Лекция 6: Кодирование целых чисел

А. М. Шур

Кафедра алгебры и фундаментальной информатики УрФУ

5 апреля 2020 г.

О числовых данных

В данной лекции обсуждается сжатие “количественных” данных

- “Сырой” аудиофайл представляет собой лог датчика микрофона, измеряющего звуковое давление с высокой частотой (например, 44100 раз в секунду)
- Обычная 24-битная картинка формата bmp хранится как последовательность троек байтов
    - байты в тройке означают числовые характеристики пикселя — интенсивности красного, зеленого и синего компонентов цвета

О числовых данных

В данной лекции обсуждается сжатие “количественных” данных

- “Сырой” аудиофайл представляет собой лог датчика микрофона, измеряющего звуковое давление с высокой частотой (например, 44100 раз в секунду)
- Обычная 24-битная картинка формата bmp хранится как последовательность троек байтов
  - байты в тройке означают числовые характеристики пикселя — интенсивности красного, зеленого и синего компонентов цвета

И для картинок, и для аудио сжатие без потерь достаточно востребовано

★ Картинка двумерна, и ее лучше сжимают алгоритмы, использующие эту двумерность (например, lossless JPEG)

★ Аудиофайл одномерен, и приходится просто сжимать последовательность чисел, используя ее особенности (что и делает, например, FLAC)

О числовых данных

В данной лекции обсуждается сжатие “количественных” данных

- “Сырой” аудиофайл представляет собой лог датчика микрофона, измеряющего звуковое давление с высокой частотой (например, 44100 раз в секунду)
- Обычная 24-битная картинка формата bmp хранится как последовательность троек байтов
  - байты в тройке означают числовые характеристики пикселя — интенсивности красного, зеленого и синего компонентов цвета

И для картинок, и для аудио сжатие без потерь достаточно востребовано

★ Картинка двумерна, и ее лучше сжимают алгоритмы, использующие эту двумерность (например, lossless JPEG)

★ Аудиофайл одномерен, и приходится просто сжимать последовательность чисел, используя ее особенности (что и делает, например, FLAC)

Кроме того,

- Нужно хранить логи датчиков температуры, давления и других измеримых величин для различных устройств
- Нужно хранить логи биржевых котировок, биржевых индексов и др.
- Нужно хранить числа, генерируемые внутри алгоритмов сжатия на стадии моделирования (см. лекции о словарных методах сжатия)

Равномерное кодирование чисел

- Обычно для числовых данных используется равномерное кодирование (т.е. коды фиксированной длины)
    - например, каждое целое число в диапазоне $[-2^{31}..2^{31}-1]$ получает 32-битный код, в котором левый бит кодирует знак, а остальные представляют собой двоичную запись модуля числа, дополненную до 31 бита ведущими нулями

- Обычно для числовых данных используется равномерное кодирование (т.е. коды фиксированной длины)
  - например, каждое целое число в диапазоне $[-2^{31}..2^{31}-1]$ получает 32-битный код, в котором левый бит кодирует знак, а остальные представляют собой двоичную запись модуля числа, дополненную до 31 бита ведущими нулями
- Числа с плавающей точкой физически хранятся как пары целых чисел, так что рассмотрение ниже только целых чисел не ограничивает общности
  - например, в 64-битном стандарте IEEE 754 старший бит отводится под знак числа, следующие 11 бит — под экспоненту, а оставшиеся 52 — под мантиссу; хранимое число выглядит как $(\text{знак})\,1.(\text{мантисса}) \cdot 2^{\text{экспонента}-1023}$

★ Коды фиксированной длины важны для быстрого выполнения арифметических операций, но не экономичны с точки зрения хранения данных

