Оглавление

Лекция 10: Сортировки. Сжатие и защита информации.

15.11.2023

0.1 Сжатие по Хаффману

Пример. (из Романовского)
Рассмотрим текст из 223 знаков:

```
ехали_медведи_на_велосипеде_а_за_ними_кот_задом
_наперед_а_за_ним_комарики_на_воздушном_шарике_а
_за_ними_раки_на_хромой_собаке_волки_на_кобыле_
львы_в_автомобиле_зайчики_в_трамвайчике_жаба_
на_метле_едут_и_смеются_пряники_жуют
```

Применение сжатия по Хаффману дает следующий результат:

```
яь 3 жю 4 ыч 4 хш 4 йу 6 пяь 6 сб 8 жюыч 8 зхш 10
йупяь 12 тр 14 лд 14 сбжюыч 16 взхш 20 нк 22 мйупяь 24
отр 27 лдсбжюыч 30 евзхш 36 инк 42 амйупяь 48
отрлдсбжюыч 57 евзхш_ 76 инкамйупяь 90
отрлдсбжюычевзхш_ 133
```

Если взять кодовые последовательности для этих символов и закодировать ими текст, получим сжатие до 119 байтов, что почти вдвое меньше исходного текста.

Но код Хаффмана никак не учитывет закономерность распределения символов в тексте, поэтому он не является оптимальным.

0.2 Сжатие по Лемпелю-Зиву (LZ77, LZ78, LZO, LZMA, LZW etc.)

Алгоритм. Основная идея: Кодируемый текст разбивается на небольшие строки, каждая из которых составлена из одной из предыдущих строк (предопределена пустая строка, имеющая номер 0) и еще одного символа. Сначала записаны номера строк, затем сами строки, затем представление каждой из этих строк в виде пар (номер предыдущей строки, дополняющий символ).

Пример. На примере текста "aababaacbbbcccaaaaa": (сначала строки, затем представление строк в виде пар (номер предыдущей строки и дополняющий символ), затем номер строки)

```
 a ab aba ac  b bb  c cc aa aaa
0a 1b  2a 1c 0b 5b 0c 7c 1a  9a
 1  2   3  4  5  6  7  8  9  10
```

0.3 Метод Барроуза-Уилера (BWT)

Рассмотрим две строки: *aababbbbaa* и *aaaaabbbbb*. Вторая лучше с точки зрения сжатия, потому что она устроена проще.

Обычные тексты устроены сложнее, в них не просто буквы повторяются, а целые буквосочетания (например, артикли).

Идея преобразования Барроуза-Уилера (BWT — Barrous-Wheeler transformation) в том, чтобы так переставить буквы, чтобы:

1. одинаковые буквы по возможности шли подряд;
2. по преобразованной строке можно восстановить исходную.

Но это только преобразование, а не сжатие, ведь перестановки букв длину текста не уменьшают.

Алгоритм.
1. В конец строки добавляем специальный символ, минимальный лексикографически (для обратного преобразования)
2. Выписываем всем возможные циклические сдвиги строки.
3. Сортируем их лексикографически.
4. В качестве результата берем последний столбец полученной матрицы.

Обратное преобразование:

1. Берем исходную строку и записываем в первый столбец матрицы.

2. Сортируем строки матрицы лексикографически и записываем в следующий столбец.
3. Записываем в следующий столбец предыдущий + исходная строка слева.
4. Продолжаем, пока не получим столбец, в котором длина строк равна длине исходной строки, берем из этого столбца ту строку, в которой послений символ – специальный.

Пример. Имеется строка каркаркар. Добавляем в конец специальный символ, например, $.

```
каркаркар$      $каркаркар
аркаркар$к      ар$каркарк
ркаркар$ка      аркар$карк
каркар$кар      аркаркар$к
аркар$карк      кар$каркар
ркар$карка      каркар$кар
кар$каркар      каркаркар$
ар$каркарк      р$каркарка
р$каркарка      ркар$карка
$каркаркар      ркаркар$ка
```

Получили строку ркккрр$ааа, которая и будет результатом преобразования.

Обратное преобразование:

```
0 1  2  3  4     ...      n
р $ р$ $к р$к    ...   $каркаркар
к а ка ар кар    ...   ар$каркарк
к а ка ар кар    ...   аркар$карк
к а ка ар кар    ...   аркаркар$к
р к рк ка рка    ...   кар$каркар
р к рк ка рка    ...   каркар$кар
$ к $к ка $ка    ...   каркаркар$
а р ар р$ ар$    ...   р$каркарка
а р ар рк арк    ...   ркар$карка
а р ар рк арк    ...   ркаркар$ка
```

Получили строку каркаркар$

Замечание. (на лекции этого не было, но посчитал это важным.) Преобразование и в ту, и в другую сторону можно реализовать за линейное время. (докажите сами...)

Для того, чтобы преобразование Барроуза-Уилера имело смысл, нужно

применить к полученной строке алгоритм сжатия.

Алгоритм (MTF (Move To Front)). Заведем вспомогательный массив, в котром будем хранить уникальные символы на текущей итерации. При встрече символа будет вставлять его в начало массива. Также заведем массив индексов, в котором будем хранить позиции символов в массиве уникальных символов.

1. Начальное положение: в массив уникальных символов записываем текущий элемент, в массив индексов записываем 0.
2. На i-ой итерации: если символа нет в массиве уникальных символов, то записываем его в начало массива, иначе находим его позицию в массиве и записываем эту позицию в массив индексов.

По итогу получим на выходе алфавит и набор индексов, по которым можно легко декодировать исходную строку.

Пример.

```
  i
 0: р 0 р
 1: к 0 кр
 2: к 1 кр
 3: к 1 кр
 4: р 2 рк
 5: р 1 рк
 6: $ 0 $рк
 7: а 0 а$рк
 8: а 1 а$рк
 9: а 1 а$рк
10: а 1 а$рк
```

Далее получившийся набор индексов можно закодировать с помощью кода Хаффмана.