**Задания к учебной практике**

Задание 1. Разработать программу, реализующую алгоритм автоматической группировки (кластеризации) k-средних (обычный, с евклидовой мерой расстояния). Использовать любой алгоритмический язык программирования. Входные данные должны быть представлены в текстовом файле следующего формата.

Первая строка содержит 2 числа, разделенные пробелами:
N k

здесь N – число кластеризуемых объектов (векторов данных), k – число групп (кластеров), на которые разбиваются объекты.

Далее во входном файле следуют N строк, каждая из которых содержит два числа — признаки, разделенные пробелами. Каждая строка представляет один объект, характеризуемый двумя признаками.

Пример файла:

----------начало примера----------
```
10 3
12.3 17.9
13.0 19.4
11.3 17.8
11.4 18.8
13.1 19.1
12.3 17.9
13.0 19.4
11.3 17.8
11.4 18.8
13.1 19.1
```
----------конец примера----------

В выходном файле должен иметь следующий формат. Каждая строка содержит одно число — номер кластера, которому принадлежит объект.

Программу можно реализовать в предположении, что число объектов N будет не более 100000, а число кластеров — не более 100.

Описание алгоритма k-средних смотрите в методичке.
Подсказки:
- Сам алгоритм довольно прост, но нужно тщательно продумать все структуры данных, в которых будут храниться исходные и промежуточные данные.
- Вам обязательно понадобится реализовать функцию, вычисляющую евклидово расстояние между двумя точками.
- Исходные данные удобно представить двумерным массивом.
- Понадобится двумерный массив, в котором будут храниться координаты k центроидов (центров или «центров масс») кластеров.
- Понадобится еще и массив (одномерный), в котором будут храниться номера кластеров, соответствующие каждому объекту.
- координаты k начальных центроидов выбираются случайным образом из массива исходных данных.

- далее координаты центроида для каждого кластера рассчитываются как среднее арифметическое сооветствующей координаты всех объектов, входящих в кластер.
- поищите примеры готовой реализации алгоритма в англоязычном сегменте Интернета.
- пошаговое описание реализации алгоритма в MS Excel есть в методичке. Можно попробовать поработать сперва в Excel, чтобы понять принцип.

Задание 2. Разработать программу, реализующую алгоритм автоматической группировки (кластеризации) k-средних с манхэттенской (прямоугольной) мерой расстояния. Использовать любой алгоритмический язык программирования.

Формат файлов и подсказки см. в задании 1.

Дополнительные подсказки:
- Вам обязательно понадобится реализовать функцию, вычисляющую прямоугольное (манхэттенское) расстояние между двумя точками.
- координаты k начальных центроидов выбираются случайным образом из массива исходных данных.
- далее координаты центроида для каждого кластера рассчитываются как медианное (т.е.среднее по номеру — не путать со средним арифметическим) значение сооветствующей координаты всех объектов, входящих в кластер.
- задание очень похоже на Задание 1, поэтому удобно объединиться в группу и реализовывать два алгоритма одновременно.

Задания 3, 4 и 5 являются частями одного большого задания: «разработка системы группировки текстов на основе алгоритма k-медоид с метрикой Жаккара»

Задание 3. Пусть имеется входной текстовый файл, содержащий текст на английском языке (обрабатываются только буквы английского алфавита, все остальные символы игнорируются). Программа должна выводить в выходной файл множество всех слов, встречающихся во входном файле. Слова в выходном файле не должны повторяться. В выходном файле все слова должны быть выведены в одну строку и должны быть разделены пробелами.
Программу можно реализовать в предположении, что не бывает слов длиннее 50 букв, и не бывает текстов с числом слов более 10000. В выходном файле все слова должны быть записаны заглавными буквами.

Пример входного файла:
----------начало примера ----------
In mathematics and computer science, an algorithm is a set of instructions, to solve a class of problems or perform a computation. Algorithms are unambiguous specifications to perform calculation and other tasks.

