Лекция 6:
Алгоритмы сжатия данных

Цель лекции: изучить основные виды и алгоритмы сжатия данных и научиться решать задачи сжатия данных по методу Хаффмана и с помощью кодовых деревьев.

Основоположником науки о сжатии информации принято считать Клода Шеннона. Его теорема об оптимальном кодировании показывает, к чему нужно стремиться при кодировании информации и насколько та или иная *информация* при этом сожмется. Кроме того, им были проведены опыты по эмпирической оценке избыточности английского текста. Шенон предлагал людям угадывать следующую букву и оценивал *вероятность* правильного угадывания. На основе ряда опытов он пришел к выводу, что *количество информации* в английском тексте колеблется в пределах 0,6 – 1,3 бита на символ. Несмотря на то, что результаты исследований Шеннона были по-настоящему востребованы лишь десятилетия спустя, трудно переоценить их *значение*.

Сжатие данных – это процесс, обеспечивающий уменьшение объема данных путем сокращения их избыточности. Сжатие данных связано с компактным расположением порций данных стандартного размера. Сжатие данных можно разделить на два основных типа:

- *Сжатие без потерь (полностью обратимое)* – это метод сжатия данных, при котором ранее закодированная порция данных восстанавливается после их распаковки полностью без внесения изменений. Для каждого типа данных, как правило, существуют свои оптимальные алгоритмы сжатия без потерь.
- *Сжатие с потерями* – это метод сжатия данных, при котором для обеспечения *максимальной степени* сжатия исходного массива данных часть содержащихся в нем данных отбрасывается. Для текстовых, числовых и табличных данных использование программ, реализующих подобные методы сжатия, является неприемлемыми. В основном такие алгоритмы применяются для сжатия аудио- и видеоданных, статических изображений.

Алгоритм сжатия данных (алгоритм архивации) – это *алгоритм*, который устраняет *избыточность* записи данных.

Введем ряд определений, которые будут использоваться далее в изложении материала.

Алфавит кода – множество всех символов входного потока. При сжатии англоязычных текстов обычно используют множество из 128 *ASCII* кодов. При сжатии изображений множество значений пиксела может содержать 2, 16, 256 или другое количество элементов.

Кодовый символ – наименьшая *единица* данных, подлежащая сжатию. Обычно символ – это 1 *байт*, но он может быть битом, тритом {0,1,2}, или чем-либо еще.

Кодовое слово – это последовательность кодовых символов из алфавита кода. Если все слова имеют одинаковую длину (число символов), то такой код называется *равномерным (фиксированной длины)*, а если же допускаются слова разной длины, то – *неравномерным (переменной длины)*.

Код – полное множество слов.

Токен – *единица* данных, записываемая в сжатый *поток* некоторым алгоритмом сжатия. *Токен* состоит из нескольких полей фиксированной или переменной длины.

Фраза – фрагмент данных, помещаемый в словарь для дальнейшего использования в сжатии.

Кодирование – процесс сжатия данных.

Декодирование – *обратный* кодированию процесс, при котором осуществляется восстановление данных.

Отношение сжатия – одна из наиболее часто используемых величин для обозначения эффективности метода сжатия.

$$Отношение\ сжатия = \frac{размер\ выходного\ потока}{размер\ входного\ потока}$$

Значение 0,6 означает, что данные занимают 60% от первоначального объема. Значения больше 1 означают, что выходной *поток* больше входного (отрицательное сжатие, или расширение).

Коэффициент сжатия – величина, обратная отношению сжатия.

$$Коэффициент\ сжатия = \frac{размер\ входного\ потока}{размер\ выходного\ потока}$$

Значения больше 1 обозначают сжатие, а значения меньше 1 – расширение.

Средняя длина кодового слова – это величина, которая вычисляется как взвешенная вероятностями сумма длин всех кодовых слов.

$L_{ср}=p_1L_1+p_2L_2+...+p_nL_n,$

где – вероятности кодовых слов;

$L_1,L_2,...,L_n$ – длины кодовых слов.

Существуют два основных способа проведения сжатия.

Статистические методы – методы сжатия, присваивающие коды переменной длины символам входного потока, причем более короткие коды присваиваются символам или группам символам, имеющим большую *вероятность* появления во входном потоке. Лучшие *статистические методы* применяют *кодирование* Хаффмана.

Словарное сжатие – это методы сжатия, хранящие фрагменты данных в "словаре" (некоторая *структура данных*). Если строка новых данных, поступающих на вход, идентична какому-либо фрагменту, уже находящемуся в словаре, в выходной *поток* помещается *указатель* на этот фрагмент. Лучшие словарные методы применяют метод Зива-Лемпела.

Рассмотрим несколько известных алгоритмов сжатия данных более подробно.

Метод Хаффмана

Этот *алгоритм кодирования информации* был предложен Д.А. Хаффманом в 1952 году. **Хаффмановское кодирование (сжатие)** – это широко используемый метод сжатия, присваивающий *символам алфавита* коды переменной длины, основываясь на вероятностях появления этих символов.

