Структура дерева кодирования

Дерево кодирования в алгоритме Хаффмана является бинарным деревом. Оно уникальным образом кодирует каждый из используемых в кодируемом файле символов по префиксному правилу: ни один из кодов какого-либо элемента не является началом кода другого элемента. Это свойство кодов Хаффмана позволяет однозначно декодировать зашифрованные сообщения. Все внутренние узлы дерева Хаффмана являются вспомогательными, то есть они лишь служат для хранения листьев и других вспомогательных узлов. Каждый лист хранит в себе кодируемый символ и соответствующий ему код в алгоритме Хаффмана для данного конкретного файла.

Компоненты дерева кодирования

Дерево кодирования состоит из узлов, которые представлены в виде класса **Symbol**.

Его поля:

Char symbol – символ, который представляет данный узел в дереве Хаффмана

Int occurrence – частота, с которой этот символ встречается в тексте

Symbol left и Symbol right – левый и правый потомки узла соответственно. Если узел является листом, оба его потомка равны **null**. Левый потомок соответствует ветви кодирования 0, правый – ветви 1.

Если узел является внутренним, то его частота – это сумма частот его потомков.

Все дерево целиком представлено классом **Tree.**

Его поля:

Symbol root – корень дерева Хаффмана

Дерево наполняет предоставленный ему **HashMap** в методе **huffmanCodes** кодами своих листьев.

Алгоритм построения дерева

Дерево Хаффмана строится по следующему алгоритму. У каждого символа в файле подсчитывается количество раз, которое этот символ встречается в файле. Далее все символы упорядочиваются по возрастанию в некоторый список (не в программном понимании слова «список»). Из списка извлекаются два узла с самыми маленькими частотами. На их основе создается новый узел-родитель, частота которого является суммой частот узлов, которые являются его детьми. Далее этот созданный узел помещается в список упорядоченным образом. Процесс извлечения узлов повторяется до тех пор, пока в списке не останется одного элемента, который и будет корнем дерева.

Данный алгоритм построен таким образом, чтобы символам, которые часто встречаются в тексте, соответствовали более короткие коды, что позволяет уменьшить итоговый объем сообщения.

Программное представление алгоритма кодирования

В программе алгоритм кодирования представлен в виде последовательного использования классов **Occurrences, Ordered** и **TreeBuilder. Occurrences** читает переданный ему файл и подсчитывает количество вхождений каждого символа в файл, храня информацию об этом в **HashMap<Character, Integer>. Ordered** упорядочивает полученные пары **<Character, Integer>** по возрастанию значения и хранит их в виде экземпляров класса **Symbol** в **PriorityQueue**. Далее **TreeBuilder** по описанному алгоритму строит дерево Хаффмана **Tree**.