Python. Список задач 2.

1 Numpy введение

При решении научных задач с помощью численных методов зачастую приходится иметь дело с большими массивами данных в виде массивов и матриц. Стандартные средства Python (например, использование списков для хранения данных и прямой перебор элементов с помощью циклов) плохо подходят для таких задач, так как будут работать крайне медленно, в отличие от программ на компилируемых языках (Fortran, C). Пакет NumPy (http://www.numpy.org/) — это основной пакет, используемый в научных вычислениях в Python. NumPy предназначен для эффективной работы с многомерными массивами, в том числе для эффективного применения математических операций к массивам и матрицам и их модификации без прямого перечисления элементов массивов. Помимо этого NumPy обладает и другими возможностями — линейная алгебра, удобный ввод/вывод в файлы, моделирование случайных величин, быстрое преобразование Фурье, объединение программ на Fortran и C и другое (подробнее см. оф. сайт), поэтому важно изучить возможности данной библиотеки как можно лучше. Прежде чем пытаться самостоятельно писать алгоритм, требующий перебора элементов массива, проверьте, нельзя ли это сделать средствами NumPy!

1. Внимательно изучите уроки 1-4 по NumPy https://pythonworld.ru/numpy и проделайте все примеры, либо официальный туториал https://docs.scipy.org/doc/numpy/user/basics.html).
2. Изучите официальное руководство пользователя http://docs.scipy.org/doc/numpy/user/.
3. Внимательно изучите Reference Guide (полное описание всех функций) http://docs.scipy.org/doc/numpy/reference/, именно в нем перечислены все возможности NumPy, этой ссылкой придется регулярно пользоваться.

2 Matplotlib

Matplotlib (http://matplotlib.org/index.html) — это библиотека для создания графиков в Python. Matplotlib позволяет строить графики любой сложности типографского качества, которые можно вставить в статью. Подробнее о возможностях см. http://matplotlib.org/users/intro.html. Matplotlib — большой и сложный инструмент, но на практике в научных вычислениях часто пользуются частью библиотеки matplotlib.pyplot, которая имитирует построение графиков в стиле MATLAB. В качестве достаточно полного введения можно прочитать http://matplotlib.org/users/index.html.

На практике бывает очень трудно помнить все возможности `Matplotlib`, поэтому часто используют галереи готовых примеров, а потом модифицируют их под свои нужды. Примеры таких подборок можно найти по следующим ссылкам:

- `http://matplotlib.org/gallery.html`
- `http://matplotlib.org/examples/index.html`

Уроки на русском языке `https://github.com/whitehorn/Scientific_graphics_in_python`

Изучите введение в Matplotlib и проделайте все примеры (примеры нужно показать) `http://matplotlib.org/users/pyplot_tutorial.html` и `http://www.loria.fr/~rougier/teaching/matplotlib/`.

3 Стиль

Стиль написания кода (внешний вид кода, структура кода, последовательность действий, порядок хранения файлов и модулей и прочее) — это важный элемент написания программ. Хороший стиль сэкономит вам много времени и сил при отладке кода. Внимательно изучите рекомендации по стилю написания кода, их специально составили очень опытные программисты, чтобы облегчить всем жизнь:

1. Прочитайте перевод статьи “Style Guide for Python Code”.
2. “Советы Google по кодированию на языке Python”:
   `http://habrahabr.ru/post/179271/`
   `http://habrahabr.ru/post/180509/`
3. Также можно почитать `http://docs.python-guide.org/en/latest/`

4 Перколяция

Перколяция (percolation, протекание) — модель протекания жидкости через пористую среду или проводимости случайной среды. Рассмотрим матрицу $n \times n$, заполненную 0 и 1 случайным образом, причем элемент матрицы равен 1 с вероятностью p и 0 с вероятность $(1 - p)$. Будем считать, что две единицы стоят рядом, если они находятся в соседних клетках в одном ряду или столбце (единицы, расположенные по диагонали не считаются стоящими рядом). Так единицы могут образовывать группы в матрице, такие группы называют *кластерами*. Если в матрице существует кластер, соединяющий первый и последний столбцы матрицы, то будем называть такую матрицу *проводящей*. Если сгенерировать матрицу с определенным значением p, она может быть как проводящей, так и непроводящей, но очевидно, что чем больше p, тем выше вероятность того, что матрица будет проводящей. При определенном значении вероятности p, практически все матрицы станут проводящими. Можно показать, что при $n \to \infty$ значение вероятности p_c станет вполне определенным, это значение называют *порогом перколяции*.

