УДК 519.17 **Технические науки**

Ильичев Владимир Юрьевич, *научный руководитель, к.т.н., доцент кафедр «Тепловые двигатели и гидромашины» и «Мехатроника и робототехнические системы»*

Калужский филиал ФГОУ ВО «Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет), г. Калуга, Россия

Сафронова Мария Евгеньевна, *магистрант кафедры «Экология и промышленная безопасность»*

Калужский филиал ФГОУ ВО «Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет), г. Калуга, Россия

РАЗРАБОТКА МЕТОДИКИ ИСПОЛЬЗОВАНИЯ БИБЛИОТЕКИ GRAPHVIZ ДЛЯ СОЗДАНИЯ НАПРАВЛЕННЫХ ГРАФОВ

Аннотация: В статье приведена разработанная авторами методика автоматизированого создания направленных графов при помощи библиотеки Graphviz для языка программирования Python, а в частности с использованием основного модуля данной библиотеки Digraph. Показан процесс вывода на экран логического кода созданного графа. Произведён расчёт нескольких разнотипных примеров, наглядно демонстрирующих применение графов как при решении задач, так и при отображении их результатов. Отмечены преимущества и ограничения разработанной методики, приведены отрасли её предпочтительного использования.

Ключевые слова: теория графов, библиотека Graphviz, логическое кодирование, язык Python, модуль Digraph.

Annotation: The article presents the technique developed by the authors to

automatically create directed graphs using the Graphviz library for the Python programming language, and in particular using the main module of this Digraph library. Shows the process of displaying the logical code of the created graph. Several different types of examples were calculated, which clearly demonstrate the use of graphs both in solving problems and in displaying their results. The advantages and limitations of the developed methodology were noted, branches of its preferred use were given.

Keywords: graph theory, Graphviz library, logical coding, Python language, Digraph module.

Введение

Графом называется изображение (схема), формируемое с помощью условно обозначаемых прямоугольных или овальных вершин (узлов - nodes), соединяемых стрелками (рёбрами - edges), представляющих из себя логические связи между вершинами [1]. Теория графов является основополагающей в математике и связанных с ней науках благодаря следующим особенностям [2]:

- обладает наглядностью и отчётливо отображает практические задачи и связи между объектами;

- имеет чётко определённые правила отображения;

- обладает простотой в изображении и при этом свой круг пока не решённых задач.

Особенно актуальной является теория графов в современном высокотехничном обществе, отличающимся огромным количеством взаимосвязей между объектами. В наука она также является очень востребованной; достаточно привести такой широко используемый сейчас метод как построение нейросетевых моделей [3], собственно и представляющих из себя графы с разной «прочностью» (удельным весом) связей нейронов-узлов. Также локальные корпоративные компьютерные сети и сеть Интернет по сути принято изображать в виде графов, по которым можно проследить путь прохождение информации от одного пункта (сервера) к другому [4]. Теория

графов часто используется и в таких областях, как биология [5], электротехника и энергетика вообще, химия, социология, экономика и во многих других.

Такое распространение применения графов связано с наличием взаимосвязей между различными объектами и явлениями, которые наиболее наглядно можно изобразить в виде схем, а не формул.

Естественным поэтому является возникновение потребности автоматизации построения таких схем по результатам исследований баз данных, математических расчётов.

Целью данной работы являлась разработка кода программы на свободно распространяемом языке Python [6], с применением появившейся также бесплатной специальной библиотеки функций Graphviz [7], позволяющей строить графы разного вида, в том числе с численным и текстовым обозначением узлов и рёбер, а также с отображением графов в различном наглядном виде, наиболее подходящем для иллюстрации исследований в публикациях, презентациях и прочих визуализированных материалах [8]. Код используемых в статье библиотек функций является достаточно сложным, хотя само их применение доступно для освоения всеми желающими. Далее рассмотрено несколько примеров применения методики построения графов с помощью указанных программных средств.

Материал и методы исследования

Для создания графа любого вида необходимо произвести набор некоторых действий:

1. Из библиотеки Graphviz импортировать модуль Digraph (для создания направленных рёбер) или Graph (в случае использования ненаправленных рёбер). Остальная часть кода в-основном состоит из создания и связывания между собой объектов данными типами рёбер на так называемом языке DOT.

2. Создание узлов, каждый из которых отличается своим названием и текстовой меткой.

3. Создание группы рёбер (массива), каждое из которых соединяет указанные точки, либо так называемой границы, связывающей только две точки

(и над которой можно разместить необходимую надпись).

4. Вывести на экран созданный код DOT в текстовом (логическом) виде.

5. Произвести сохранение созданного кода и произвести его рендер в формат PDF с помощью библиотеки Grapviz.

На рис. 1 показан граф самой простой конфигурации, но созданный с применением указанных выше средств.