Равномерное кодирование чисел

- Обычно для числовых данных используется равномерное кодирование (т.е. коды фиксированной длины)
  - например, каждое целое число в диапазоне $[-2^{31}..2^{31}-1]$ получает 32-битный код, в котором левый бит кодирует знак, а остальные представляют собой двоичную запись модуля числа, дополненную до 31 бита ведущими нулями
- Числа с плавающей точкой физически хранятся как пары целых чисел, так что рассмотрение ниже только целых чисел не ограничивает общности
  - например, в 64-битном стандарте IEEE 754 старший бит отводится под знак числа, следующие 11 бит — под экспоненту, а оставшиеся 52 — под мантиссу; хранимое число выглядит как $(\text{знак})\,1.(\text{мантисса})\cdot 2^{\text{экспонента}-1023}$

★ Коды фиксированной длины важны для быстрого выполнения арифметических операций, но не экономичны с точки зрения хранения данных

За счет чего можно хранить числа компактнее:

- Диапазон реальных значений измеряемой величины может быть много меньше диапазона типа данных
- Соседние значения могут быть сильно зависимы (например, первая разность измеряемой величины может быть почти всюду мала)
- Могут присутствовать дополнительные зависимости (например, периодические, как в аудиофайлах или в датчиках наружной температуры)

Кодирование с переполнением

Рассмотрим массив чисел, в котором основная часть элементов находится в диапазоне примерно 0–200, при этом отдельные значения могут достигать 100000. Будем использовать коды фиксированной длины с переполнением.

Схема кодирования:

- числа $X \in [0..254]$: 8 бит, двоичное представление X, дополненное ведущими нулями
- числа $X \in [255..65789]$: 24 бита, 11111111 + двоичное представление числа $X - 255$, дополненное до 16 бит ведущими нулями
- числа $X \geqslant 65790$: 40 бит, 24 единицы + двоичное представление числа $X - 65790$, дополненное до 16 бит ведущими нулями

Декодирование:

- читаем байт B (пусть его номер k), если $B \neq 255$, то это очередное число
- если $B = 255$, добавляем к 255 число, записанное $k+1$ и $k+2$ байтами, а если получили 65790 – то еще и число, записанное $k+3$ и $k+4$ байтами

Кодирование с переполнением

Рассмотрим массив чисел, в котором основная часть элементов находится в диапазоне примерно 0–200, при этом отдельные значения могут достигать 100000. Будем использовать коды фиксированной длины с переполнением.

Схема кодирования:

- числа $X \in [0..254]$: 8 бит, двоичное представление X, дополненное ведущими нулями
- числа $X \in [255..65789]$: 24 бита, 11111111 + двоичное представление числа $X - 255$, дополненное до 16 бит ведущими нулями
- числа $X \geqslant 65790$: 40 бит, 24 единицы + двоичное представление числа $X - 65790$, дополненное до 16 бит ведущими нулями

Декодирование:

- читаем байт B (пусть его номер k), если $B \neq 255$, то это очередное число
- если $B = 255$, добавляем к 255 число, записанное $k+1$ и $k+2$ байтами, а если получили 65790 – то еще и число, записанное $k+3$ и $k+4$ байтами

Пусть в файле 80% чисел меньше 255 и только 2% больше 65789. Тогда число в среднем кодируется $0.8 \cdot 8 + 0.18 \cdot 24 + 0.02 \cdot 40 \approx 11.5$ битами вместо $17 = \lfloor \log 100000 \rfloor + 1$ бит, требуемых на обычные коды фиксированной длины.

