УДК 378

В.А.Щербакова

ПРИМЕНЕНИЕ JUPYTER NOTEBOOK ДЛЯ ИЗУЧЕНИЯ АЛГОРИТМОВ ОБХОДА В ШИРИНУ И ОБХОДА В ГЛУБИНУ НА НЕОРИЕНТИРОВАННЫХ ГРАФАХ

Аннотация

В статье приводится описание использования Jupyter Notebook при изучении методов обхода в ширину и обхода в глубину в неориентированных графах в учебной дисциплине «Дискретная математика» у студентов направления «Прикладная математика» УрФУ.

1. Введение

Данная статья посвящена проблеме получения студентами практических навыков в использовании методов и алгоритмов решения задач на графах. В монографии И. А. Липского и М. С. Чвановой [1] взаимосвязь теоретической и практической деятельности названа основным признаком того, что полученное знание является методологическим.

Обычным способом тренировки студентов в использовании алгоритмов, рассказанных лектором, является решение задач «вручную». Но для применения в хозяйственной деятельности нужны программные продукты, в которых выполняется комбинирование различных алгоритмов. Следовательно, получение студентами навыков реализации алгоритмов в виде компьютерной программы является важной методологической задачей для преподавателя.

При использовании Jupyter Notebook преподаватель получает возможность тренировать студентов в создании компьютерных программ на языке высокого уровня Python. При этом у преподавателя есть возможность давать подсказки с целью уменьшения трудностей, связанных с техническими особенностями процесса программирования.

2. Краткое описание Jupyter Notebook

Система Jupyter Notebook является средой для написания и выполнения программ на языке высокого уровня Python.

Основным объектом в системе является файл с расширением .ipynb, называемый «блокнот». Блокнот может включать в себя обычный текст, рисунки, ссылки на внешние ресурсы, и тексты программ на языке Python. Такой блокнот, как и любой другой файл, можно копировать, пересылать по почте, выкладывать в «облако» и т.д. Идея объединения в одном документе текста, рисунков, формул и выполняемых программ, вычисляющих значения формул, также используется в файлах системы Mathcad, получившей широкое применение в инженерных дисциплинах.

Для выполнения программ, записанных в блокноте, используется веб-приложение и сервер Jupyter. Также имеется возможность установить на свой компьютер приложение для локального запуска Jupyter Notebook. Подробное руководство для работы с Jupyter Notebook содержится в [2].

3. Методы обхода в ширину и в глубину в неориентированном графе

Алгоритмы поиска в ширину и в глубину в неориентированных графах были придуманы достаточно давно, их можно назвать «классическими» для теории графов. В различных модификациях эти алгоритмы применяются в большом количестве задач на графах. Один из вариантов изложения алгоритмов можно найти в [3].

Поиск в ширину в неориентированном графе продолжает идею поиска в ширину в дереве, рисунок которого имеет стандартное изображение. Начинается поиск в дереве с корня, далее выполняется просмотр всех сыновей корня, далее просматриваются их сыновья и т.д. Результат поиска – последовательность вершин дерева, упорядоченная в соответствии с порядком просмотра.

Для произвольного неориентированного графа в описании алгоритма поиска в ширину используется объект «очередь». Очередью называется линейный массив $[a_1,a_2,\ldots,a_n]$, в который можно добавлять элементы в «хвост» (правый конец) и извлекать элемент из начала (левый конец). Обычно очередь описывают фразой: «Первым пришел – первым ушел».

Алгоритм поиска в ширину (Breadth-first search).

Вход алгоритма – связный граф (V, E).

Шаг 0. Создать пустую очередь L.

Шаг 1. Выбрать любую вершину v_1.

Занести в L вершину v_1.

Шаг 2. Извлечь из очереди L вершину u и пометить ее «развернутая».

Занести в L все неразвернутые вершины, смежные вершине u, не находящиеся в очереди.

Шаг 3.... Повторить шаг 2, если L не пусто.

В противном случае – остановиться.

Выход алгоритма – упорядоченная последовательность развернутых вершин, образующая дерево.

Поиск в глубину в неориентированном графе продолжает идею поиска в глубину в дереве, рисунок которого имеет стандартное изображение. Начинается поиск в дереве с корня, далее выполняется просмотр самого левого сына корня, далее просматриваются его левый сын и т.д. Если у просматриваемой вершины нет сына, выполняется возврат на один уровень выше и просматривается следующий сын и т.д. Результат поиска – последовательность вершин дерева, упорядоченная в соответствии с порядком просмотра.

Алгоритм поиска в глубину (depth-first search).

Вход алгоритма – связный граф (V, E).

Шаг 0. Покрасить все вершины в белый цвет.