Идея алгоритма состоит в следующем: зная вероятности вхождения символов в исходный текст, можно описать процедуру построения кодов переменной длины, состоящих из целого количества битов. Символам с большей вероятностью присваиваются более короткие коды. Таким образом, в этом методе при сжатии данных каждому символу присваивается оптимальный *префиксный код*, основанный на вероятности его появления в тексте.

Префиксный код – это код, в котором никакое *кодовое слово* не является префиксом любого другого кодового слова. Эти коды имеют переменную длину.

Оптимальный префиксный код – это *префиксный код*, имеющий минимальную среднюю длину.

Алгоритм Хаффмана можно разделить на два этапа.

1. Определение вероятности появления символов в исходном тексте.

Первоначально необходимо прочитать исходный текст полностью и подсчитать вероятности появления символов в нем (иногда подсчитывают, сколько раз встречается каждый символ). Если при этом учитываются все 256 символов, то не будет разницы в сжатии текстового или файла иного формата.

2. Нахождение оптимального префиксного кода.

Далее находятся два символа a и b с наименьшими вероятностями появления и заменяются одним фиктивным символом x, который имеет вероятность появления, равную сумме вероятностей появления символов a и b. Затем, используя эту процедуру рекурсивно, находится оптимальный *префиксный код* для меньшего множества символов (где символы a и b заменены одним символом x). Код для исходного множества символов получается из кодов замещающих символов путем добавления 0 или 1 перед кодом замещающего символа, и эти два новых кода принимаются как коды заменяемых символов. Например, *код символа* a будет соответствовать коду x с добавленным нулем перед этим кодом, а для символа b перед *кодом символа* x будет добавлена единица.

Коды Хаффмана имеют уникальный *префикс*, что и позволяет однозначно их декодировать, несмотря на их переменную длину.

Алгоритм Хаффмана универсальный, его можно применять для сжатия данных любых типов, но он малоэффективен для файлов маленьких размеров (за счет необходимости сохранения словаря). В настоящее время данный метод практически не применяется в чистом виде, обычно используется как один из этапов сжатия в более сложных схемах. Это единственный *алгоритм*, который не увеличивает размер исходных данных в худшем случае (если не считать необходимости хранить таблицу перекодировки вместе с файлом).

Кодовые деревья

Рассмотрим реализацию алгоритма Хаффмана с использованием кодовых деревьев.

Кодовое дерево (дерево кодирования Хаффмана, H-дерево) – это *бинарное дерево*, у которого:

- листья помечены символами, для которых разрабатывается кодировка;
- узлы (в том числе корень) помечены суммой вероятностей появления всех символов, соответствующих листьям *поддерева*, корнем которого является соответствующий узел.

Метод Хаффмана на входе получает таблицу частот встречаемости символов в исходном тексте. Далее на основании этой таблицы строится *дерево* кодирования Хаффмана.

Алгоритм построения дерева Хаффмана.

Шаг 1. Символы входного алфавита образуют *список* свободных узлов. Каждый *лист* имеет *вес*, который может быть равен либо вероятности, либо количеству вхождений символа в сжимаемый текст.

Шаг 2. Выбираются два свободных узла дерева с наименьшими весами.

Шаг 3. Создается их родитель с весом, равным их суммарному весу.

Шаг 4. Родитель добавляется в *список* свободных узлов, а двое его детей удаляются из этого списка.

Шаг 5. Одной *дуге*, выходящей из родителя, ставится в соответствие *бит* 1, другой – *бит* 0.

Шаг 6. Повторяем шаги, начиная со второго, до тех пор, пока в списке свободных узлов не останется только один свободный узел. Он и будет считаться *корнем дерева*.

Существует два подхода к построению кодового дерева: от корня к листьям и от листьев к корню.

Пример построения кодового дерева. Пусть задана исходная последовательность символов:

```
aabbbbbbbbccccdeeeee.
```

Ее исходный объем равен 20 *байт* (160 *бит*). В соответствии с приведенными на рис. 1 данными (*таблица* вероятности появления символов,
кодовое *дерево* и *таблица* оптимальных *префиксных кодов*) закодированная исходная последовательность символов будет выглядеть следующим образом:

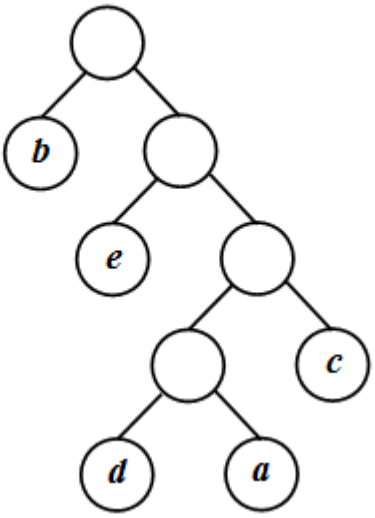
```
11011101000000001111111111111001010101010.
```

Следовательно, ее объем будет равен 42 бита. Коэффициент сжатия приближенно равен 3,8.