★ Полученную при кодировании последовательность байт можно дополнительно сжать методом Хаффмана или арифметическим методом

Коды переменной длины

Для декодирования нужен префиксный код (Лекция 3)

Особенность: очень большой алфавит (типичный случай: в списке длины N встречается примерно $N/2$ различных чисел, большинство по одному разу)

Коды переменной длины

Для декодирования нужен префиксный код (Лекция 3)

Особенность: очень большой алфавит (типичный случай: в списке длины N встречается примерно $N/2$ различных чисел, большинство по одному разу)

- При кодировании по Хаффману (Лекция 4) нужно выписать в явном виде исходные коды всех символов
  - в статическом варианте — при передаче дерева
  - в динамическом — при кодировании символов `new`

...это катастрофически ухудшает сжатие

Коды переменной длины

Для декодирования нужен префиксный код (Лекция 3)

Особенность: очень большой алфавит (типичный случай: в списке длины N встречается примерно $N/2$ различных чисел, большинство по одному разу)

- При кодировании по Хаффману (Лекция 4) нужно выписать в явном виде исходные коды всех символов
  - в статическом варианте — при передаче дерева
  - в динамическом — при кодировании символов `new`

...это катастрофически ухудшает сжатие

- При арифметическом кодировании (Лекция 5) тоже не избежать явного выписывания алфавита:
  - в статическом варианте нужна готовая таблица частот
  - в динамическом варианте нужен динамический алфавит (представьте, что будет с энтропией, если инициализировать единицами счетчики для 2^{32} символов-чисел)

Нужны способы кодирования, заточенные под большие переменные алфавиты

Peter Elias, “Universal codeword sets and representations of the integers”, IEEE Transactions in Information Theory, 1975.

Питер Элиас и его любимый PDP-1
в Массачусетском технологическом в 1960-е

Peter Elias, “Universal codeword sets and representations of the integers”, IEEE Transactions in Information Theory, 1975.

Питер Элиас и его любимый PDP-1
в Массачусетском технологическом в 1960-е

Элиас предложил несколько вариантов префиксных кодов, “заточенных” под кодирование целых чисел

Гамма-код Элиаса

Рассмотрим кодирование натуральных чисел. Разобьем их на диапазоны

- i-й диапазон равен $[2^i..2^{i+1}-1]$ ($i \geqslant 0$), т.е. состоит из 2^i чисел, в двоичной записи которых $i+1$ цифра
- Число N из i-го диапазона кодируется $2i+1$ битом: i нулей, затем 1, затем двоичная запись числа $N-2^i$ ("сдвиг" относительно нижней границы диапазона), дополненная до i бит ведущими нулями:

№	Диапазон	Код	Длина кода
0	1	1	1
1	2-3	01X	3
2	4-7	001XX	5
3	8-15	0001XXX	7
4	16-31	00001XXXX	9
5	32-63	000001XXXXX	11
...	...	...	...

Гамма-код Элиаса

Рассмотрим кодирование натуральных чисел. Разобьем их на диапазоны

- i-й диапазон равен $[2^i..2^{i+1}-1]$ ($i \geqslant 0$), т.е. состоит из 2^i чисел, в двоичной записи которых $i+1$ цифра
- Число N из i-го диапазона кодируется $2i+1$ битом: i нулей, затем 1, затем двоичная запись числа $N - 2^i$ (“сдвиг” относительно нижней границы диапазона), дополненная до i бит ведущими нулями:

№	Диапазон	Код	Длина кода
0	1	1	1
1	2-3	01X	3
2	4-7	001XX	5
3	8-15	0001XXX	7
4	16-31	00001XXXX	9
5	32-63	000001XXXXX	11
...	...	...	...

Декодирование:

- считаем нули до первой единицы, их число (i) — это номер диапазона
- следующие $i+1$ бит — это и есть декодируемое число
- декодирование следующего числа начинается с первого непрочитанного бита

Гамма-код Элиаса

Рассмотрим кодирование натуральных чисел. Разобьем их на диапазоны

- i-й диапазон равен $[2^i..2^{i+1}-1]$ ($i \geqslant 0$), т.е. состоит из 2^i чисел, в двоичной записи которых $i+1$ цифра
- Число N из i-го диапазона кодируется $2i+1$ битом: i нулей, затем 1, затем двоичная запись числа $N-2^i$ ("сдвиг" относительно нижней границы диапазона), дополненная до i бит ведущими нулями:

№	Диапазон	Код	Длина кода
0	1	1	1
1	2-3	01X	3
2	4-7	001XX	5
3	8-15	0001XXX	7
4	16-31	00001XXXX	9
5	32-63	000001XXXXX	11
...	...	...	...

