Описание структуры дерева кодирования

Структура дерева кодирования

Дерево кодирования, реализованное в данном проекте, представляет собой бинарное дерево, построенное на основе частотного анализа символов исходного файла. Оно состоит из листьев (*leaves*) и внутренних узлов (*internal nodes*). Все символы исходного файла отображаются в листьях дерева, а внутренние узлы содержат частоту появления символов, объединённых в их поддеревьях.

Принципы кодирования

- Дерево кодирования строится из частотного словаря, где каждому символу соответствует частота его появления в исходном файле.
- Для каждого символа создаётся уникальный путь от корня дерева до листа:
  - Левое ветвление обозначается как «0».
  - Правое ветвление обозначается как «1».
- Все коды являются префиксными: ни один код символа не является началом другого. Это гарантирует уникальное декодирование.

Структура классов

Класс `Node`

Класс `Node` представляет узел дерева кодирования и содержит следующие поля:

- `Integer frequency`: частота символов, представленных данным узлом.
- `Node left`: ссылка на левого потомка (для кодирования «0»).
- `Node right`: ссылка на правого потомка (для кодирования «1»).

Методы:

- `int compareTo(Node n)`: сравнение узлов по частоте (используется в приоритетной очереди).
- `void fillCodeMap(String character, HashMap<Byte, String> codeMap)`: рекурсивное заполнение словаря кодов символов.

Класс `Leaf`

Класс `Leaf` наследуется от `Node` и представляет лист дерева, ассоциированный с конкретным символом. Поля:

- `Byte symbol`: байт, представляющий символ.
- `String prefix`: закодированный префикс для символа.

Методы:

- `String toString()`: возвращает строковое представление символа и его кода.
- `void fillCodeMap(String character, HashMap<Byte, String> codeMap)`: добавляет текущий символ и его код в словарь.

Алгоритм построения дерева кодирования

Дерево кодирования строится методом `buildHuffmanTree` класса `HuffmanCodec` следующим образом:

1. Рассчитывается частота появления каждого символа в исходном файле (`calculateWeightMap`).
2. Создаются начальные узлы (`Leaf`) для каждого символа с частотой и добавляются в приоритетную очередь (`PriorityQueue`), где узлы сортируются по частоте.
3. Узлы объединяются в дерево:
   - Из очереди извлекаются два узла с минимальной частотой.
   - Создаётся новый узел (`Node`) с суммарной частотой этих двух узлов. Левый потомок узла связывается с первым узлом, правый — со вторым.
   - Новый узел добавляется обратно в очередь.
4. Процесс повторяется, пока в очереди не останется один узел — корень дерева.

Структура кодов

Словарь кодов (*code map*) представляет собой отображение символов на их бинарные коды. Пример для строки `"HELLO"`:

```
H 00
E 01
L 10
O 11
```

Каждая запись соответствует пути от корня дерева до листа, представляющего символ.