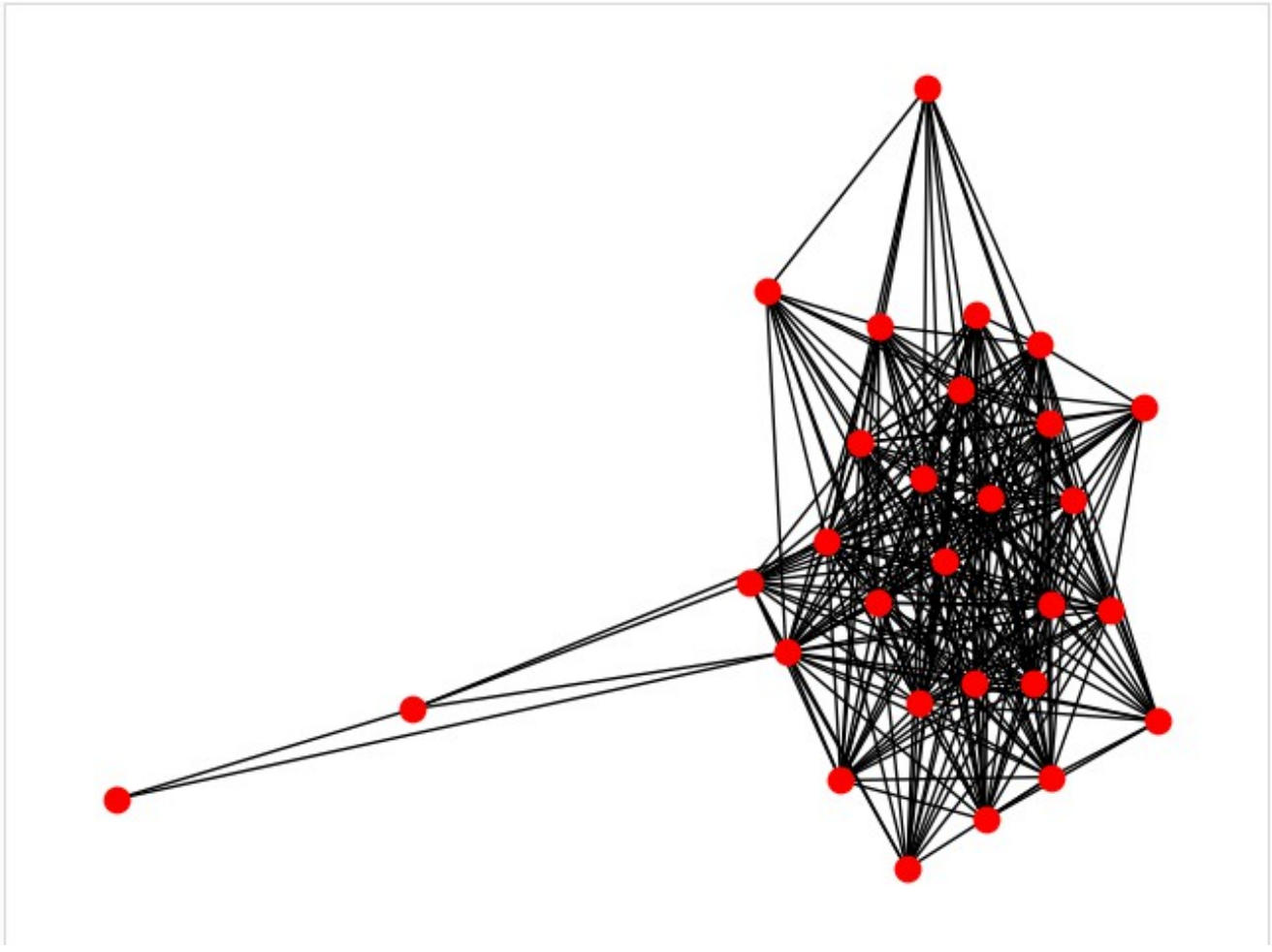
Айнбунд Ася
Анисимов Андрей
Анопренко Михаил
Антонова Анастасия
Бессонов Кирилл
Блащук Роман
Бородулина Дарья
Гнатюк Дмитрий
Горбачев Егор
Дворецкий Семен
Ермейчук Мария
Загретдинов Ильдар
Заварин Андрей
Иванова Александра
Ким Елена
Конева Елизавета
Мельников Олег
Мильшин Владислав
Никитюк Борис
Новикова Александра
Олейник Иван
Родин Иван
Сарафанников Даниил
Сафонов Иван
Симарова Юлия
Скрипка Дмитрий
Соловьев Владимир
Суворова Екатерина
Толокно Изабелла
Щенников Григорий

start_ids.txt:
Ася Айнбунд, 12560124

Выходные данные:
Ася Айнбунд, vk.com/id12560124
Иван Сафонов, vk.com/id271061482
Даниил Сарафанников, vk.com/id255019394
Юлия Симарова, vk.com/id211929533
Изабелла Толокно, vk.com/id196971550
Дарья Бородулина, vk.com/id185223440
Мария Ермейчук, vk.com/id135887294
Дмитрий Скрипка, vk.com/id110524586
Михаил Анопренко, vk.com/id95498376
Андрей Анисимов, vk.com/id12627876

