МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ

УЧРЕЖДЕНИЕ

ВЫСШЕГО ОБРАЗОВАНИЯ

**«Национальный исследовательский ядерный университет «МИФИ»**

**(НИЯУ МИФИ)**

**Предуниверситарий НИЯУ МИФИ**

**Выпускная работа обучающегося IT-класса предуниверситария НИЯУ МИФИ**

«Telegram-бот, оповещающий о возможном заражении вирусом»

Обучающиеся Рыженко Роман Владиславович, 11И класс

Сниховский Роман Тимурович, 11И класс

Научный руководитель Резник Максим Семёнович

Должность, степень, звание Студент НИЯУ МИФИ

Дата защиты: 16.02.2023

Результат защиты:

Москва 2022

**Реферат**

*Аннотация:* Вирусы стремительно распространяются из-за небрежного отношения людей к этому факту. Для того, чтобы люди стали заботиться о своём здоровье, необходимо оповестить их о том, что они могут быть заражены – так сказать, «подтолкнуть» к действиям.

Реализация осуществлена при помощи Telegram-бота, базы данных в виде excel-таблиц и платформы (сервер, название сайта)

Отчет состоит из 17 страниц, содержит скриншоты файлов и пояснительные рисунки.

*Ключевые слова:* Telegram bot, pyTelegramBotApi, python, pandas, numpy, сервер, (название сервера, ну и что ещё с ним связано, хз), графы, кротчайший путь, алгоритм Дейкстры

Оглавление

[Реферат](#page2) [2](#page2)

[Введение](#page4) [4](#page4)

[Связанные работы](#page6) [6](#page6)

[Основная часть проекта](#page9) [7](#page9)

[Описание](#page9) бота [8](#page9)

[Структура](#page9) базы данных [1](#page11)1

Сервер [11](#page11)

Алгоритм расчёта пересечений в коридорах [13](#page11)

Алгоритм Дейкстры [1](#page11)5

[Заключение](#page18) [18](#page18)

**Введение**

В наши дни всё более острой становится проблема болезней. Возрастают вспышки гриппа и ОРВИ, вирус COVID-19 мутирует и перестаёт быть безопасным для людей с прививками. ПО исследованиям учёных риск повторного заражения этим вирусом высок для тех, кто переболел или сделал прививку. Один из факторов, из-за которых болезни распространяются и делают это так быстро – это непринятие своевременно мер по защите собственного здоровья. Человек может заразиться и не подозревать об этом, пока болезнь будет усиливаться в его организме. Необходимо как можно раньше узнать о возможном заражении для предотвращения такого варианта развития событий или во избежание действительного заражения после принятия профилактических мер.

Система оповещения о заражении человека, которая обеспечит уменьшение распространения вируса благодаря информированности людей, - очень актуальная проблема в современном мире. Решением этой проблемы может послужить telegram-бот.

Telegram-бот — это «приложение внутри приложения», которое позволяет пользователям совершать разные действия через мессенджер. Оно помогает пользователям автоматизировать рутину, сэкономить время и упростить жизнь. Боты доступны 24/7, работает по принципу «вопрос-ответ», обеспечивая моментальный ответ пользователю, удобны в использовании. Мессенджер telegram также очень популярен: в июне 2022 число постоянных пользователей превысило 700млн, и на некоторых телефонах он даже предустановлен при покупке, так что не требуется установка дополнительных программ, приложений, чтобы иметь возможность пользоваться ботом, ведь его использование ведётся напрямую через мессенжер. Поэтому использование telegram-бота - очень удобное и практичное решение.

Целью нашего проекта является уменьшение риска распространения вируса в учебных заведениях путём информирования учеников о возможном заражении.

Задача проекта – разработка инструмента, который позволит информировать учеников о потенциальном заражении. Это способствует уменьшению риск распространения вируса. Реализовать этот инструмент решено при помощи telegram-бота.

Выполнение этой задачи было поделено на следующие этапы:

1. Разработка каркаса бота – основные команды.

2. Формирование баз данных учащихся и их расписаний.

3. Проверка на то, что ученик обучается в данной школе.

4. Добавление функциональности (кнопки и др.).

5. Возможность выбора времени заражения.

6. Рассчёт, кому должно будет прийти уведомление о заражении.