An algorithm can be expressed within a finite amount of space and time[1] and in a well-defined formal language[2] for calculation of a function.[3]
----------конец примера ----------

Соответствующий пример выходного файла:
----------начало примера (все должно быть в одну строку) ----------
IN MATHEMATICS AND COMPUTER SCIENCE AN ALGORITHM IS A SET OF INSTRUCTIONS TO SOLVE CLASS PROBLEMS OR PERFORM COMPUTATION ALGORITHMS ARE UNAMBIGUOUS SPECIFICATIONS PERFORM CALCULATION OTHER TASKS CAN BE EXPRESSED WITHIN FINITE AMOUNT SPACE TIME WELL DEFINED FORMAL LANGUAGE FUNCTION
---------- конец примера ----------

Подсказки:
- нужно будет организовать массив строковых данных (МассивСлов), в котором каждый элемент будет представлять слово, которое уже встретилось в тексте.
- все знаки препинания, цифры, не-английские буквы, переводы строк и т. п. игнорируются и приравниваются к пробелу.
- алгоритм считывания слов можно организовать так:

Шаг 1. Предыдущий_символ=““; Слово=““; Число_слов=0.
Шаг 2. Считать текущий_символ из входного файла; Если достигнут конец файла, то текущий_символ=“ “.
Шаг 3. Если текущий_символ является английской буквой, то перевести его в заглавные буквы.
Шаг 4. Если текущий_символ является английской буквой и предыдущий_символ является английской буквой, то Слово=Слово+текущий_символ (т. е. дополнить слово текущим символом);
Шаг 5. Если текущий_символ не является английской буквой, но предыдущий_символ является английской буквой, то выполнить процедуру ПроверкаИДобавлениеСлова();
Шаг 6. Если достигнут конец файла, то вывести МассивСлов в выходной файл и остановить программу.
Шаг 7. предыдущий_символ=текущий_символ;

Шаг 8. Перейти к Шагу 2.

```
Процедура ПроверкаИДобавлениеСлова();
Для i от 0 до Число_слов-1 выполнять:
    Если МассивСлов[i]==Слово, то возврат;
КонецЦикла;
МассивСлов[ЧислоСлов]=Слово;
ЧислоСлов=ЧислоСлов+1.
```

- вместо массивов строк в конкретном выбранном Вами языке программирования, возможно, имеются более удобные для этой задачи структуры данных (например, списки). Массивы в предыдущей подсказке используются просто полтому, что они есть в любом языке программирования.

Задание 4. Имеется входной текстовый файл, в котором каждая строка представляет собой множество слов, разделенных пробелом. Слова в пределах строки не повторяются.

Составить программу, рассчитывающую матрицу попарных расстояний в метрике Жаккара (в жаккаровых/жаккардовых расстояний, Jaccard metric) между множествами слов.

Пример входного файла:
----------начало примера ----------\
```
CHINA RUSSIA EGYPT GERMANY
GERMANY FRANCE MONACO CHINA
GERMANY RUSSIA FRANCE
CHINA VIETNAM THAILAND
MONACO SAN MARINO LICHTENSTEIN ANDORRA
RUSSIA KAZAKHSTAN BELARUS
CHINA RUSSIA KAZAKHSTAN INDIA
```
----------конец примера ----------

Матрица попарных расстояний записывается в выходной файл следующего формата.
В первой строке содержится одно число N — число объектов (множеств), равное числу строк входного файла.
Далее следуют N строк, в каждой из которых N чисел, разделенных пробелами — расстояния между объектами.

Пример выходного файла (не имеет отношения к приведенному примеру входного файла):
---------- начало примера----------
```
5
0 0.2 0.2 0.4 1
0.2 0 0.8 0.6 0.6
0.2 0.8 0 0.2 1
0.4 0.6 0.2 0 0.4
1 0.6 1 0.4 0
```
----------конец примера----------
(реальный фал будет намного больше и будет содержать не 5, а, например, 50 или 500 строк данных).

Подсказки.