Декодирование:

- считаем нули до первой единицы, их число (i) — это номер диапазона
- следующие $i+1$ бит — это и есть декодируемое число
- декодирование следующего числа начинается с первого непрочитанного бита

Пример: 00000101010001...

Гамма-код Элиаса

Рассмотрим кодирование натуральных чисел. Разобьем их на диапазоны

- i-й диапазон равен $[2^i..2^{i+1}-1]$ ($i \geqslant 0$), т.е. состоит из 2^i чисел, в двоичной записи которых $i+1$ цифра
- Число N из i-го диапазона кодируется $2i+1$ битом: i нулей, затем 1, затем двоичная запись числа $N-2^i$ (“сдвиг” относительно нижней границы диапазона), дополненная до i бит ведущими нулями:

№	Диапазон	Код	Длина кода
0	1	1	1
1	2-3	01X	3
2	4-7	001XX	5
3	8-15	0001XXX	7
4	16-31	00001XXXX	9
5	32-63	000001XXXXX	11
...	...	...	...

Декодирование:

- считаем нули до первой единицы, их число (i) — это номер диапазона
- следующие $i+1$ бит — это и есть декодируемое число
- декодирование следующего числа начинается с первого непрочитанного бита

Пример: 00000101010001... $\to i = 5$

Рассмотрим кодирование натуральных чисел. Разобьем их на диапазоны

- i-й диапазон равен $[2^i..2^{i+1}-1]$ $(i \geqslant 0)$, т.е. состоит из 2^i чисел, в двоичной записи которых $i+1$ цифра
- Число N из i-го диапазона кодируется $2i+1$ битом: i нулей, затем 1, затем двоичная запись числа $N-2^i$ ("сдвиг" относительно нижней границы диапазона), дополненная до i бит ведущими нулями:

№	Диапазон	Код	Длина кода
0	1	1	1
1	2-3	01X	3
2	4-7	001XX	5
3	8-15	0001XXX	7
4	16-31	00001XXXX	9
5	32-63	000001XXXXX	11
...	...	...	...

Декодирование:

- считаем нули до первой единицы, их число (i) — это номер диапазона
- следующие $i+1$ бит — это и есть декодируемое число
- декодирование следующего числа начинается с первого непрочитанного бита

Пример: 00000101010001... $\to N = [101010]_2 = 42$

Гамма-код Элиаса (2)

- Требует $2\lfloor \log N \rfloor + 1$ бит для записи числа N против $\lfloor \log N \rfloor + 1$ в обычной двоичной записи
  - это плата за запись набора чисел "без запятых"

Гамма-код Элиаса (2)

- Требует $2\lfloor \log N \rfloor + 1$ бит для записи числа N против $\lfloor \log N \rfloor + 1$ в обычной двоичной записи
  - это плата за запись набора чисел “без запятых”
- Если нужно кодировать неотрицательные числа, кодируем $N + 1$ вместо N

Гамма-код Элиаса (2)

- Требует $2\lfloor\log N\rfloor + 1$ бит для записи числа N против $\lfloor\log N\rfloor + 1$ в обычной двоичной записи
  - это плата за запись набора чисел “без запятых”
- Если нужно кодировать неотрицательные числа, кодируем $N+1$ вместо N

Если нужно кодировать все целые числа, выстроим их в последовательность

$$0, 1, -1, 2, -2, 3, -3, 4, \ldots$$

и для каждого числа будем кодировать его номер в этой последовательности.

