УДК 004.42

ПРИМЕНЕНИЕ ЯЗЫКА ПРОГРАММИРОВАНИЯ PYTHON ПРИ ИЗУЧЕНИИ ТЕОРИИ ГРАФОВ

А. И. ЯКИМОВ, А. А. ЗАЙЦЕВ, Е. И. САВИЦКИЙ
Белорусско-Российский университет
Могилев, Беларусь

В соответствии с учебным планом специальности 1-53 01 02 «Автоматизированные системы обработки информации» изучается дисциплина «Теория графов», включающая теоретическую и практическую часть. Для реализации концепции непрерывной подготовки по программированию предложено использовать язык программирования Python и набор библиотек, реализующих алгоритмы теории графов: алгоритмы поиска в глубину, нахождения кратчайшего пути, поиска сильных компонент связности и т. д. Такие алгоритмы помещены в специальные библиотеки Python, для доступа к которым необходимо знать основы языка.

Методические рекомендации для предварительного изучения языка Python содержат следующие вопросы.

1. Синтаксис (например, Python не содержит операторных скобок, вместо этого блоки выделяются отступами: пробелами или табуляцией, а вход в блок из операторов осуществляется двоеточием).
2. Типы данных (базовые типы: bool, int, float, complex и str) и структуры данных (списки (lists), кортежи (tuples) и словари (dictionaries)).
3. Строки (обособляются кавычками двойными «"» или одинарными «'»).
4. Модули (math – один из наиважнейших в Python, предоставляет обширный функционал для работы с числами).
5. Операторы (If, While, For).
6. Функции (например, функция print(), которая выводит некоторое значение на консоль).
7. Классы (внутренние переменные и внутренние методы классов начинаются с двух знаков нижнего подчеркивания «__», например, «__myprivatevar»).
8. Подключение библиотек (подключить модуль можно с помощью инструкции import с указанием названия модуля).

Библиотека networkx (свободное программное обеспечение, распространяемое под BSD-лицензией) создана на языке Python и предназначена для работы с графами и другими сетевыми структурами. Основные возможности библиотеки: классы для работы с неориентированными, ориентированными и взвешенными графами; узлом может быть временная последовательность, текст, изображение, XML; сохранение / загрузка графов в/из наиболее распространённых форматов файлов хранения графов; встроенные процедуры для создания графов базовых

типов; методы для обнаружения подграфов, клик и К-дольных графов; получение таких характеристик графа, как степени вершин, высота графа, диаметр, радиус, длины путей, центр и т. д.; визуализация сети в виде 2D- и 3D-графиков.

Библиотека matplotlib – это библиотека двумерной графики для языка программирования Python, с помощью которой можно создавать высококачественные рисунки различных форматов [1].

Фрагмент Python-программы применения алгоритма Флойда с использованием библиотеки networkx и matplotlib.

```
#библиотека для использования алгоритма Флойда
import networkx as nx
#библиотека для визуализации графа
import matplotlib.pyplot as plt
#матрица смежности вершин
edges = [(1, 2, {'weight': 4}), (1, 3, {'weight': 2}), (2, 3, {'weight': 1}), (2, 4, {'weight': 5}), (3, 4, {'weight': 8}), (3, 5, {'weight': 10}), (4, 5, {'weight': 2}), (4, 6, {'weight': 8}), (5, 6, {'weight': 5})]
#нагрузки на ребрах
edge_labels = {(1, 2): 4, (1, 3): 2, (2, 3): 1, (2, 4): 5, (3, 4): 8, (3, 5): 10, (4, 5): 2, (4, 6): 8, (5, 6): 5}
G = nx.Graph()
#добавление вершин в граф
for i in range(1, 7):
    G.add_node(i)
#добавление ребер с нагрузками
G.add_edges_from(edges)
#позиционирование узлов без пересечений ребер
pos = nx.planar_layout(G)
#построение графа G (with_labels = True - видимость меток узлов)
nx.draw(G, pos, with_labels=True, node_size = 1300, node_color = 'yellow', font_size = 18)
#построение связей с указанием веса ребер
nx.draw_networkx_edge_labels(G, pos, edge_labels=edge_labels, font_size=18)
#визуализация графа (рисунок 1)
plt.show()
#применение алгоритма Флойда
fw = nx.floyd_warshall(G, weight='weight')
#вывод кратчайших путей
results = {a: dict(b,) for a, b in fw.items()}
print(results)
```

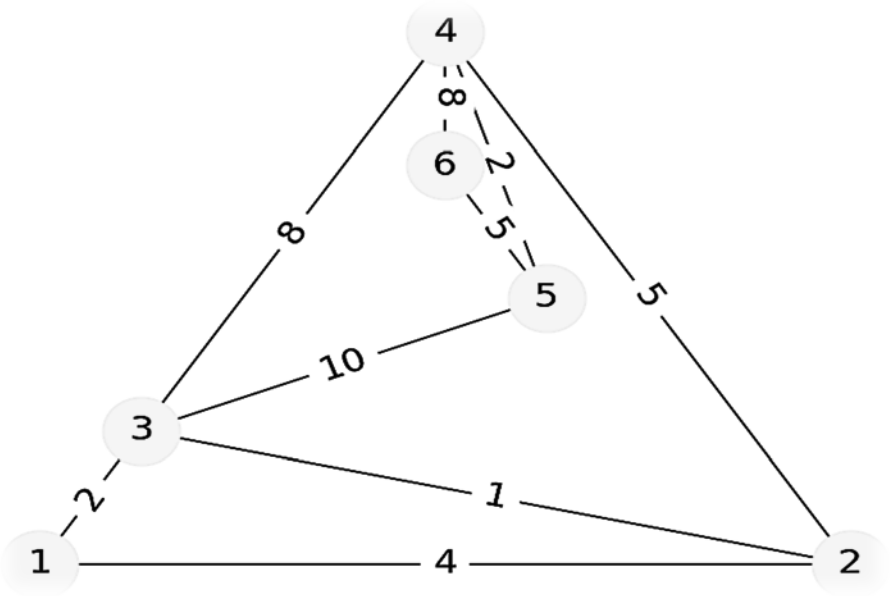

Рис. 1. Визуализация графа

В последнее время при визуализации данных применяется библиотека graphviz, которая используется для создания деревьев решений и эффектов блок-схем. Движок диаграмм использует язык описания графов DOT, который представляет собой текстовое описание структуры графа: вершины, их связи, группы и атрибуты для их визуального оформления [2].

Для углубленного развития навыков по языку программирования Python предложено прохождение дополнительно обучающего курса на образовательной интернет-платформе Stepik с получением сертификата [3].

На последующих практических занятиях предлагается реализовать операции удаления вершин, удаления ребра, дополнения графа, объединения графов, композиции графов и основных алгоритмов теории графов в виде отдельных программных модулей.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. **Шабанов, П. А.** Научная графика в Python [Электронный ресурс] / П. А. Шабанов // github.com. – Режим доступа: https://github.com/whitehorn/Scientific_graphics_in_python/ blob/ master/P1%20Chapter%201%20Pyplot.ipynb. – Дата доступа: 15.01.2023.

2. Использование библиотеки graphviz в Python [Электронный ресурс] // russianblogs.com. – Режим доступа: https://russianblogs.com/article/58061468970/. – Дата доступа: 15.01.2023.

3. Программирование на Python [Электронный ресурс] // stepik.org. – Режим доступа: https://stepik.org/course/67/syllabus. – Дата доступа: 15.01.2023.