Шаг 1. Выбрать любую вершину v_1. Вызвать подпрограмму DFS(v_1).

Описание подпрограммы DFS(v).

1. Перекрасить вершину v в серый цвет.
2. Для каждой белой вершины w, смежной v, вызвать DFS(w).
3. Перекрасить вершину v в черный цвет.

Выход алгоритма – упорядоченная последовательность черных вершин, образующая дерево.

4. Полный текст блокнота с заданием на тему «Обход в ширину и обход в глубину»

Алгоритмы на графах: поиск в ширину и поиск в глубину

Этот набор задач предназначен для самостоятельного решения.

Во всех задачах используется граф, заданный рисунком 1.

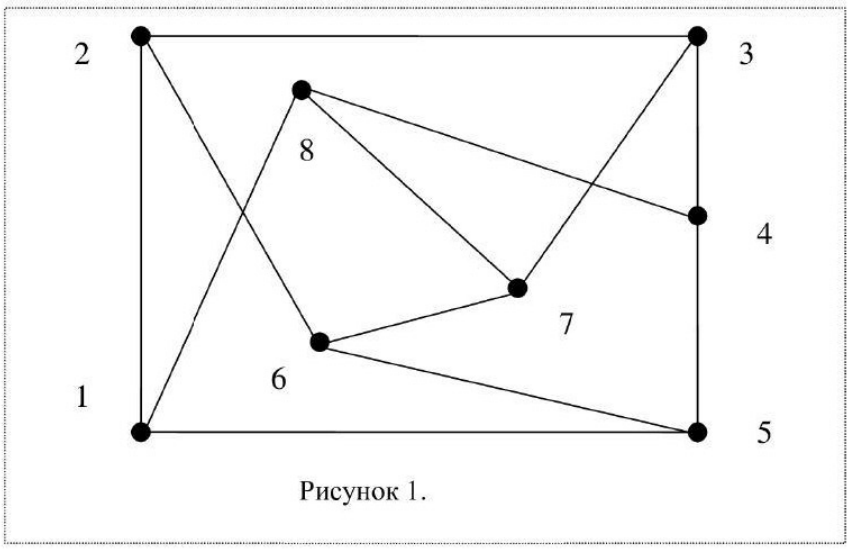

Рисунок 1.

Входной информацией о графе будет матрица смежности. Информацией о дереве будет разбиение вершин на уровни и список сыновей для вершин дерева.

1. Создание матрицы смежности. Допишите функцию make_matrix, в которой сначала вручную заполнить диагональ и верхний треугольник матрицы смежности графа на рисунке 1. Затем используя симметричность относительно главной диагонали заполнить нижний треугольник. Вывести на печать получившуюся матрицу смежности.

```
def make_matrix():
    # здесь мог бы быть ваш код
    matrix =
    return matrix
```

Проверьте работу функции make_matrix:

```
import
# вывод на печать
```

2. Поиск в ширину. Допишите функцию breadth_f_s, которая по номеру вершины N, равному номеру варианта студента, по матрице смежности графа graph, возвращает последовательность вершин, пройденных алгоритмом поиска в ширину и информацию об этом дереве (разбиение на уровни и список сыновей).

```
def breadth_f_s(N, graph):
    # здесь мог бы быть ваш код
    return
```

Проверьте, работу функции breadth_f_s:

```
import
graph = make_matrix
N =
breadth_f_s(N, graph):
# вывод на печать
```

3. Поиск в глубину. Допишите функцию depth_f_s, которая по номеру вершины N , равному номеру варианта студента, по матрице смежности графа graph, возвращает последовательность вершин, пройденных алгоритмом поиска в глубину и информацию об этом дереве (разбиение на уровни и список сыновей)..

```
def depth_f_s(N, graph):
    # здесь мог бы быть ваш код
    return
```

Проверьте, работу функции depth_f_s:

```
import
graph = make_matrix
N =
depth_f_s(N, graph):
# вывод на печать
```

4. Рисунок дерева. Допишите функцию make_picture, которая по входной информации о дереве выполняет рисунок дерева с названиям вершин.

```
def make_picture(graph):
    # здесь мог бы быть ваш код
    return
```

Выполните два рисунка деревьев, полученных поиском в ширину и поиском в глубину.

```
import
# вывод на печать
```

Список литературы

1. Чванова М. С., Липский И. А. Информатизация системы непрерывной подготовки специалистов: методология, теория, практика. Монография. - Тамбов-Москва, Изд. ТГУ, 2000. - 518 с.
2. Руководство для начинающих. URL: https://pythonru.com/baza-znanij/jupyter-notebook-dlja-nachinajushhih.
3. Липский В. Комбинаторика для программистов. - Москва: «Мир», 1988.- 213 с.