7. Получение уведомлений.

8. Серверная часть.

9. Дополнительные функции бота.

Связанные работы

Существуют различные системы оповещения о появлении новых или других вспышек заболеваний. Но также важно понимать, когда именно ты можешь быть подвергнут заболеванию.

Одним из аналогов проекта является приложение «Госуслуги. COVID трекер». Все смартфоны, на которых установлено приложение, обмениваются данными по Bluetooth. Программа ищет и запоминает информацию с устройств в радиусе 10 метров. Она присылает push-уведомление каждый раз, когда человек сталкивается с инфицированным.

В сравнении с Telegram-ботом это приложение имеет минусы: его нельзя установить на пк, требуется постоянное включение Bluetooth, приложение оповестит только о заболевших Covid’ом.**Основная часть проекта**

Задача проекта – разработка бота, который позволит информировать учеников о потенциальном заражении.

Для написания telegram-бота был выбран язык python, так как он прост в понимании, благодаря чему мы можем скорее перейти от изучения языка к самой разработке кода, а также он имеет огромное количество написанных библиотек, модулей, применяющихся повседневно.

Существует множество библиотек для работы с API Telegram, для языка программирования Python существуют следующие: python-telegram-bot, pyTelegramBotAPI, AIOGram, TGramBot, OrigamiBot, pytgbot.

И для работы с telegram api была выбрана библиотека pyTelegramBotAPI как самая популярная библиотека за её обширные возможности и удобность их использования.

Сюда же можно влепить и про сервер, но, возможно, лучше упомянуть его позже на отдельной странице, чтобы было больше объёма.

**Описание бота**

1. Бот должен уметь обрабатывать команду оповещения - распознать команду и выполнить соответствующие действия, а именно: при получении сообщения от ученика о том, что он заражён, необходимо определить, с кем он мог пересекаться и оповестить всех этих людей. Также заражённому будут предоставлены некоторые рекомендации по борьбе с вирусом.
2. Чтобы определить, с кем мог пересекаться заражённый, необходимо знать расписание учеников. Оповестить нужно тех, кто учился в день заболевания вместе с заражённым и кто учился в том же кабинете, что и заражённый в этот день.
   1. Расписание учеников возможно реализовать в виде таблицы в базе данных. Но поскольку расписание – это не большой набор данных, в котором придётся ориентироваться, таблицы можно реализовать не на основе баз данных, а в обычных текстовых файлах формата .csv . Для формирования таблиц из файла .csv и работы с этими таблицами в проекте была применена библиотека pandas.
3. Бот предназначается для пользования учениками учебных заведений, поэтому для проверки, учится ли человек в этом образовательном учреждении, а также определения ученика перед использованием бота необходимо пройти регистрацию. Чтобы зарегистрироваться, нужно ввести свои фамилию, имя и номер класса вместе с буквой.
4. Для проверки бот имеет ещё один .csv файл со списком всех обучающихся в данном учебном заведении, распределённых по классам.
5. Для упрощения и повышения эффективности работы с ботом необходимы кнопки. В Telegram они бывают двух видов: Inline и Keyboard – они отличаются своим расположением. Первый тип кнопок располагается прямо под сообщением, к которому они относятся, второй же тип кнопок располагается под полем для ввода текста для отправки сообщения. В данном боте применены кнопки второго типа. Кстати, мы написали «Добавление функциональности (кнопки и др.)». А что это др. значило, не помнишь? Или просто так? Просто я не помню, а если что-то ещё было, то можно в этот пункт дописать
6. Чтобы ограничить круг оповещаемых только теми людьми, кому действительно нужно прислать оповещение, не «пугая лишний раз» остальных, пользователю предоставлена функция ввода дня, в который он заразился.
7. Уведомить о заражении необходимо только тех, кто действительно пересекался с заражённым, поэтому необходимо рассчитать эти пересечения.
   1. Наибольшую вероятность заражения имеют те, кто учился вместе с заражённым. Необходимо оповестить его одноклассников.
   2. Подвержены заражению также могут быть те, кто учился в том же кабинете, что и заражённый, в тот же день после него.
   3. Заражённый также может и пересекается с другими людьми не только в кабинетах. Возможно найти кратчайшие пути между двумя кабинетами и оповестить тех, у кого кратчайший путь пересечётся с кратчайшим путём заражённого. Алгоритмы для этого описаны далее в этом документе. Да, мы их не реализовали, но чтобы написать побольше, я могу их описать.
8. Стоп, а мы этого не делали, да («Реализовать получение уведомлений»)? А что писать тогда тут… И что вообще под этим подразумевается? Сообщения ведь в телеге по умолчанию с уведом приходят. Я как-то даже не подумал раньше, что этот пункт означает
9. Здесь про сервер напиши. Я предлагаю не описывать всё о сервере здесь, а просто введение какое-то, а работу его уже описать отдельно дальше. Ну или как, не знаю, я просто так и не читал ту статью – организуй описание сервера по своему усмотрению.
10. На данном этапе бот будет готов к работе, но возможно дополнить его функционал. Например:
    1. Анти-спам система. Ни к чему присылать уведомления о заражении несколько раз подряд. Причём анти-спам система работает в обе стороны: как один пользователь не сможет присылать сообщение о заражении несколько раз подряд, так и один пользователь не будет получать сообщение о ещё одном заражении, если он уже был оповещён.
    2. Счётчик количества заболевших за день. Чтобы отметить важность мер предосторожности и дезинфекции, можно посылать пользователем информацию о количестве заболевших за текущий день.
    3. Если ещё есть идеи, можно добавить. Не обязательно даже всё это реализовывать, думаю - пофиг. Хотя ещё можно что-то успеть даже.

