Лабораторная работа №5. Работа с файлами. Алгоритм Дейкстры.

Работа с файлами.

Прежде, чем работать с файлом, его надо открыть. С этим замечательно справится встроенная функция open:

```
f = open('text.txt', 'r')
```

У функции open много параметров, они указаны в статье "Встроенные функции", нам пока важны 3 аргумента: первый, это имя файла. Путь к файлу может быть относительным или абсолютным. Второй аргумент, это режим, в котором мы будем открывать файл.

Режим	Обозначение
'r'	открытие на чтение (является значением по умолчанию).
'w'	открытие на запись, содержимое файла удаляется, если файла не существует, создается новый.
'x'	открытие на запись, если файла не существует, иначе исключение.
'a'	открытие на дозапись, информация добавляется в конец файла.
'b'	открытие в двоичном режиме.
't'	открытие в текстовом режиме (является значением по умолчанию).
'+'	открытие на чтение и запись

Режимы могут быть объединены, то есть, к примеру, 'rb' - чтение в двоичном режиме. По умолчанию режим равен 'rt'.

И последний аргумент, encoding, нужен только в текстовом режиме чтения файла. Этот аргумент задает кодировку.

Чтение из файла

Открыли мы файл, а теперь мы хотим прочитать из него информацию. Для этого есть несколько способов, но большого интереса заслуживают лишь два из них.

Первый - метод read, читающий весь файл целиком, если был вызван без аргументов, и n символов, если был вызван с аргументом (целым числом n).

```
>>>
```

```
>>> f = open('text.txt')

>>> f.read(1)

'H'

>>> f.read()

'ello world!\nThe end.\n\n'
```

Ещё один способ сделать это - прочитать файл построчно, воспользовавшись циклом for:

```
>>>
```

```
>>> f = open('text.txt')

>>> for line in f:

...     line

...

'Hello world!\n'

'\n'

'The end.\n'

'\n'
```

Запись в файл

Теперь рассмотрим запись в файл. Попробуем записать в файл вот такой вот список:

```
>>>
```

```
>>> l = [str(i)+str(i-1) for i in range(20)]

>>> l

['0-1', '10', '21', '32', '43', '54', '65', '76', '87', '98',
'109', '1110', '1211', '1312', '1413', '1514', '1615', '1716',
'1817', '1918']
```

Откроем файл на запись:

```
>>>
```

```
>>> f = open('text.txt', 'w')
```

Запись в файл осуществляется с помощью метода write:

```
>>>
```

```
>>> for index in l:

...     f.write(index + '\n')

...

4

3

3

3

3
```

Для тех, кто не понял, что это за цифры, поясню: метод write возвращает число записанных символов.

После окончания работы с файлом его **обязательно нужно закрыть** с помощью метода close:

```
>>>
```

```
>>> f.close()
```

Теперь попробуем воссоздать этот список из получившегося файла. Откроем файл на чтение (надеюсь, вы поняли, как это сделать?), и прочитаем строки.

```
>>>
```

```
>>> f = open('text.txt', 'r')

>>> l = [line.strip() for line in f]

>>> l

['0-1', '10', '21', '32', '43', '54', '65', '76', '87', '98',
'109', '1110', '1211', '1312', '1413', '1514', '1615', '1716',
'1817', '1918']

>>> f.close()
```

Реализация алгоритма Дейкстры

1 вариант

```
inf = float('Inf')

def dijkstra(G, s):
    n = len(G)

    Q = [(0, s)]

    d = [inf for i in range(n)]
    d[s] = 0
    while len(Q)!=0:
        (cost, u) = Q.pop(0)

        for v in range(n):
            if d[v] > d[u] + G[u][v]:
                d[v] = d[u] + G[u][v]
                Q.append((d[v], v))
    return d

G = [\
        [inf,  7.0,  9.0,  inf, 14.0, inf],\
        [7.0,  inf,  10.0, 15.0, inf, inf],\
        [9.0,  10.0, inf,  11.0, 2.0, inf],\
        [inf,  15.0, 11.0, inf, 6.0, inf],\
        [14.0, inf,  2.0,  6.0, inf, 9.0],\
        [inf,  inf,  inf,  inf, 9.0, inf]]

d = dijkstra(G, 3)
print(d)
```