- Диапазоны симметричны относительно 0: $0,\ \{\pm 1\},\ \{\pm 2, \pm 3\}, \ldots$
- Натуральное N кодируется как $2N$, а отрицательное $-N$ — как $2N+1$
- Декодирование не усложняется:
  - посчитали диапазон (i)
  - следующие i бит — модуль закодированного числа, а (i+1)-й бит — его знак

Гамма-код Элиаса (2)

- Требует $2\lfloor \log N \rfloor + 1$ бит для записи числа N против $\lfloor \log N \rfloor + 1$ в обычной двоичной записи
  - это плата за запись набора чисел “без запятых”
- Если нужно кодировать неотрицательные числа, кодируем $N+1$ вместо N

Если нужно кодировать все целые числа, выстроим их в последовательность

$$0, 1, -1, 2, -2, 3, -3, 4, \ldots$$

и для каждого числа будем кодировать его номер в этой последовательности.

- Диапазоны симметричны относительно 0: $0,\ \{\pm 1\},\ \{\pm 2, \pm 3\}, \ldots$
- Натуральное N кодируется как $2N$, а отрицательное $-N$ — как $2N+1$
- Декодирование не усложняется:
  - посчитали диапазон (i)
  - следующие i бит — модуль закодированного числа, а $(i{+}1)$-й бит — его знак

Пример: 00000101010001...

Гамма-код Элиаса (2)

- Требует $2\lfloor \log N \rfloor + 1$ бит для записи числа N против $\lfloor \log N \rfloor + 1$ в обычной двоичной записи
  - это плата за запись набора чисел “без запятых”
- Если нужно кодировать неотрицательные числа, кодируем $N+1$ вместо N

Если нужно кодировать все целые числа, выстроим их в последовательность

$$0, 1, -1, 2, -2, 3, -3, 4, \dots$$

и для каждого числа будем кодировать его номер в этой последовательности.

- Диапазоны симметричны относительно 0: $0,\ \{\pm 1\},\ \{\pm 2, \pm 3\}, \dots$
- Натуральное N кодируется как $2N$, а отрицательное $-N$ — как $2N+1$
- Декодирование не усложняется:
  - посчитали диапазон (i)
  - следующие i бит — модуль закодированного числа, а (i+1)-й бит — его знак

Пример: 00000101010001... $\to i = 5$

- Требует $2\lfloor \log N \rfloor + 1$ бит для записи числа N против $\lfloor \log N \rfloor + 1$ в обычной двоичной записи
  - это плата за запись набора чисел "без запятых"
- Если нужно кодировать неотрицательные числа, кодируем $N+1$ вместо N

Если нужно кодировать все целые числа, выстроим их в последовательность

$$0, 1, -1, 2, -2, 3, -3, 4, \ldots$$

и для каждого числа будем кодировать его номер в этой последовательности.

- Диапазоны симметричны относительно 0: $0,\ \{\pm 1\},\ \{\pm 2, \pm 3\}, \ldots$
- Натуральное N кодируется как $2N$, а отрицательное $-N$ — как $2N+1$
- Декодирование не усложняется:
  - посчитали диапазон (i)
  - следующие i бит — модуль закодированного числа, а $(i{+}1)$-й бит — его знак

Пример: 0000010101 0001... $\to |N| = [10101]_2 = 21$

Гамма-код Элиаса (2)

- Требует $2\lfloor \log N \rfloor + 1$ бит для записи числа N против $\lfloor \log N \rfloor + 1$ в обычной двоичной записи
  - это плата за запись набора чисел “без запятых”
- Если нужно кодировать неотрицательные числа, кодируем $N+1$ вместо N

Если нужно кодировать все целые числа, выстроим их в последовательность

$$0, 1, -1, 2, -2, 3, -3, 4, \dots$$

и для каждого числа будем кодировать его номер в этой последовательности.

