1 Задания

1.1 Автоматическая классификация документов

Реализуйте систему, которая на основе базы вопросов и тегов к ним, буде предлагать варианты тегов, которые подходят к новым вопросам.

Формат запуска программы в режиме обучения:

```
./prog learn --input <input file> \
             --output <stats file>
```

Ключ	Значение
`--input`	входной файл с вопросами
`--output`	выходной файл с рассчитанной статистикой

Формат запуска программы в режиме классификации:

```
./prog classify --stats <stats file> \
                --input <input file> \
                --output <output file>
```

Ключ	Значение
`--stats`	файл со статистикой полученной на предыдущем этапе
`--input`	входной файл с вопросами
`--output`	выходной файл с тегами к вопросам

Формат входных файлов при обучении:

```
<Количество строк в вопросе [n]>
<Тег 1>,<Тег 2>,...,<Тег m>
<Заголовок вопроса>
<Текст вопроса [n строк]>
```

Формат входных файлов при запросах:

```
<Количество строк в вопросе [n]>
<Заголовок вопроса>
<Текст вопроса [n строк]>
```

Формат выходного файла: для каждого запроса в отдельной строке выводится предполагаемый набор тегов, через запятую.

1.2 Исправление ошибок в поисковых запросах

Реализуйте систему, которая на основе статистики о предыдущих запросах пользователей, будет предлагать исправление опечаток в новых запросах.

Формат запуска программы в режиме обучения:

```
./prog learn --input <input file> \
             --output <stats file>
```

Ключ	Значение
`--input`	входной файл с запросами пользователей
`--output`	выходной файл с рассчитанной статистикой

Формат запуска программы в режиме исправления ошибок:

```
./prog correct --stats <stats file> \
               --input <input file> \
               --variants <number of correction variants required> \
               --output <output file>
```

Ключ	Значение
`--stats`	файл со статистикой полученной на предыдущем этапе
`--input`	входной файл с запросами
`--variants`	количество необходимых вариантов исправления
`--output`	выходной файл с вариантами исправлений

Формат выходного файла: на каждый запрос выводится требуемое количество исправлений по одному исправлению в строке, если исправлений не хватает, то выводятся пустые строки до требуемого числа.

1.3 Параллельная внешняя сортировка

Осуществить параллельную сортировку файлов с парами ключ-значение, размер которых превышает доступный объём оперативной памяти.

Ключи: последовательности байт фиксированной длины не длиннее 1KB.

Значения: последовательности байт фиксированной длины не длиннее 16KB.

Формат запуска:

```
./prog --keys <keys file> \
       --values <values file> \
       --out-keys <keys output file> \
       --out-values <values output file>
```

Ключ	Значение
`--keys`	входной файл с ключами
`--values`	входной файл со значениями
`--out-keys`	выходной файл с ключами
`--out-values`	выходной файл со значениями

Формат файлов:

```
uint64_t [n] --- количество значений в файле
uint64_t [l] --- длина значений в байтах
(uint8_t * l) * n --- n значений по l байт
```

1.4 diff

Реализуйте аналог стандартной утилиты `diff`.

Утилита diff предназначена для выявления минимальных различий между двумя файлами, т.е. нахождениию минимальной последовательности операций вставки, удаления или замены строк в первом файле на строки из второго файла, чтобы в итоге получился второй файл.

Решение основанное на методе динамического программирования может претендовать только на оценку 3 в силу ограничения на размерности анализируемых файлов, вытекающих из большого размера таблицы, необходимой для метода динамического программирования.

Работа программы, процедура её запуска, формат ввода и вывода должны быть аналогична работе исходной утилиты diff. Должны поддерживаться следующие опции: -i, -E, -Z, -b, -w, -B, -t.

1.5 Архиватор

Необходимо реализовать два известных метода сжатия данных для сжатия одного файла.

Методы сжатия выбираются из следующих групп:

- Арифметическое кодирование, кодирование по Хаффману
- LZ77, LZW, BWT+MTF+RLE

Формат запуска должен быть аналогичен формату запуска программы gzip. Должны быть поддерживаться следующие ключи: -c, -d, -k, -l, -r, -t, -1, -9. Должно поддерживаться указание символа дефиса в качестве стандартного ввода.

1.6 Текстовый поиск

Реализуйте систему для поиска статей по заданным словам.

Формат запуска в режиме индексирования:

```
./prog index --input <input file> \
             --output <index file>
```

Ключ	Значение
`--input`	входной файл со статьями
`--output`	выходной файл с индексом

Формат запуска в режиме поиска:

```
./prog search --index <index file> \
              --input <input file> \
              --output <output file> \
              [--full-output]
```

Ключ	Значение
`--index`	входной файл с индексом
`--input`	входной файл с запросами
`--output`	выходной файл с ответами на запросы
`--full-output`	переключение формата выходного файла на подробный

Формат файлов для индексации:

```
<doc id="12" url="https://en.wikipedia.org/wiki?curid=12" title="Anarchism">
<текст статьи>
</doc>
<doc id="25" url="https://en.wikipedia.org/wiki?curid=25" title="Autism">
<текст статьи>
</doc>
```

Формат файла с запросами:

```
<word 1>
<word 1> & <word 2>
<word 1> | <word 2>
~<word 1>
~(<word 1> & <word 2> & <word 3>) | (<word 4> & (<word 5> | ~<word 6>))
```

Формат выходного файла:

Если опция `--full-output` не указана: на каждый запрос в отдельной строке выводится количество документов подпадающих под запрос.