Вероятности появления символов

Символ	*Вероятность*
a	0,1
b	0,4
c	0,2
d	0,05
e	0,25

Кодовое дерево



Оптимальные префиксные коды

Символ	*Код*
a	1101
b	0
c	111
d	1100
e	10

Рис. 1. Создание оптимальных префиксных кодов

Классический *алгоритм* Хаффмана имеет один существенный недостаток. Для восстановления содержимого сжатого текста при *декодировании* необходимо знать таблицу частот, которую использовали при кодировании. Следовательно, *длина* сжатого текста увеличивается на длину таблицы частот, которая должна посылаться впереди данных, что может свести на нет все усилия по сжатию данных. Кроме того, необходимость наличия полной частотной статистики перед началом собственно кодирования требует двух проходов по тексту: одного для построения модели текста (таблицы частот и дерева Хаффмана), другого для собственно кодирования.

Для осуществления *декодирования* необходимо иметь кодовое *дерево*, которое приходится хранить вместе со сжатыми данными. Это приводит к некоторому незначительному увеличению объема сжатых данных. Используются самые различные форматы, в которых хранят это *дерево*. Обратим внимание на то, что узлы кодового дерева являются пустыми. Иногда хранят не само *дерево*, а исходные данные для его формирования, то есть сведения о вероятностях появления символов или их количествах. При этом процесс *декодирования* предваряется построением нового кодового дерева, которое будет таким же, как и при кодировании.

Ключевые термины

Сжатие данных – это процесс, обеспечивающий уменьшение объема данных путем сокращения их избыточности.

Сжатие без потерь (полностью обратимое) – это метод сжатия данных, при котором ранее закодированная порция данных восстанавливается после их распаковки полностью без внесения изменений.

Сжатие с потерями – это метод сжатия данных, при котором для обеспечения *максимальной степени* сжатия исходного массива данных часть содержащихся в нем данных отбрасывается.

Алгоритм сжатия данных (алгоритм архивации) – это *алгоритм*, который устраняет *избыточность* записи данных.

Алфавит кода – это множество всех символов входного потока.

Кодовый символ – это наименьшая *единица* данных, подлежащая сжатию.

Кодовое слово – это последовательность кодовых символов из алфавита кода.

Токен – это *единица* данных, записываемая в сжатый *поток* некоторым алгоритмом сжатия.

Фраза – это фрагмент данных, помещаемый в словарь для дальнейшего использования в сжатии.

Кодирование – это процесс сжатия данных.

Декодирование – это *обратный* кодированию процесс, при котором осуществляется восстановление данных.

Отношение сжатия – это величина для обозначения эффективности метода сжатия, равная отношению размера выходного потока к размеру входного потока.

Коэффициент сжатия – это величина, обратная отношению сжатия.

Средняя длина кодового слова – это величина, которая вычисляется как взвешенная вероятностями сумма длин всех кодовых слов.

Статистические методы – это методы сжатия, присваивающие коды переменной длины символам входного потока, причем более короткие коды присваиваются символам или группам символам, имеющим большую *вероятность* появления во входном потоке.

Словарное сжатие – это методы сжатия, хранящие фрагменты данных в некоторой структуре данных, называемой словарем.

Хаффмановское кодирование (сжатие) – это метод сжатия, присваивающий *символам алфавита* коды переменной длины основываясь на вероятностях появления этих символов.

Префиксный код – это код, в котором никакое *кодовое слово* не является префиксом любого другого кодового слова.

Оптимальный префиксный код – это *префиксный код*, имеющий минимальную среднюю длину.

Кодовое дерево (дерево кодирования Хаффмана, Н-дерево) – это *бинарное дерево*, у которого: *листья* помечены символами, для которых разрабатывается *кодировка*; узлы (в том числе корень) помечены суммой вероятностей появления всех символов, соответствующих листьям *поддерева*, корнем которого является соответствующий узел.

Выводы

1. Сжатие данных является процессом, обеспечивающим уменьшение объема данных путем сокращения их избыточности.
2. Сжатие данных может происходить с потерями и без потерь.
3. Отношение сжатия характеризует степень сжатия данных.

4. Существуют два основных способа проведения сжатия: *статистические методы* и словарное сжатие.

5. Алгоритм Хаффмана относится к *статистическим методам* сжатия данных.

6. Идея алгоритма Хаффмана состоит в следующем: зная вероятности вхождения символов в исходный текст, можно описать процедуру построения кодов переменной длины, состоящих из целого количества битов.

7. Коды Хаффмана имеют уникальный префикс, что и позволяет однозначно их декодировать, несмотря на их переменную длину.

8. Алгоритм Хаффмана универсальный, его можно применять для сжатия данных любых типов, но он малоэффективен для файлов маленьких размеров.

9. Классический алгоритм Хаффмана на основе кодового дерева требует хранения кодового дерева, что увеличивает его трудоемкость.