**Структура базы данных**

Базы данных хранятся в виде файлов .csv. Имеются два файла:

* Расписание учеников
* Список учеников

В первом файле хранится информация о дне недели, номере урока и кабинете, в котором проходит урок, для каждого класса. Номера классов с буквой класса – атрибуты, день недели, номер урока – кортежи, а домен – номера кабинетов.

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как стол

Автоматически созданное описание

Во втором файле хранится информация об учениках: фамилия и имя, номер класса с буквой, а также дополнительная информация для работы бота: личный ID в таблице в качестве первичного ключа, момент последнего оповещения (отправки и получения – два разных столбца), а также ID аккаунта в Telegram. Всё это – атрибуты таблицы, кортежи – каждый новый ученик, экземпляр.

**Изображение выглядит как стол

Автоматически созданное описаниеИзображение выглядит как текст

Автоматически созданное описание**

**Алгоритм расчёта пересечений в коридорах**

Для расчёта пересечений в коридорах будет применена теория графов и алгоритмы для работы с графами.

Чтобы представить школьное здание в виде графа, его вершинами будут кабинеты, а рёбрами будут коридоры, лестницы – пути между кабинетами. Таким образом получится связный неориентированный взвешенный (нагруженный) граф.

Переведя школьное здание на язык графов, можно работать с ним по правилам математики, используя известные алгоритмы для работы с ними.

Для нахождения в графе кротчайших путей между вершинами существуют различные алгоритмы. Поскольку в конкретной задаче граф взвешенный, причём имеет только положительные веса, а также требуется вычисление не расстояния, а именно пути, для решения подойдёт алгоритм Дейкстры.

Алгоритм Дейкстры в простейшем случае имеет сложность O(n2). Учитывая, что в школе обычно не более 100 кабинетов, алгоритм выполнится меньше, чем за секунду. Даже в случае с большим количеством вершин в графе работа бота не будет нарушена, так как алгоритм необходимо выполнить всего 1 раз – при загрузке расписания, а дальше использовать уже запомненные пути.

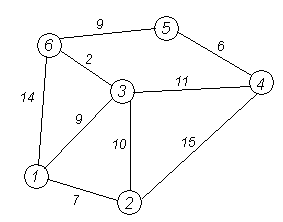
Сохраняться все пути будут в виде матрицы смежности, элементами которой будут сами кротчайшие пути, записанные в виде строк.