Если опция `--full-output` указана: на каждый запрос выводится отдельная строка, с количеством документов подпадающих под запрос, а затем названия всех документов подпадающих под запрос по одному названию в строке.

1.7 Поиск на графе

Реализуйте систему для поиска пути в графе дорог с использованием эвристических алгоритмов.

```
./prog preprocess --nodes <nodes file> \
                  --edges <edges file> \
                  --output <preprocessed graph>
```

Ключ	Значение
`--nodes`	входной файл с перекрёстками
`--edges`	входной файл с дорогами
`--output`	выходной файл с графом

```
./prog search --graph <preprocessed graph> \
              --input <input file> \
              --output <output file> \
              [--full-output]
```

Ключ	Значение
`--graph`	входной файл с графом
`--input`	входной файл с запросами
`--output`	выходной файл с ответами на запросы
`--full-output`	переключение формата выходного файла на подробный

Файл узлов:

```
<id> <lat> <lon>
```

Файл рёбер:

```
<длина дороги в вершинах [n]> <id 1> <id 2> ... <id n>
```

Выходной файл:

Если опция `--full-output` не указана: на каждый запрос в отдельной строке выводится длина кратчайшего пути между заданными вершинами с относительной погрешностью не более 1e-6.

Если опция `--full-output` указана: на каждый запрос выводится отдельная строка, с длиной кратчайшего пути между заданными вершинами с относительной погрешностью не более 1e-6, а затем сам путь в формате как в файле рёбер.

Расстояние между точками следует вычислять как расстояние между точками на сфере с радиусом 6371км, если пути между точками нет, вывести -1 и длину пути в вершинах 0.

1.8 Аудиопоиск

Реализуйте систему для поиска аудиозаписи по небольшому отрывку.

```
./prog index --input <input file> \
             --output <index file>
```

Ключ	Значение
`--input`	входной файл с именами файлов для индексации
`--output`	выходной файл с индексом

```
./prog search --index <index file> \
              --input <input file> \
              --output <output file>
```

Ключ	Значение
`--index`	входной файл с индексом
`--input`	входной файл с запросами
`--output`	выходной файл с ответами на запросы

Все файлы будут даны в формате MP3 с частотой дискретизации 44100Гц.

Входные файлы содержат в себе имена файлов с аудио записями по одному файлу в строке.

Результатом ответа на каждый запрос является строка с названием файла, с которым произошло совпадение, либо строка "`! NOT FOUND`", если найти совпадение не удалось.

1.9 Пространственный поиск

Реализуйте систему для определения принадлежности точки одному из многоугольников на плоскости.

```
./prog index --input <input file> \
             --output <index file>
```

Ключ	Значение
`--input`	входной файл с многоугольниками
`--output`	выходной файл с индексом

```
./prog search --index <index file> \
              --input <input file> \
              --output <output file>
```

Ключ	Значение
`--index`	входной файл с индексом
`--input`	входной файл с запросами
`--output`	выходной файл с ответами на запросы

Формат входного файла:

```
<количество вершин многоугольника [n]> <x 1> <y 1> <x 2> <y 2> ... <x n> <y n>
```

Формат файла запросов:

```
<x i> <y i>
```

Для каждого запроса выведите номер многоугольника внутри которго содержится точка (многоугольники нумеруются с нуля) либо -1.

1.10 Поиск ближайших соседей

Дано множество точек в многомерном пространстве. Для каждой точки из файла с запросами вам необходимо вывести номер ближайшей к ней точки из исходного множества в смысле простого евклидова расстояния. Если ближайших точек несколько, то выведите номер любой из них.

Формат входных файлов при построении структуры:

```
./prog build --input <input file> \
             --output <stats file>
```

Ключ	Значение
`--input`	входной файл с точками
`--output`	выходной файл со структурой для поиска

Формат входного файла:

```
<dimensions>
<x_1,1> <x_1,2> ... <x_1,dimensions>
<x_2,1> <x_2,2> ... <x_2,dimensions>
...
```

Формат запуска программы в режиме поиска:

```
./prog search --struct <struct file> \
                --input <input file> \
                --output <output file>
```

Ключ	Значение
`--struct`	файл со структурой для поиска, полученной на предыдущем этапе
`--input`	входной файл с запросами
`--output`	выходной файл с ответами на запросы

Формат входного файла:

```
<dimensions>
<q_1,1> <q_1,2> ... <q_1,dimensions>
<q_2,1> <q_2,2> ... <q_2,dimensions>
...
```

2 Литература

- Керниган Б., Пайк Р. “Практика программирования”
- Ватолин Д., Ратушняк А., Смирнов М., Юкин В. “Методы сжатия данных. Устройство архиваторов, сжатие изображений и видео.”
- Кормен Т., Лейзерсон Ч., Ривест Р. “Алгоритмы. Построение и анализ”
- Кнут Д. “Искусство программирования”
- Седжвик Р. “Фундаментальные алгоритмы на C++”
- Christopher D. Manning, Prabhakar Raghavan and Hinrich Schütze, “Introduction to Information Retrieval”
- Tao Li, Mitsunori Ogihara, George Tzanetakis, “Music Data Mining”