Иван Олейник, vk.com/id230520575
Ильдар Загретдинов, vk.com/id40580520
Григорий Щенников, vk.com/id98097069
Александра Иванова, vk.com/id263188129
Олег Мельников, vk.com/id99314280
Елена Ким, vk.com/id161523000
Дмитрий Гнатюк, vk.com/id52760951
Александра Новикова, vk.com/id36279748
Владислав Мильшин, vk.com/id64880318
Анастасия Антонова, vk.com/id38088563
Семен Дворецкий, vk.com/id223505277
Егор Горбачев, vk.com/id132834512
Екатерина Суворова, vk.com/id44522096
Иван Родин, vk.com/id137604152
Владимир Соловьев, vk.com/id147702612
Андрей Заварин, vk.com/id171165131
Борис Никитюк, vk.com/id95174550
Роман Блащук, vk.com/id253352316
Кирилл Бессонов, vk.com/id364377555

Граф друзей:



Найдено: 29/30

Пример 2.
Поиск страничек нескольких небольших групп случайных учеников из разных классов 239:

Входные данные:

students.txt:

Ананов Георгий
Андреев Яков
Буберман Алексей
Вирачев Арсений
Думпис Сергей
Евтушевская Ариадна
Жуков Матвей
Оранский Станислав
Пенская Таисия
Петров Степан
Петров Федор
Седов Денис
Смирнова Анастасия
Сухорукова Екатерина
Чугреева Марина
Шилов Антон
Юрьев Михаил
Александров Алексей
Бобылёва Наталия
Бондарев Евгений
Бочаров Глеб
Скрипка Дмитрий
Соловьев Владимир
Суворова Екатерина
Толокно Изабелла
Абрамов Евгений
Агеев Роман
Харичкин Иван
Цветков Георгий
Шаймуханов Тимур
Арония Мариони
Баженов Иван
Бурмистрова Анна
Буянов Александр
Владимирцев Дмитрий

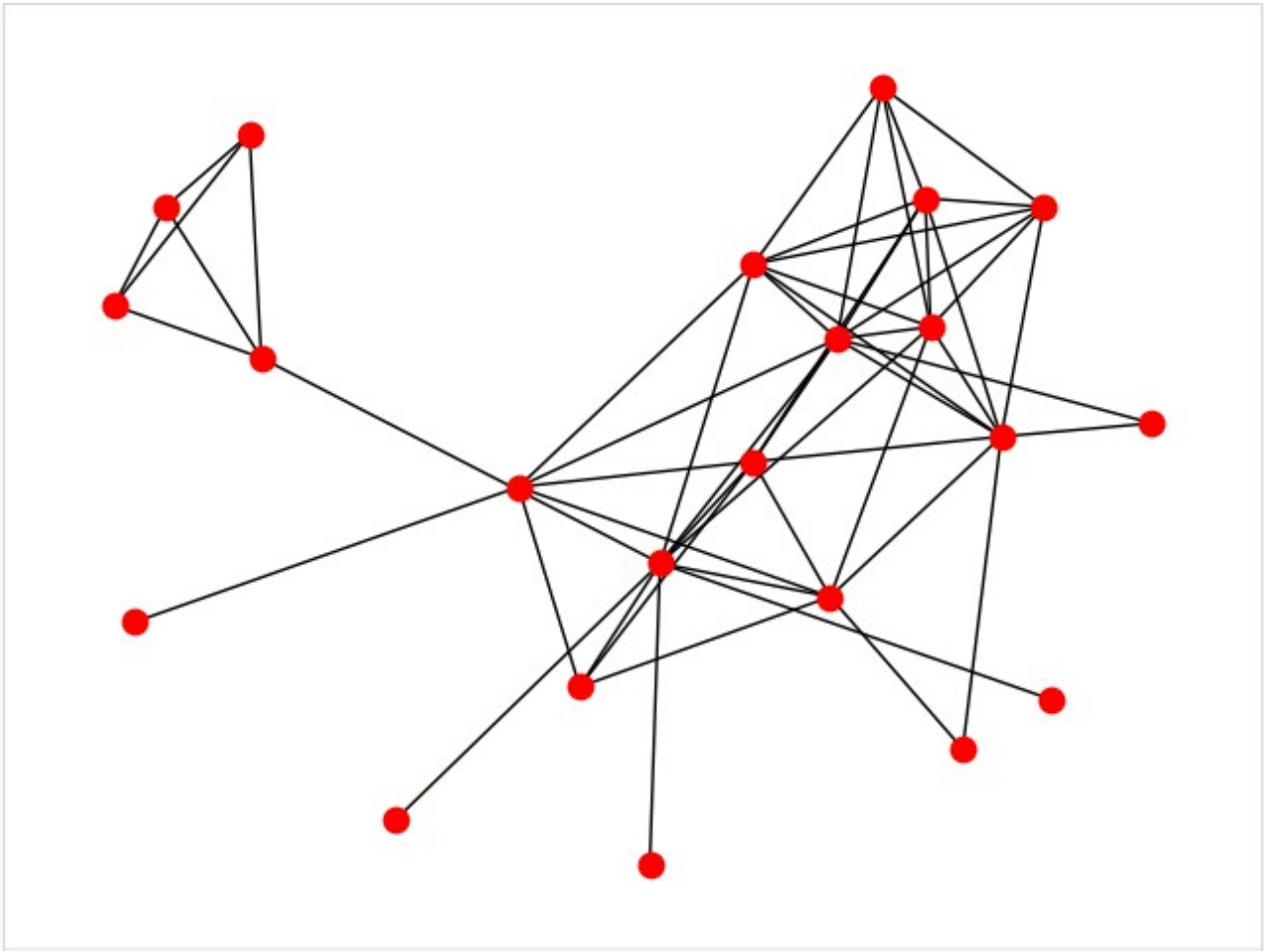
start_ids.txt:

Изабелла Толокно, 196971550

Выходные данные:

Изабелла Толокно, vk.com/id196971550
Владимир Соловьев, vk.com/id147702612
Дмитрий Скрипка, vk.com/id110524586
Екатерина Суворова, vk.com/id44522096
Сергей Думпис, vk.com/id38481243
Анастасия Смирнова, vk.com/id127021747
Федор Петров, vk.com/id94643239
Степан Петров, vk.com/id94642914
Екатерина Сухорукова, vk.com/id71115248
Станислав Оранский, vk.com/id157014438
Марина Чугреева, vk.com/id104570574
Таисия Пенская, vk.com/id62907304
Денис Седов, vk.com/id44167241
Наталия Бобылёва, vk.com/id5346476
Алексей Александров, vk.com/id46574360
Евгений Бондарев, vk.com/id58448929
Глеб Бочаров, vk.com/id72416867
Михаил Юрьев, vk.com/id101627481
Антон Шилов, vk.com/id44723266
Анна Бурмистрова, vk.com/id88730895
Алексей Буберман, vk.com/id53374413
Ариадна Евтушевская, vk.com/id23778886

Граф друзей:



Найдено: 22/35

Пример 3.
Поиск страничек нескольких случайных групп учеников параллели 10 классов 239:

Входные данные:

students.txt:

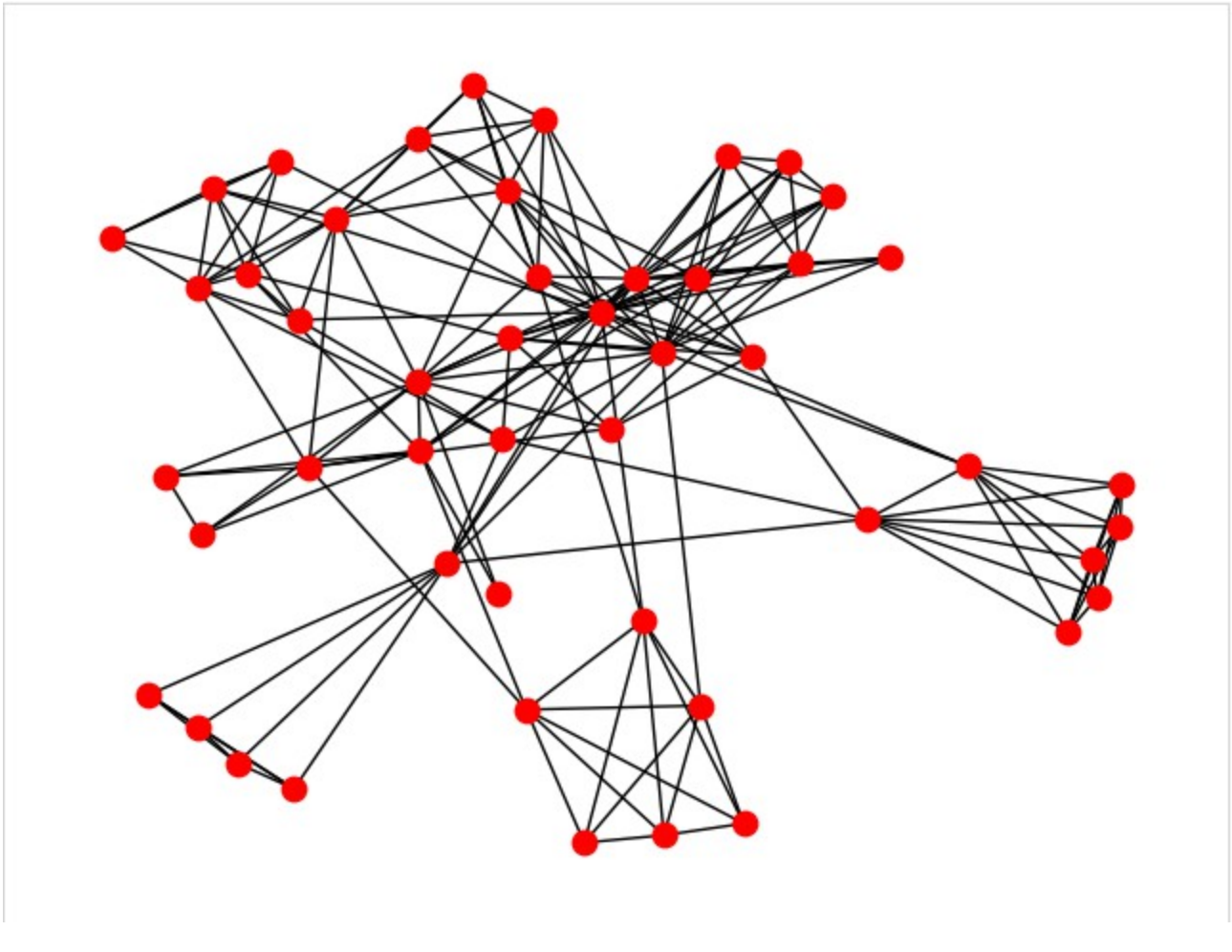
Айнбунд Ася
Анисимов Андрей
Анопренко Михаил
Гнатюк Дмитрий
Щенников Григорий
Дав Яков
Филиппов Степан
Чижиков Савелий
Чумарикова Лолита
Абдурахманов Рауль
Бей Святослав

Берхман Евгений
Ведь Олеся
Еремеев Иван
Иванова Марианна
Мартыненко Анастасия
Попелышко Аким
Ширшин Даниил
Воробьев Александр
Горбунов Арсений
Демичев Даниил
Рылов Денис
Сенов Михаил
Сорокин Илья
Аминджанов Руслан
Бондаренко Михаил
Бузина Екатерина
Гальчич Маргарита
Сумароков Александр
Усачёв Михаил
Дубровина Варвара
Загребин Иван
Зайцев Георгий
Зинин Дмитрий
Лямин Владимир
Матюшова Ольга
Мельник Лев
Насонов Иван
Джиблави Ибрагим
Зимин Сергей
Калайда Анна
Кандратович Федор
Караборчев Алексей
Ушаков Александр
Фомичев Даниил
Козловцева Мария
Крылова Анна
Лашина Маргарита
Минюк Елизавета
Салаватов Вадим
Юдин Алексей

start_ids.txt:
Ася Айнбунд, 12560124

Выходные данные:

Ася Айнбунд, vk.com/id12560124
Степан Филиппов, vk.com/id267969928
Дмитрий Гнатюк, vk.com/id52760951
Григорий Щенников, vk.com/id98097069
Михаил Анопренко, vk.com/id95498376
Андрей Анисимов, vk.com/id12627876
Яков Дав, vk.com/id267909138
Савелий Чижиков, vk.com/id169204229
Олеся Ведь, vk.com/id43605412
Лолита Чумарикова, vk.com/id154768909
Анастасия Мартыненко, vk.com/id35738804
Михаил Усачёв, vk.com/id177316763
Марианна Иванова, vk.com/id138163617
Даниил Ширшин, vk.com/id78686210
Святослав Бей, vk.com/id135400137
Иван Еремеев, vk.com/id236734072
Евгений Берхман, vk.com/id137640821
Рауль Абдурахманов, vk.com/id53672562
Аким Попелышко, vk.com/id118237598
Маргарита Гальчич, vk.com/id5715434
Екатерина Бузина, vk.com/id73197364
Александр Сумароков, vk.com/id166188383
Михаил Бондаренко, vk.com/id136805953
Руслан Аминджанов, vk.com/id105118972
Даниил Демичев, vk.com/id150117817



