Python. Домашнее задание по практике 1

Богдан Бугаев, Антон Крыщенко*

1. Филиал Хаскелла с привкусом Лиспа

Написать связные списки и функции для работы с ними в хаскеллолисповском стиле. Для реализации необходимо использовать контейнер `namedtuple` из библиотеки `collections`.

Нельзя использовать циклы. Нельзя менять один раз созданную переменную.

Заготовка для задачи (с тестами) прилагается.

2. Быстрое сравнение подматриц

У нас есть большая целочисленная матрица $N \times M$. Нам нужно научиться быстро (за $\mathcal{O}(1)$) сравнивать любые две её подматрицы. Для этого можно использовать полиномиальные хеши (вероятностью ошибиться пренебрежем). Про них можно прочитать, например, здесь: `http://habrahabr.ru/post/142589/`. Про то, как их разумнее реализовывать, можно прочитать здесь: `http://codeforces.com/blog/entry/17507/`.

Входные данные

Входные данные нужно считывать из `stdin`. На первой строке через пробел идет пара чисел N и M: количество строк и столбцов в матрице. Затем идет N строк по M целых чисел a_{ij} через пробел в каждой — сама матрица. На следующей строке идет число Q — количество запросов. После идет Q строк по восемь чисел i_1, j_1, h_1, w_1, i_2, j_2, h_2, w_2 в каждой; эти числа обозначают координаты левого верхнего угла и высоту с шириной для первой и второй подматриц соответственно.

*Идеи задач и сами задачи могут быть позаимствованы откуда угодно (а могут и не быть), источники не указываются в учебных целях.

Ограничения: $1 \leqslant N, M \leqslant 10^3$, $1 \leqslant Q \leqslant 10^5$, $0 \leqslant a_{ij} \leqslant 10^6$. Нумерация строк и столбцов начинается с нуля.

Гарантируется, что входные данные корректны.

Выходные данные

В `stdout` нужно вывести Q строк — ответы на каждый запрос. Если подматрицы равны, нужно выводить «1», иначе «0» (без кавычек).

Пример входных и выходных данных

Input	Output
2 2 10 20 10 20 1 0 0 1 2 1 0 1 2	1

Прочее

Заготовка для задачи есть в приложении. Использовать сторонние библиотеки нельзя.

3. Сталкер 3.0

Не так давно мы писали скрипт, умеющий отслеживать, когда человек появляется онлайн в VK и когда исчезает. Теперь мы хотим научиться мониторить список друзей: кого человек добавляет, кого удаляет. Понятно, что с учетом существования скрытых друзей эту задачу полностью решить законным способом нельзя, зато вполне себе можно отслеживать всех открытых.

Итак, нам нужно:

1. Получать список идентификаторов друзей с помощью VK API. Для этого мы предлагаем вам использовать библиотеки `requests` и `json`.
2. Далее из этого списка сформировать множество (`set`).
3. Затем, в зависимости от параметров командной строки:
   - сохранить это множество на диске (с использованием `cPickle`);

- сравнить с уже имеющимся на диске множеством и вывести разницу: кто пришел, кто ушел.

Может показаться, что советы по использованию библиотек носят рекомендательный характер, однако это не так — мы прямо-таки настаиваем, чтобы вы использовали всё перечисленное.

При запуске скрипт должен принимать параметром идентификатор пользователя и ключ, указывающий, что нужно сделать: `-s` или `--save` если нужно сохранить информацию на диск, и `-d` или `--diff`, если нужно показать разницу. Ключ и id могут идти в любом порядке. Например:

```
./task3.py -s id1
./task3.py id1 --diff
```

Для обработки параметров рекомендуем использовать `getopt`.

Если параметры указаны некорректно, нужно вывести справку и завершиться с ненулевым кодом возврата. Если по каким-то причинам не удалось выполнить работу (например, задан неверный id) нужно вывести сообщение об ошибке и также завершиться с ненулевым кодом возврата.

Скриптом должно быть более-менее приятно пользоваться, так что выводить информацию нужно в пристойном виде: с именем того, за кем следим, с именами ушедших и пришедших друзей, а не одни идентификаторы.

4. Список с пропусками

Реализовать список с пропусками. Подробнее про него прочитать можно тут: `http://neerc.ifmo.ru/wiki/index.php?title=%D0%A1%D0%BF%D0%B8%D1%81%D0%BE%D0%BA_%D1%81_%D0%BF%D1%80%D0%BE%D0%BF%D1%83%D1%81%D0%BA%D0%B0%D0%BC%D0%B8`

Ваш список должен поддерживать итераторы. Заготовка прилагается.

5. Частотные словари

У нас есть книга. Как известно, книга состоит из слов (помимо прочего). Рассмотрим каждое в отдельности, в отрыве от контекста. Может оказаться так, что существует несколько различных слов, которые

в некоторой морфологической форме принимают вид рассматриваемого слова.

Давайте построим по нашей книге частотные словари (`https://ru.wikipedia.org/wiki/%D0%A7%D0%B0%D1%81%D1%82%D0%BE%D1%82%D0%BD%D1%8B%D0%B9_%D1%81%D0%BB%D0%BE%D0%B2%D0%B0%D1%80%D1%8C`) для каждой части речи. Т. е. разобьем книгу на слова, проанализируем каждое с помощью морфологического анализатора, посмотрим на его возможные начальные формы и для каждой увеличим счетчик вхождений (счетчики заведем отдельные для каждой части речи).

Технические подробности

Путь к книге в текстовом формате передается скрипту первым параметром.

Для простоты словом будем считать непрерывную последовательность букв; значит, все символы помимо букв являются разделителями слов. Для разбиения строки на слова рекомендуем использовать регулярные выражения (модуль `re`).

Для морфологического анализа необходимо использовать библиотеку `pymorphy2` (`https://pymorphy2.readthedocs.org/`). В этой библиотеке названия частей речи чуть не совпадают с привычными школьными; мы будем пользоваться библиотечными. В результате анализа слова получается список вариантов разбора; у каждого есть параметр `score`. Мы будем брать в расчет только те варианты, у которых `score` $\geqslant 0.1$. Если для какого-то слова не удалось определить часть речи, просто не считаем его.

Для счетчиков настоятельно рекомендуем использовать контейнер `Counter` из модуля `collections`.

После того, как все посчитано, нужно для каждой части речи выбрать 10 самых часто встречающихся слов и вывести в виде таблички (в первом столбике слова, во втором значения счетчика) в CSV-файлик. Из примера будет понятен формат, в котором это нужно сделать. Для работы с CSV стоит использовать модуль `csv`.

Пример работы скрипта

```
$ ls
decameron.txt  task5.py
$ ./task5.py decameron.txt
$ ls
ADJF.csv  CONJ.csv       INTJ.csv  PREP.csv
```

```
ADJS.csv  decameron.txt  NOUN.csv  PRTF.csv
VERB.csv  ADVB.csv       GRND.csv  NPRO.csv
PRCL.csv  PRTS.csv       COMP.csv  INFN.csv
NUMR.csv  PRED.csv       task5.py
$ cat NOUN.csv
человек,1340
любовь,1170
время,1108
дама,969
раз,932
имя,931
жена,885
дом,827
муж,780
дело,776
```

Заготовка и интересная книга прилагаются.