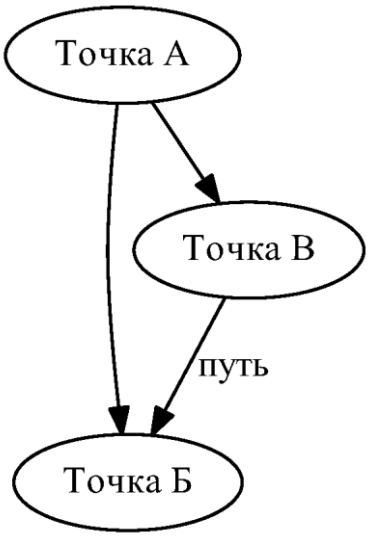

Рис. 1. Простой граф, созданный с помощью описанной методики.

Узлами данного графа являются точки А, Б, В, рёбрами – направленные линии (стрелки), соединяющие точки А и Б, А и В, границей – поименованная как «путь» стрелка, соединяющая точки В и Б.

Выведенный в п. 4 на экран компьютера логический код графа представлен на рис. 2.

```
// Тестовый граф
digraph {
        A [label="Точка А"]
        "Б" [label="Точка Б"]
        "В" [label="Точка В"]
        A -> "Б"
        A -> "В"
        "В" -> "Б" [label="путь"]
}
```

Рис. 2. Логический код созданного графа.

Данный логический код вполне понятен и без описания.

На следующем рис. 3 приведён граф, созданный с помощью гораздо более разветвлённого описания и применения другого стиля модуля Digraph – в частности, задан цвет областей названий точек, заполненный их стиль и максимальный размер. Отсутствует предварительное задание названий точек; названия задаются прямо во время формирования рёбер.

