Базовые темы курсовых

Алгоритм LZ77

Ваша программа должна читать входные данные из стандартного потока ввода и выводить ответ на стандартный поток вывода.

Вам будут даны входные файлы двух типов.

Первый тип:

```
compress
<text>
```

Текст состоит только из малых латинских букв. В ответ на него вам нужно вывести тройки, которыми будет закодирован данный текст.

Второй тип:

```
decompress
<triplets>
```

Вам даны тройки (<offset, len, char>) в которые был сжат текст из малых латинских букв, вам нужно его разжать.

Алгоритм LZW

Ваша программа должна читать входные данные из стандартного потока ввода и выводить ответ на стандартный поток вывода.

Вам будут даны входные файлы двух типов.

Первый тип:

```
compress
<text>
```

Текст состоит только из малых латинских букв. В ответ на него вам нужно вывести коды, которыми будет закодирован данный текст.

Второй тип:

```
decompress
<codes>
```

Вам даны коды в которые был сжат текст из малых латинских букв, вам нужно его разжать.

Начальный словарь состоит выглядит следующим образом:

```
a -> 0
b -> 1
c -> 2
...
x -> 23
y -> 24
z -> 25
EOF -> 26
```

Алгоритм BWT+MTF+RLE

Ваша программа должна читать входные данные из стандартного потока ввода и выводить ответ на стандартный поток вывода.

Вам будут даны входные файлы двух типов.

Первый тип:

```
compress
<text>
```

Текст состоит только из малых латинских букв. В ответ на него вам нужно вывести коды, которыми будет закодирован данный текст.

Второй тип:

```
decompress
<codes>
```

Вам даны коды в которые был сжат текст из малых латинских букв, вам нужно его разжать.

Для MTF используется следующее начальное распределение кодов:

```
a -> 0
b -> 1
c -> 2
...
x -> 23
y -> 24
z -> 25
```

diff

Ваша программа должна читать входные данные из стандартного потока ввода и выводить ответ на стандартный поток вывода.

В двух строках вам даны два текста, со словами разделёнными пробелами. Вам необходимо найти наибольшую общую подпоследовательность слов в этих текстах без использования динамического программирования.

Автоматическая классификация документов

Ваша программа должна читать входные данные из стандартного потока ввода и выводить ответ на стандартный поток вывода.

Вам даны данные в следующем формате:

```
<training> <test>
<class_1>
<data_1>
<class_2>
<data_2>
...
<class_training>
<data_training>
<query_1>
<query_2>
...
<query_test>
```

В первой строке даны два числа: количество обучающих данных и количество тестовых запросов. Обучающие данные представлены парами строк, в первой строке дано одно число 0 или 1, отвечающее за номер класса, во второй строке дан весь текст документа. Тестовые данные содержат по одному документу в строке, которые необходимо классифицировать.

В ответ на каждый тестовый запрос выведите единственное число 0 или 1 — предполагаемый класс документа.

Необходимо реализовать наивный байесовский классификатор, который будет обучен на первой части входных данных и классифицировать им вторую часть.

Текстовый поиск

Ваша программа должна читать входные данные из стандартного потока ввода и выводить ответ на стандартный поток вывода.

Реализуйте инвертированный индекс с дельта и Simple9 кодированием, затем проведите поиск текстов содержащих заданные наборы слов.

В первой строке входного файла вам даны два числа n и m — количество текстов и количество запросов в файле. В следующих n строках даны тексты, по одному тексту в строке, представленные наборами слов разделёнными пробелами. В следующих m строках вам даны запросы по одному в строке представленные наборами слов разделённых пробелами.

В ответ на каждый запрос выведите два числа количество подходящих под запрос документов и следующую сумму: $(\sum_i t_i \cdot i)$ — где t_i — номера текстов в которых встречались все слова из запроса c_i и t_i начинаются с 1.

Эвристический поиск в графе

Ваша программа должна читать входные данные из стандартного потока ввода и выводить ответ на стандартный поток вывода.

Реализуйте алгоритм A^* для неориентированного графа.

В первой строке вам даны два числа n и m — количество вершин и рёбер в графе. В следующих n строках вам даны пары чисел, описывающие положение вершин графа в двумерном пространстве. В следующих m строках даны пары чисел в отрезке от 1 до n, описывающие рёбра графа.

Далее дано число q и в следующих q строках даны запросы в виде пар чисел на поиск кратчайшего пути между двумя вершнами.

В ответ на каждый запрос выведите единственное число — длину кратчайшего пути между заданными вершинами с абсолютной либо относительной точностью 10^{-6}, если пути между вершинами не существует выведите 1". Расстояние между соседями вычисляется как простое евклидово расстояние на плоскости.

Эвристический поиск на решётках

Ваша программа должна читать входные данные из стандартного потока ввода и выводить ответ на стандартный поток вывода.

Данные

Реализуйте алгоритм A^* для графа на решётке.

Первые четыре строки входного файла выглядят следующим образом:

```
type octile
height <x>
width <y>
map
```

Где "x"и "y— высота и ширина карты соответственно.

Далее в x строках задана сама карта в виде решётки символов, в которой символы '.' и 'G' обозначают проходимые клетки. Переход между ячейками возможен только по сторонам.

Далее дано число q и в следующих q строках даны запросы в виде четвёрок чисел на поиск кратчайшего пути между двумя позициями в решётке.

В ответ на каждый запрос выведите единственное число — расстояние между ячейками из запроса.

Быстрое преобразование Фурье

Реализуйте алгоритм быстрое преобразование Фурье для действительного сигнала.

В качестве первого аргумента вашей программе передаётся название mp3 файла который необходимо обработать. Для каждых 4096 отсчётов

с шагом задаваемым вторым аргументом. Перед преобразованием Фурье необходимо подействовать на отсчёты окном Ханна.

В качестве результата для каждого набора отсчётов выведите наибольшее заначение по абсолютной величине полученное после преобразования Фурье.

Раздел Appendix A.

Персистентные структуры данных

Вам дан набор горизонтальных отрезков, и набор точек, для каждой точки определите сколько отрезков лежит строго над ней.

В первой строке вам даны два числа n и m — количество отрезков и количество точек соответственно. В следующих n строках ва заданы отрезки, в виде троек чисел l, r и h — координаты x левой и правой границ отрезка и координата y отрезка соответственно. В следующих m строках вам даны пары чисел — координаты точек.

Для каждой точки выведите количество отрезков над ней.

Ваше решение должно работать *online*, то есть должно обрабатывать запросы по одному после построения необходимой структуры данных по входным данным. Чтение входных данных и запросов вместе и построение по ним общей структуры запрещено.