- Диапазоны симметричны относительно 0: $0,\ \{\pm 1\},\ \{\pm 2, \pm 3\}, \dots$
- Натуральное N кодируется как $2N$, а отрицательное $-N$ — как $2N+1$
- Декодирование не усложняется:
  - посчитали диапазон (i)
  - следующие i бит — модуль закодированного числа, а (i+1)-й бит — его знак

Пример: 00000101010001... $\rightarrow N > 0 \rightarrow N = 21$

Гамма-код Элиаса (2)

- Требует $2\lfloor\log N\rfloor + 1$ бит для записи числа N против $\lfloor\log N\rfloor + 1$ в обычной двоичной записи
  - это плата за запись набора чисел “без запятых”
- Если нужно кодировать неотрицательные числа, кодируем $N+1$ вместо N

Если нужно кодировать все целые числа, выстроим их в последовательность

$$0, 1, -1, 2, -2, 3, -3, 4, \ldots$$

и для каждого числа будем кодировать его номер в этой последовательности.

- Диапазоны симметричны относительно 0: 0, $\{\pm 1\}$, $\{\pm 2, \pm 3\}, \ldots$
- Натуральное N кодируется как $2N$, а отрицательное $-N$ — как $2N+1$
- Декодирование не усложняется:
  - посчитали диапазон (i)
  - следующие i бит — модуль закодированного числа, а (i+1)-й бит — его знак

Пример: 00000101010001... $\to N > 0 \to N = 21$

Этот же способ кодирования произвольных целых чисел применим и к рассматриваемым далее аналогам гамма-кода

Дельта-код Элиаса

- Очевидный недостаток гамма-кода: диапазон кодируется в "единичной" системе счисления, что неэкономно
  - на кодирование i-го диапазона нужно $i+1$ бит

Дельта-код Элиаса

- Очевидный недостаток гамма-кода: диапазон кодируется в “единичной” системе счисления, что неэкономно
  - на кодирование i-го диапазона нужно i+1 бит
- Дельта-код исправляет это, кодируя i-й диапазон гамма-кодом числа i+1:
  - на кодирование i-го диапазона нужно $2\lfloor\log(i+1)\rfloor+1$ бит
  - на кодирование числа N нужно $\lfloor\log N\rfloor+2\lfloor\log(\lfloor\log N\rfloor+1)\rfloor+1$ бит
  - превышение длины дельта-кода над длиной двоичной записи N составляет $\approx 2\log\log N$, что асимптотически меньше, чем превышение $\approx\log N$ у гамма-кода

№	Диапазон	Гамма-код	Длина γ-кода	Дельта-код	Длина δ-кода
0	1	1	1	1	1
1	2-3	01X	3	010X	4
2	4-7	001XX	5	011XX	5
3	8-15	0001XXX	7	00100XXX	8
4	16-31	00001XXXX	9	00101XXXX	9
5	32-63	000001XXXXX	11	00110XXXXX	10
...	...	...	...	...	...

Дельта-код Элиаса

- Очевидный недостаток гамма-кода: диапазон кодируется в “единичной” системе счисления, что неэкономно
  - на кодирование i-го диапазона нужно $i+1$ бит
- Дельта-код исправляет это, кодируя i-й диапазон гамма-кодом числа $i+1$:
  - на кодирование i-го диапазона нужно $2\lfloor\log(i+1)\rfloor+1$ бит
  - на кодирование числа N нужно $\lfloor\log N\rfloor+2\lfloor\log(\lfloor\log N\rfloor+1)\rfloor+1$ бит
  - превышение длины дельта-кода над длиной двоичной записи N составляет $\approx 2\log\log N$, что асимптотически меньше, чем превышение $\approx \log N$ у гамма-кода

№	Диапазон	Гамма-код	Длина γ-кода	Дельта-код	Длина δ-кода
0	1	1	1	1	1
1	2-3	01X	3	010X	4
2	4-7	001XX	5	011XX	5
3	8-15	0001XXX	7	00100XXX	8
4	16-31	00001XXXX	9	00101XXXX	9
5	32-63	000001XXXXX	11	00110XXXXX	10
...	...	...	...	...	...