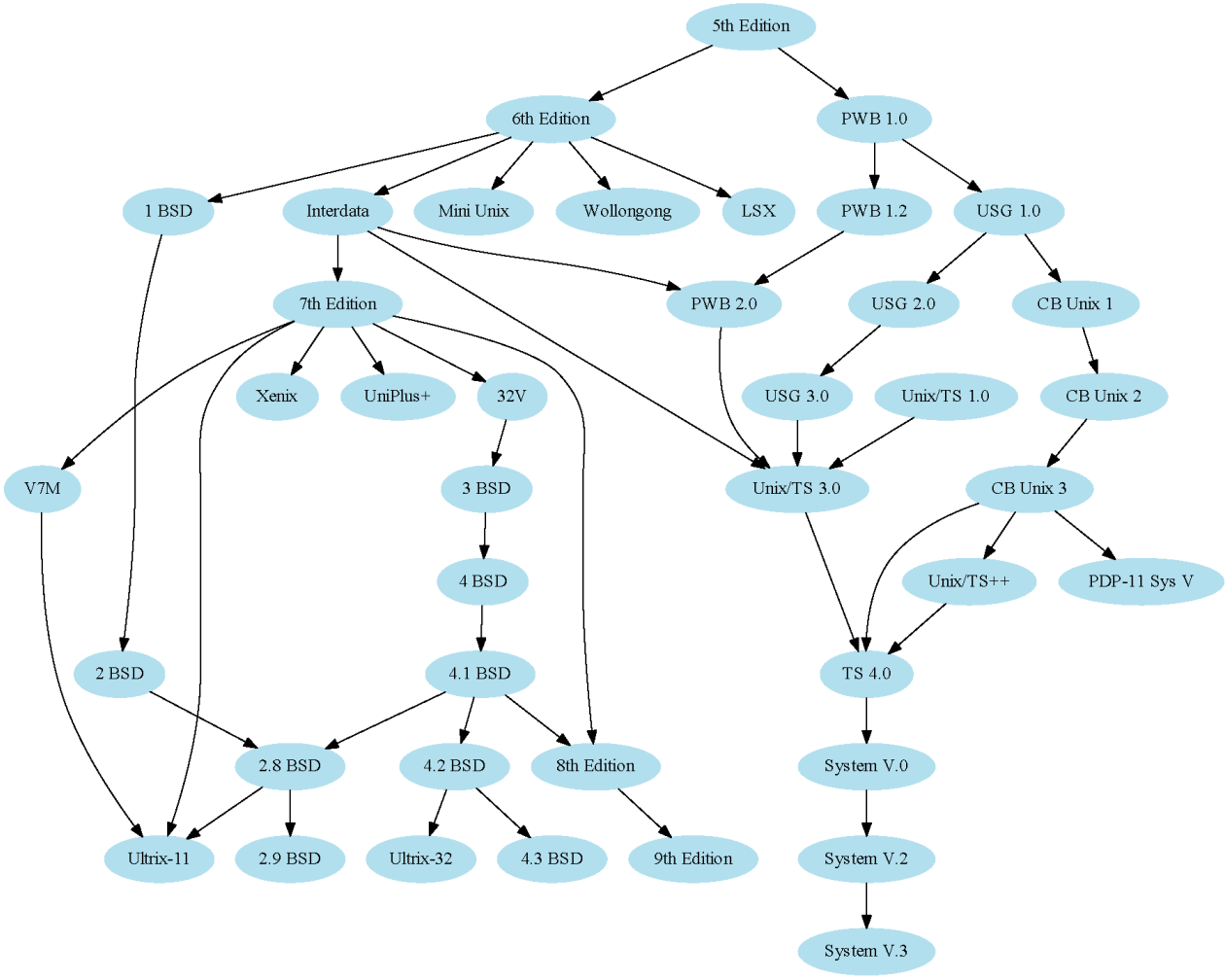

Рис. 3. Граф сложной конфигурации, созданный с помощью описанной методики.

Пример расчёта

В качестве примера рассмотрим решение простой задачи с помощью графа. Задача заключается в определении всех рёбер между точками, имеющими значения от i=0…9 + j=0…9, умноженное на случайное значение,

генерируемое функцией Random модуля Random [9] (один набор значений для начала и один – для окончания ребра). Random может изменяться от 0 до 1. Затем полученное значение (*i*+*j*)**Random* округляется до целого.

Результат выполняемой по данному алгоритму программы получается очень интересным (рис. 4).

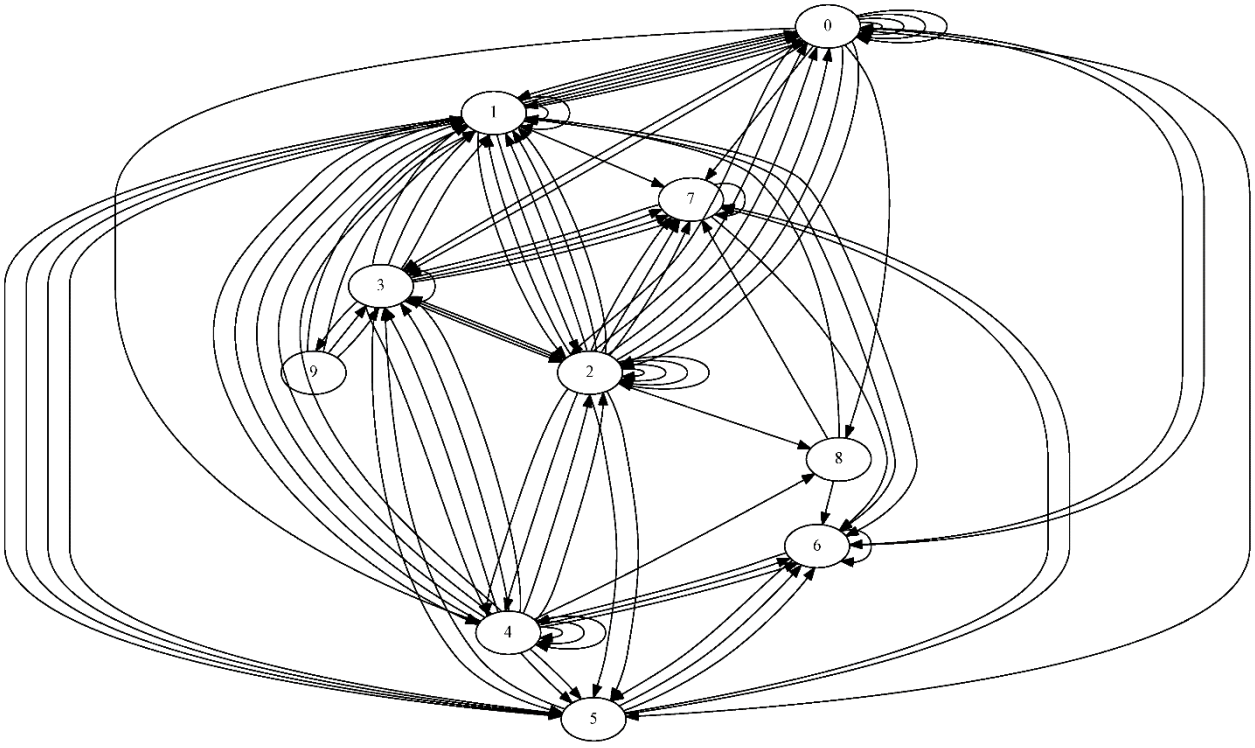

Рис. 4. Результирующий граф, построенный из решения примера.