Алгоритм Дейкстры вычисляет кротчайший путь от одной вершины до всех. Поскольку необходимо знать пути между всеми вершинами, придётся применить его для всех вершин, кроме одной, так как из имения кротчайшего пути от каждой вершины до какой-то одной следует имение кротчайшего пути от этой вершины до всех остальных (так как кротчайший путь от вершины A до вершины B соответствует кротчайшему пути от вершины B до вершины A). Чтобы уменьшить время работы алгоритма, можно для каждой новой вершины считать пути до всех вершин кроме тех, которые уже участвовали в алгоритме (так как кротчайший путь от вершины A до вершины B соответствует кротчайшему пути от вершины B до вершины A). Но в усложнении алгоритма нет необходимости, потому как, как уже было сказано, алгоритм будет выполняться всего 1 раз, а затем пересечения будут сохранены, так что при повторение алгоритма n раз, то есть получение сложности O(n3), не повлияет на основную работу бота.

**Алгоритм Дейкстры**

Рассмотрим выполнение алгоритма на примере графа, показанного на рисунке.

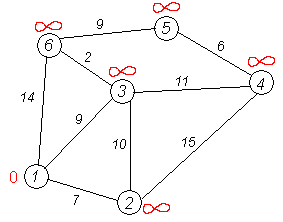
Пусть требуется найти кратчайшие расстояния от 1-й вершины до всех остальных.

[](https://commons.wikimedia.org/wiki/File:Dijkstra_graph0.PNG?uselang=ru)

Кружками обозначены вершины, линиями — пути между ними (рёбра графа).

В кружках обозначены номера вершин, над рёбрами обозначен их вес — длина пути.

Рядом с каждой вершиной красным обозначена метка — длина кратчайшего пути в эту вершину из вершины 1.

[](https://commons.wikimedia.org/wiki/File:Dijkstra_graph1.PNG?uselang=ru)

Выполнять алгоритм будем в цикле до тех пор, пока на текущей итерации цикла будут изменения.

**Первая итерация цикла.**

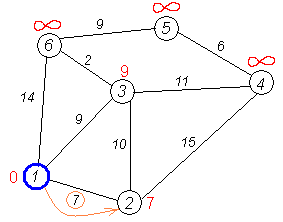
**Первый шаг**.

Минимальную метку имеет вершина 1. Её соседями являются вершины 2, 3 и 6.

Первый по очереди сосед вершины 1 — вершина 2, потому что длина пути до неё минимальна.

Длина пути в неё через вершину 1 равна сумме значения метки вершины 1 и длины ребра, идущего из 1-й в 2-ю, то есть 0 + 7 = 7.

Это меньше текущей метки вершины 2, бесконечности, поэтому новая метка 2-й вершины равна 7.

[](https://commons.wikimedia.org/wiki/File:Dijkstra_graph3.PNG?uselang=ru)

Аналогичную операцию проделываем с двумя другими соседями 1-й вершины — 3-й и 6-й.

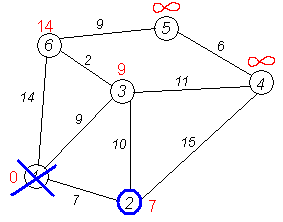
Все соседи вершины 1 проверены.

Текущее минимальное расстояние до вершины 1 считается окончательным и пересмотру на данной итерации цикла не подлежит.

Вычеркнем её из графа, чтобы отметить, что эта вершина посещена.

**Второй шаг**.

Снова находим «ближайшую» из непосещённых вершин. Это вершина 2 с меткой 7.

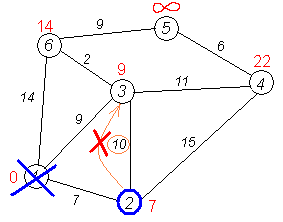
[](https://commons.wikimedia.org/wiki/File:Dijkstra_graph7.PNG?uselang=ru)

Снова пытаемся уменьшить метки соседей выбранной вершины, пытаясь пройти в них через 2-ю вершину. Соседями вершины 2 являются вершины 1, 3 и 4.

Первый (по порядку) сосед вершины 2 — вершина 1. Но она уже посещена, поэтому с 1-й вершиной ничего не делаем.

Следующий сосед — вершина 3, так как имеет минимальную метку.

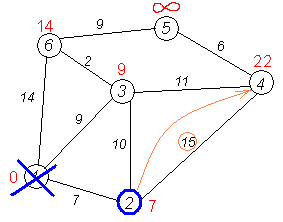
Если идти в неё через 2, то длина такого пути будет равна 17 (7 + 10 = 17). Но текущая метка третьей вершины равна 9, а это меньше 17, поэтому метка не меняется.