Декодирование:

- считаем нули до первой единицы, их число (j) — номер диапазона числа $i+1$
- следующие $j+1$ бит — число $i+1$, где i — номер диапазона числа N
- декодируемое число N получается дописыванием слева 1 к следующим i битам

★ Очевидный недостаток гамма-кода: диапазон кодируется в “единичной” системе счисления, что неэкономно
  - на кодирование i-го диапазона нужно $i{+}1$ бит
- Дельта-код исправляет это, кодируя i-й диапазон гамма-кодом числа $i{+}1$:
  - на кодирование i-го диапазона нужно $2\lfloor\log(i+1)\rfloor+1$ бит
  - на кодирование числа N нужно $\lfloor\log N\rfloor + 2\lfloor\log(\lfloor\log N\rfloor+1)\rfloor+1$ бит
  - превышение длины дельта-кода над длиной двоичной записи N составляет $\approx 2\log\log N$, что асимптотически меньше, чем превышение $\approx \log N$ у гамма-кода

№	Диапазон	Гамма-код	Длина γ-кода	Дельта-код	Длина δ-кода
0	1	1	1	1	1
1	2-3	01X	3	010X	4
2	4-7	001XX	5	011XX	5
3	8-15	0001XXX	7	00100XXX	8
4	16-31	00001XXXX	9	00101XXXX	9
5	32-63	000001XXXXX	11	00110XXXXX	10
...	...	...	...	...	...

Декодирование:
- считаем нули до первой единицы, их число (j) — номер диапазона числа $i{+}1$
- следующие $j{+}1$ бит — число $i{+}1$, где i — номер диапазона числа N
- декодируемое число N получается дописыванием слева 1 к следующим i битам

Пример: 00111100011000...

★ Очевидный недостаток гамма-кода: диапазон кодируется в “единичной” системе счисления, что неэкономно
  - на кодирование i-го диапазона нужно $i{+}1$ бит
- Дельта-код исправляет это, кодируя i-й диапазон гамма-кодом числа $i{+}1$:
  - на кодирование i-го диапазона нужно $2\lfloor\log(i+1)\rfloor+1$ бит
  - на кодирование числа N нужно $\lfloor\log N\rfloor+2\lfloor\log(\lfloor\log N\rfloor+1)\rfloor+1$ бит
  - превышение длины дельта-кода над длиной двоичной записи N составляет $\approx 2\log\log N$, что асимптотически меньше, чем превышение $\approx \log N$ у гамма-кода

№	Диапазон	Гамма-код	Длина γ-кода	Дельта-код	Длина δ-кода
0	1	1	1	1	1
1	2-3	01X	3	010X	4
2	4-7	001XX	5	011XX	5
3	8-15	0001XXX	7	00100XXX	8
4	16-31	00001XXXX	9	00101XXXX	9
5	32-63	000001XXXXX	11	00110XXXXX	10
...	...	...	...	...	...

Декодирование:

- считаем нули до первой единицы, их число (j) — номер диапазона числа $i{+}1$
- следующие $j{+}1$ бит — число $i{+}1$, где i — номер диапазона числа N
- декодируемое число N получается дописыванием слева 1 к следующим i битам

Пример: 00111100011000... $\to j=2$

Дельта-код Элиаса

- Очевидный недостаток гамма-кода: диапазон кодируется в “единичной” системе счисления, что неэкономно
  - на кодирование i-го диапазона нужно $i{+}1$ бит
- Дельта-код исправляет это, кодируя i-й диапазон гамма-кодом числа $i{+}1$:
  - на кодирование i-го диапазона нужно $2\lfloor \log(i+1) \rfloor + 1$ бит
  - на кодирование числа N нужно $\lfloor \log N \rfloor + 2\lfloor \log(\lfloor \log N \rfloor + 1) \rfloor + 1$ бит
  - превышение длины дельта-кода над длиной двоичной записи N составляет $\approx 2 \log \log N$, что асимптотически меньше, чем превышение $\approx \log N$ у гамма-кода

№	Диапазон	Гамма-код	Длина γ-кода	Дельта-код	Длина δ-кода
0	1	1	1	1	1
1	2-3	01X	3	010X	4
2	4-7	001XX	5	011XX	5
3	8-15	0001XXX	7	00100XXX	8
4	16-31	00001XXXX	9	00101XXXX	9
5	32-63	000001XXXXX	11	00110XXXXX	10
...	...	...	...	...	...

