УДК 004.42

ИЗУЧЕНИЕ ТЕОРИИ ГРАФОВ С ПРИМЕНЕНИЕМ ЯЗЫКА ПРОГРАММИРОВАНИЯ PYTHON

Якимов А.И., Харламов И.В., Шунькин В.А.
Межгосударственное образовательное учреждение высшего образования
«Белорусско-Российский университет», Беларусь, Могилев

Аннотация. *Предложено использовать язык программирования Python и набор библиотек, реализующих алгоритмы теории графов. Разработаны тексты программ для реализации алгоритма поиска в ширину и алгоритма поиска в глубину на языке программирования Python для изучения теории графов.*

Ключевые слова: *теория графов, алгоритм поиска в ширину, алгоритм поиска в глубину, Python.*

Annotation. *It is proposed to use the Python programming language and a set of libraries implementing graph theory algorithms. Programs for implementing the breadth-first search and depth-first search algorithms in the Python programming language for studying graph theory have been developed.*

Key words: *graph theory, breadth-first search algorithm, depth-first search algorithm, Python.*

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:

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

Фрагмент Python-программы для реализации алгоритма поиска в ширину.

```
def bfs(graph, source):
    visited = set()  # отслеживание уже посещенных узлов
    result = list()  # результат обхода BFS
    queue = list()  # очередь

    # помещаем корневой узел в очередь и отмечаем его как посещенный
    queue.append(source)
    visited.add(source)

    # пока очередь не опустеет
    while queue:
        # извлекаем передний узел очереди и добавляем его в result
        current_node = queue.pop(0)
        result.append(current_node)

        # проверяем все соседние узлы текущего узла
        for neighbour_node in graph[current_node]:
            # если соседние узлы еще не посещены,
            # помещаем их в очередь и помечаем как посещенные
            if neighbour_node not in visited:
                visited.add(neighbour_node)
                queue.append(neighbour_node)

    return result

# граф
graph = {
    5: [3],
    1: [2, 3, 4],
    2: [1],
    3: [1, 4, 5],
    4: [3, 1]}

# вывод результата обхода графа
result = bfs(graph, 1)
print(f"BFS: {result}")
```

Фрагмент Python-программы для реализации алгоритма поиска в глубину.

```
def dfs(v):
    if v in result:  # если вершина уже посещена, выходим
        return
    result.append(v)  # посетили вершину v
    for i in graph[v]:  # все смежные с v вершины
        if not i in result:
            dfs(i)

# граф
graph = {
    1: [2, 8],
    2: [1, 3, 8],
    3: [2, 4, 8],
```

```
    4: [3, 7, 9],
    5: [6, 7],
    6: [5],
    7: [4, 5, 8],
    8: [1, 2, 3, 7],
    9: [4],}

result = list()

start = 1
dfs(start)  # start – начальная вершина обхода
print(f"DFS: {result}")
```

Для углубленного развития навыков по языку программирования Python предложено прохождение дополнительно обучающего курса на образовательной интернет-платформе Stepik
с получением сертификата [1].

На последующих практических занятиях планируется реализовать элементарные алгоритмы теории графов в виде отдельных программных модулей.

Список литературы:

1. Программирование на Python // stepik.org URL: https://stepik.org/course/67/syllabus (дата обращения: 14.01.2023).