Александр Воробьев, vk.com/id72021747
Арсений Горбунов, vk.com/id263381892
Александр Ушаков, vk.com/id106559571
Михаил Сенов, vk.com/id31864461
Денис Рылов, vk.com/id183363503
Илья Сорокин, vk.com/id212494651
Анна Калайда, vk.com/id124591914
Ольга Матюшова, vk.com/id190180986
Дмитрий Зинин, vk.com/id21346577
Мария Козловцева, vk.com/id224454609
Вадим Салаватов, vk.com/id23685262
Елизавета Минюк, vk.com/id123319365
Алексей Юдин, vk.com/id109893971
Маргарита Лашина, vk.com/id64158654
Анна Крылова, vk.com/id89672871
Иван Загребин, vk.com/id213991716
Владимир Лямин, vk.com/id182637808
Иван Насонов, vk.com/id156607095
Георгий Зайцев, vk.com/id99692162
Лев Мельник, vk.com/id34309160
Ибрагим Джиблави, vk.com/id377741170
Алексей Караборчев, vk.com/id306192026
Даниил Фомичев, vk.com/id48344051
Сергей Зимин, vk.com/id35382773

Граф друзей:
Найдено: 49/51.

8. Анализ правильности решения

Будет оценивать правильность решения по примерам работы из пункта **7**.

Несложно убедиться, что в примере 1 все найденные страницы были найдены правильно: просто посмотрим вручную на эти страницы. Единственная ненайденная страница принадлежит Лизе Коневой. Она не найдена лишь потому, что имя и фамилия указаны на страничке латиницей (vk.com/id272426112, Elizaveta Koneva).
Все страницы, которые можно было найти без учета сильного различия написания имен и фамилий на странице, были найдены.
Из общих соображений ясно, что ученики одного класса будут образовывать достаточно плотный граф. Видно, что это так, на графе друзей. Такая ситуация является наиболее благоприятной для работы алгоритма.
Этот пример показывает правильность реализации алгоритма, но не его эффективность (с точки зрения количества найденных страниц). Хотя было найдено 97% людей, из примера не следует, что на более больших группах, на

которые был рассчитан алгоритм, будет достигаться такой же результат.

Пример 3 показывает, что на более больших группах людей, содержащих в себе более тесно связанные (относительно самой группы) подгруппы, алгоритм также хорошо работает. Для работы на примерно такой же ситуации и был рассчитан алгоритм. Были найдены все страницы, кроме двух. Забегая вперед, скажем, что эти две страницы не были найдены и при поиске всех учеников лицея, так что, возможно, их либо нет в ВК, либо у них на страницах указаны не те имена (или указаны латиницей).
Правильность нахождения страниц можно проверить, посмотрев вручную некоторые из них.
Этот пример показывает эффективность самого алгоритма.

Пример 2 показывает, в каких ситуациях алгоритм работает плохо. Там программа не нашла никого из присутствующих в списке людей из двух определенных классов. Причем они составляют большую часть ненайденных людей. Здесь граф друзей (если его построить для всех страниц людей из списка), скорее всего, не связен (можно проверить, посмотрев на друзей некоторых ненайденных страниц), поэтому эти страницы и не могли быть найдены таким способом – люди списка не соответствуют требованиям для работы алгоритма.

Также с небольшими модификациями алгоритм запускался для поиска страничек всех учеников 239 и выпускников прошлого года. Начальная страничка была одна: страничка Аси Айнбунд (vk.com/id12560124). Из 985 людей из списка были найдены странички 848, т. е. примерно 86%. О правильности нахождения судить сложнее, но можно посмотреть на несколько случайных страничек вручную. В целом, кажется, что они найдены весьма точно. Эти странички искались для возможности вставки ссылок на людей, празднующих день рождения, в поздравляшках 239 (vk.com/pozdravlashki239).

Таким образом, для данных, полностью соответствующих условиям задачи, алгоритм находит почти все страницы. На реальных данных он находит страницы также достаточно большой части людей. При этом, в результат не попадают страницы случайно встреченных людей с теми же фамилией и именем, и ошибки с ложным определением страницы практически не встречаются.