2 вариант

```
import heapq as hq
inf = float('Inf')

def dijkstra(G, s):
    n = len(G)

    Q = [(0, s)]

    d = [inf for i in range(n)]
    d[s]=0


    while len(Q)!=0:
        (cost, u) = hq.heappop(Q)

        for v in range(n):
            if d[v] > d[u] + G[u][v]:
                d[v] = d[u] + G[u][v]
                hq.heappush(Q, (d[v], v))

    return d



G = [\
        [0.0,  7.0,  9.0,  inf, 14.0, inf],\
        [7.0,  0.0,  10.0, 15.0, inf, inf],\
        [9.0,  10.0, 0.0,  11.0, 2.0, inf],\
        [inf,  15.0, 11.0, 0.0, 6.0, inf],\
        [14.0, inf,  2.0,  6.0, inf, 9.0],\
        [inf,  inf,  inf,  inf, 9.0, 0.0]]

d = dijkstra(G, 5)
print(d)
```

3 вариант с восстановлением пути

```
inf = float('Inf')
start = 0

def dijkstra(G, s):
    n = len(G)

    Q = [(0, s)]

    d = [inf for i in range(n)]
    d[s] = 0
    while len(Q)!= 0:
        (cost, u) = Q.pop(0)

        for v in range(n):
            if d[v] > d[u] + G[u][v]:
                d[v] = d[u] + G[u][v]
                Q.append((d[v], v))
    return d

def getPath(f):
    global d
    n = len(G)

    path = [f]
    while f != start:
        for v in range(n):
            if d[v] == d[f] - G[f][v]:
                path.append(v)
                f = v
    return path[::-1]

G = [\
        [inf,  7.0,  9.0,  inf, 14.0, inf],\
        [7.0,  inf,  10.0, 15.0, inf, inf],\
        [9.0,  10.0, inf,  11.0, 2.0, inf],\
        [inf,  15.0, 11.0, inf, 6.0, inf],\
        [14.0, inf,  2.0,  6.0, inf, 9.0],\
        [inf,  inf,  inf,  inf, 9.0, inf]]

d = dijkstra(G, start)
print(d)
print(getPath(3))
```

Задание.

Задание 1. Создайте текстовый файл in.txt, в который поместите текст (объёмом примерно 0,5 страницы). Затем ваша программа должна считать данные из файла, выполнить операции в соответствии с вариантом. Результат нужно записать в текстовый файл out.txt.

1. Подсчитать сколько раз в тексте встречается заданное слово.
2. Заменить по всему тексту заданное слово на другое.
3. Определить сколько целых чисел в тексте
4. Удалить из текста лишние пробелы между словами
5. Удалить все вхождения заданного слова из текста
6. Подсчитать количество слов начинающихся с буквы 'А'
7. Подсчитать количество слов заканчивающихся буквой 'и'
8. Подсчитать количество слов содержащих слог 'ло'
9. Подсчитать количество открывающих и закрывающих круглых скобок
10. Удалить последовательности символов, заключенные в фигурные скобки
11. Определить наибольшее число подряд идущих одинаковых символов
12. Определить, каких символов больше: цифр или латинских букв

Задание 2. Постройте граф, моделирующий дорожную сеть, на основе данных фрагмента карты (например, 2ГИС или Яндекс.Карты) любой местности по Вашему выбору с числом вершин не менее 15.
Постройте с пользователем диалог таким образом, чтобы он мог запрашивать у программы информацию о кратчайшем расстоянии между двумя пунктами карты (вершинами графа), при этом получал на экране информацию о кратчайшем расстоянии и путь.
Граф должен быть задан в табличной форме в виде матрицы смежности в файле. Внутренний формат файла можно выбрать по Вашему усмотрению (простой текстовый файл с разделителями-пробелами, файл формата csv, файл формата MS Excel и пр.).