На первый взгляд, результирующий массив содержит максимальное и минимальное значения *i*=0…9 + *j*=0…9, умноженные на Random, т.е. теоретически значения от 0 до 99, а на графе содержатся лишь узлы от 0 до 9. При этом можно подметить интересную особенность, что чем больше номер узла, тем меньше связей с другими узлами он имеет (наибольшее количество стрелок подходит к узлу 0, а к узлу 9 подходит всего одна стрелка).

Этот результат решения примера расчёта можно объяснить следующим образом: массив (*i*+*j*) при переборе (цикл в цикле) *i* и *j* от 0 до 9 содержит максимально возможное число 18 всего один раз, тогда как число 0 – 10 раз. При этом следует иметь в виду, что полученное число затем умножается ещё на

случайное число от 0 до 1. В реальности получается, что при этом практически не образуется связи между узлами графа, имеющими значения больше 9.

Здесь обнаружилась также важная особенность модуля Digraph – если попадается двухзначное число, то он складывает его цифры. Отсюда можно сделать вывод, что в данном модуле присутствует некий статистический алгоритм [10], таким образом обрабатывающий числа, не подходящие под общую канву образования графа.

Заключение

Таким образом, можно сделать вывод, что цель описываемой работы полностью достигнута: создана методика создания графов с помощью библиотеки Graphviz и её модуля Digraph, а также исследованы некоторые особенности её применения и получаемых с её помощью результатов.

Для каждого оригинального расчётного примера создана своя программа на языке Python [11], позволяющая построить граф именно необходимой для данной задачи конфигурации.

Показаны и некоторые ограничения метода, созданные разработчиками Graphviz и её модуля Digraph по-видимому, для приведения графа к единообразному виду и исключения из него данных, не подходящих по определённым критериям [12]. Так как в некоторых случаях такое искажение отображаемой графической информации неприемлемо, необходимо на отдельных стадиях выполнения программы отслеживать обрабатываемые ею данные путём вывода промежуточных результатов на экран и сравнения их с ожидаемыми.

Рассмотренная в статье методика создания графов может быть использована как для первоначального ознакомления с основами теории графов в различных сферах их применения, так и для обработки данных, полученных учёными в ходе исследований для более наглядного их представления в презентациях, блок-схемах, а также для выявления новых связей между некоторыми явлениями.

Библиографический список:

1. Яксубаев К.Д., Аксютина И.В. Определение количества причинных операторов отображающих один граф в другой граф. // Современная наука: актуальные проблемы теории и практики. Серия: Естественные и технические науки. 2019. № 5. С. 109-114.

2. Махнев А.А., Белоусов И.Н., Падучих Д.В. Обратные задачи в теории графов: графы без треугольников. // Сибирские электронные математические известия. 2021. Т. 18. № 1. С. 27-42.

3. Ильичев В.Ю. Использование парсинга для создания базы метеорологических данных и разработка на её основе нейросетевой модели прогнозирования скорости ветра. // Системный администратор. 2020. № 10 (215). С. 92-95.

4. Менщиков А.А., Гатчин Ю.А. Метод обнаружения веб-роботов на основе анализа графа пользовательского поведения. // Программные продукты и системы. 2019. № 4. С. 607-612.

5. Подколодный Н.Л., Гаврилов Д.А., Твердохлеб Н.Н., Подколодная О.А. Cytoscape - плагин для построения структурных моделей биологических сетей в виде случайных графов. // Вестник Новосибирского государственного университета. Серия: Информационные технологии. 2018. Т. 16. № 3. С. 37-50.

6. Ильичев В.Ю. Использование рекурсивных функций для создания фрактальной графики средствами языка Python. // Системный администратор. 2021. № 3 (220). С. 92-95.

7. Кофнов О.В., Шелудяков А.М. Применение пакета Graphviz при разработке структурных схем технических систем. // Труды Военно-космической академии имени А.Ф.Можайского. 2018. № 663. С. 150-154.

8. Ильичев В.Ю., Ганков М.С. Разработка методики вычисления и визуализации 3 D фракталов с использованием программы Blender. // Заметки ученого. 2022. № 1-1. С. 53-58.

9. Ильичев В.Ю., Юрик Е.А. Разработка программы для исследования термодинамического цикла Ренкина. // Научное обозрение. Технические науки.

2020. № 2. С. 32-36.

10. Nasonov D., Ilichev V., Raevsky V. The experimental study of elastic-hysteresis properties of rubber elements of sleeve-pin couplings. // В сборнике: Proceedings of the 52th International JVE Conference in St. Petersburg. 2021. С. 193-197.

11. Ильичев В.Ю. Гармонический анализ сложного сигнала колебаний газотурбинного электроагрегата. // Заметки ученого. 2021. № 12-2. С. 82-86.

12. Гребенюкова А.И. Применение параметрических критериев на примере t-критерия Стьюдента. // В книге: Радиоэлектроника, электротехника и энергетика. Тезисы докладов. 2020. С. 543.