1. Напишите функцию, которая генерирует матрицу $n \times n$ из 0 и 1 с заданным значением вероятности $0 \leq p \leq 1$ (входные параметры в функцию: n, p, функция должна возвращать сгенерированную матрицу). Для хранения матрицы

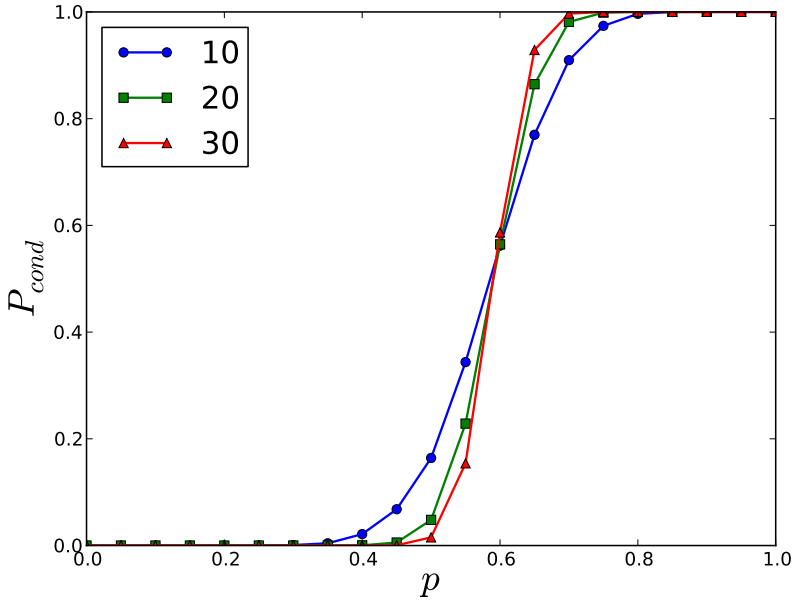

Рис. 1: Вероятность проводимости матрицы для различных значений размера матрицы в *двухмерном* случае

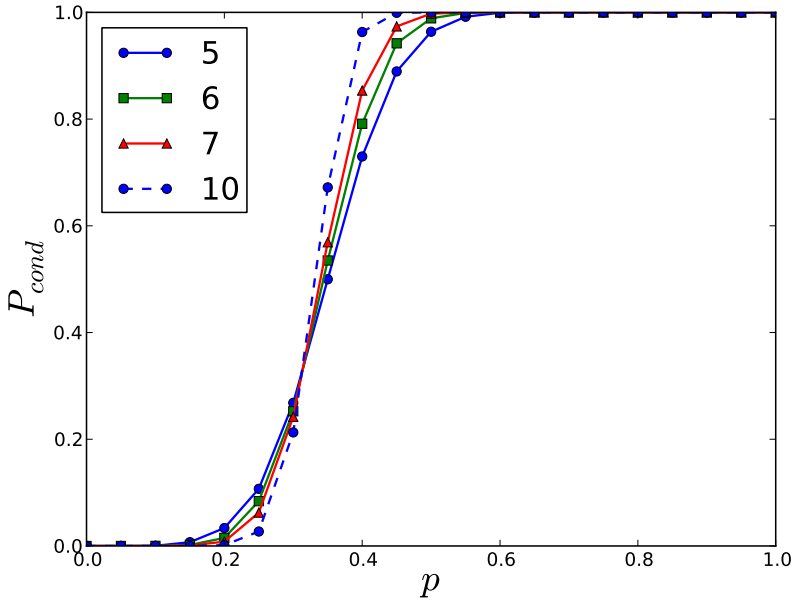

Рис. 2: Вероятность проводимости матрицы для различных значений размера матрицы в *трехмерном* случае

можно использовать списки (список из списков для двухмерной матрицы) или массивы из библиотеки NumPy. Для генерации случайных чисел используйте модуль `random` из стандартной библиотеки или модуль `random` из NumPy.

Напишите функцию, которая выводит матрицу на экран.

2. Напишите функцию[1], которая проверяет, является ли данная матрица проводящей. Обязательный входной параметр функции — матрица M, также можно задавать размер входной матрицы или какие-то другие необходимые данные. Функция должна возвращать `True`, если матрица проводящая, или `False` в обратном случае.

3. Напишите функцию, которая вычисляет вероятность проводимости P_{cond} матрицы размера $n \times n$ в зависимости от вероятности p в заданном числе экспериментов k. Функция должна k раз сгенерировать матрицу и проверить её, тогда вероятность проводимости при данных n и p – это отношение числа проводящих матриц к полному числу экспериментов k. Входные параметры: n, p, k, функция должна возвращать число – вероятность проводимости P_{cond}.

   Для того, чтобы результат был достоверным, число экспериментов должно быть достаточно большим, например $k = 10000$. Проверьте проводимость матрицы для $n = 10, 15, 20$ при различных p.

4. Постройте зависимость вероятности проводимости матрицы размера n =10, 15, 20 от p с помощью Matplotlib. При этом сделайте 2 отдельные программы: первая делает рачёт проводимости для разных значений n и сохраняет данные в файл, вторая строит графики по сохранённым данным. Найдите пороговое значение проводимости.

5. * Обобщите алгоритм на трехмерный случай. Постройте зависимость вероятности проводимости от p для n =5, 6, 7, 10.

[1] Разумеется, функций, участвующих в вычислении проводимости, может быть несколько.