- обратите внимание, что матрица расстояний — всегда симметричная, по диагонали — всегда нули.
- необходимо будет реализовать функцию, вычисляющую расстояние Жаккара (см.методичку) между двумя строками, представляющими множества слов. Эта функция сперва должна разложить каждую из строк на отдельные слова (т. е. сформировать два массива строк, каждый элемент массивов — это строка). Затем подсчитать число слов, имеющихся в обоих массивах («пройтись» по первому массиву, сравнивая его элементы с каждым элементом второго массива и подсчитывая число совпадений), а потом подсчитать общее число слов в обоих множествах (без повторений). Это число равно ЧИСЛО_СЛОВ_В_ПЕРВОМ_МНОЖЕСТВЕ + ЧИСЛО_СЛОВ_ВО_ВТОРОМ_МНОЖЕСТВЕ - ЧИСЛО_СОВПАДЕНИЙ.

Рассстояние Жаккара можно будет вычислить как 1- ЧИСЛО_СОВПАДЕНИЙ/ОБЩЕЕ_ЧИСЛО_СЛОВ.

- если такая функция реализована, то далее запускаем ее в двойном цикле:

Для каждого i от 1 до N:

    Для каждого j от 1 до N:

        Матрица_расстояний[i-1,j-1]=Расстояние_Жаккара_от_строки_i_до_строки_j.

    Конец цикла

конец цикла

Далее достаточно вывести матрицу в текстовый файл.

Задание 5.

Составить программу, реализующую алгоритм PAM (Partition Around Medoids), разделяющую множество объектов, представленное матрицей попарных расстояний, на два кластера.

Матрица попарных расстояний содержится во входном текстовом файле следующего формата.
В первой строке содержится одно число N — число объектов (множеств).
Далее следуют N строк, в каждой из которых N чисел, разделенных пробелами — расстояния между объектами.

Пример входного файла:

---------- начало примера----------

```
5
0 0.2 0.2 0.4 1
0.2 0 0.8 0.6 0.6
0.2 0.8 0 0.2 1
0.4 0.6 0.2 0 0.4
1 0.6 1 0.4 0
```

----------конец примера----------

(реальный фал будет намного больше и будет содержать не 5, а, например, 50 или 500 строк данных).

Входной файл в этом задании является выходным файлом предыдущего задания (можно скооперироваться для выполнения обоих заданий).

Подсазка:

Алгоритм PAM можно описать следующим образом.
Пусть D[] - двумерный массив, в котором содержится матрица попарных расстояний, число объектов равно N.
Щаг 0. Выбрать случайным образом два целых числа i1, i2 – номера медоидов — в диапазоне от 0 до N-1.
Шаг 1. Подсчитать S0=сумма_расстояний(); // (функция описана ниже)
Шаг 2. Для каждого i от 0 до N-1
Шаг 2.1 Присвоить i3=i1; i1=i;
Шаг 2.2. Подсчитать S1=сумма_расстояний();
Шаг 2.3. Если S1<S0, то i3=i; S0=S1.
Шаг 2.4 Конец цикла;
Шаг 3. Присвоить i1=i3.
Шаг 4. Для каждого i от 0 до N-1
Шаг 4.1. Присвоить i3=i2; i2=i;
Шаг 4.2. Подсчитать S2=сумма_расстояний();
Шаг 4.3. Если S2<S0, то i3=i; S0=S2.
Шаг 4.4 Конец цикла;
Шаг 5. Присвоить i2=i3.
Шаг 6. Перейти к Шагу 1.

Алгоритм реализует бесконечный цикл (шаг 6), но можно ограничить число итераций, например, десятком или сотней.

Функция сумма_расстояний():
Шаг 0. Присвоить S=0.0;
Шаг 1. для каждого номера объекта j от 0 до N-1 рассчитать расстояние R до ближайшего медоида. Оно будет равно минимальному из двух значений: D[j,i1] и D[j,i2]. Прибавить S=S+R;
Шаг 2. Возвратить S.