[](https://commons.wikimedia.org/wiki/File:Dijkstra_graph9.PNG?uselang=ru)

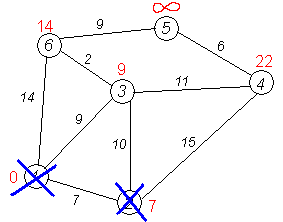
Ещё один сосед вершины 2 — вершина 4.

Если идти в неё через 2-ю, то длина такого пути будет равна сумме кратчайшего расстояния до 2-й вершины и расстояния между вершинами 2 и 4, то есть 22 (7 + 15 = 22).

Поскольку 22 < ∞∞, устанавливаем метку вершины 4 равной 22.

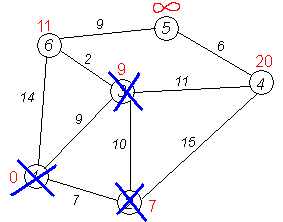
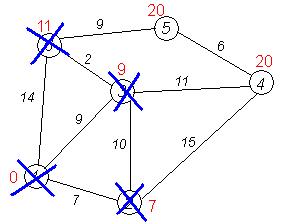
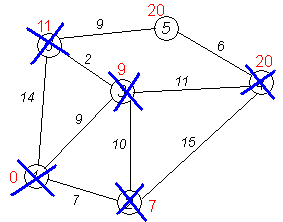
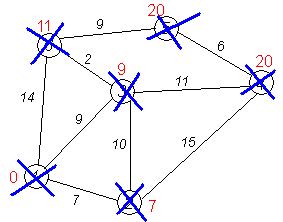
[](https://commons.wikimedia.org/wiki/File:Dijkstra_graph8.PNG?uselang=ru)

Все соседи вершины 2 просмотрены, замораживаем расстояние до неё и помечаем её как посещённую.

[](https://commons.wikimedia.org/wiki/File:Dijkstra_graph10.PNG?uselang=ru)

**Дальнейшие шаги первой итерации**.

Повторяем шаг алгоритма для оставшихся вершин. Это будут вершины 3, 6, 4 и 5, соответственно порядку.

[](https://commons.wikimedia.org/wiki/File:Dijkstra_graph11.PNG?uselang=ru)[](https://commons.wikimedia.org/wiki/File:Dijkstra_graph12.PNG?uselang=ru) [](https://commons.wikimedia.org/wiki/File:Dijkstra_graph13.PNG?uselang=ru) [](https://commons.wikimedia.org/wiki/File:Dijkstra_graph14.PNG?uselang=ru)

**Последующие итерации цикла.**

Убираем зачёркивания каждой вершины. Повторяем те же шаги, что и на первой итерации.

**Завершение выполнения алгоритма**.

Алгоритм заканчивает работу, когда на текущей итерации цикла не будет произведено изменений меток.

Результат работы алгоритма виден на последнем рисунке: кратчайший путь от вершины 1 до 2-й составляет 7, до 3-й — 9, до 4-й — 20, до 5-й — 20, до 6-й — 11.

**Заключение**

В данной работе был разработан бот для пользования одним учебным заведением. Нетрудно расширить функционал для пользования несколькими учреждениями: для каждого создавать свои .csv-файлы и работать с ними, либо же для каждого создать своего бота. Также бот применим (с небольшими правками) не только в рамках образовательного учреждения, но и, например, в лагерях или других организациях, где у каждого имеется расписание.

Также здесь применён несовершенный принцип определения пересечений учеников, вычислять можно также и другими способами, что поспособствует точности определения и большему уменьшению распространения вирусов.

Необходимо отметить, что данный бот может быть в разной степени эффективен в разных учреждениях / в разное время в зависимости от расписания. Чем больше людей будет находиться в одном кабинете, тем больше оповещений будет осуществлено из-за одного заражённого в том же кабинете, но тем меньше оповещений из-за заражённого, не бывавшего в этом кабинете. Также если кто-то весь день будет в одном и том же кабинете или в каких-то рядом стоящих обособленных кабинетах, то будет осуществлено минимальное количество оповещений.