Декодирование:

- считаем нули до первой единицы, их число (j) — номер диапазона числа $i{+}1$
- следующие $j{+}1$ бит — число $i{+}1$, где i — номер диапазона числа N
- декодируемое число N получается дописыванием слева 1 к следующим i битам

Пример: 0011110001100... $\to i = [111]_2 - 1 = 6$

★ Очевидный недостаток гамма-кода: диапазон кодируется в “единичной” системе счисления, что неэкономно

- на кодирование i-го диапазона нужно i+1 бит

- Дельта-код исправляет это, кодируя i-й диапазон гамма-кодом числа i+1:
  - на кодирование i-го диапазона нужно $2\lfloor\log(i+1)\rfloor+1$ бит
  - на кодирование числа N нужно $\lfloor\log N\rfloor + 2\lfloor\log(\lfloor\log N\rfloor+1)\rfloor+1$ бит
  - превышение длины дельта-кода над длиной двоичной записи N составляет $\approx 2\log\log N$, что асимптотически меньше, чем превышение $\approx \log N$ у гамма-кода

№	Диапазон	Гамма-код	Длина γ-кода	Дельта-код	Длина δ-кода
0	1	1	1	1	1
1	2-3	01X	3	010X	4
2	4-7	001XX	5	011XX	5
3	8-15	0001XXX	7	00100XXX	8
4	16-31	00001XXXX	9	00101XXXX	9
5	32-63	000001XXXXX	11	00110XXXXX	10
...	...	...	...	...	...

Декодирование:

- считаем нули до первой единицы, их число (j) — номер диапазона числа i+1
- следующие j+1 бит — число i+1, где i — номер диапазона числа N
- декодируемое число N получается дописыванием слева 1 к следующим i битам

Пример: 00111100011000... $\to N = [1100011]_2 = 99$

Дельта-код Элиаса

★ Очевидный недостаток гамма-кода: диапазон кодируется в "единичной" системе счисления, что неэкономно

- на кодирование i-го диапазона нужно i+1 бит

- Дельта-код исправляет это, кодируя i-й диапазон гамма-кодом числа i+1:
    - на кодирование i-го диапазона нужно $2\lfloor\log(i+1)\rfloor+1$ бит
    - на кодирование числа N нужно $\lfloor\log N\rfloor + 2\lfloor\log(\lfloor\log N\rfloor+1)\rfloor+1$ бит
    - превышение длины дельта-кода над длиной двоичной записи N составляет $\approx 2\log\log N$, что асимптотически меньше, чем превышение $\approx \log N$ у гамма-кода

№	Диапазон	Гамма-код	Длина γ-кода	Дельта-код	Длина δ-кода
0	1	1	1	1	1
1	2-3	01X	3	010X	4
2	4-7	001XX	5	011XX	5
3	8-15	0001XXX	7	00100XXX	8
4	16-31	00001XXXX	9	00101XXXX	9
5	32-63	000001XXXXX	11	00110XXXXX	10
...	...	...	...	...	...

Декодирование:

- считаем нули до первой единицы, их число (j) — номер диапазона числа i+1
- следующие j+1 бит — число i+1, где i — номер диапазона числа N
- декодируемое число N получается дописыванием слева 1 к следующим i битам

Пример: 00111100011000... $\to N = [1100011]_2 = 99$

- Гамма-код короче при $2 \leqslant N \leqslant 3$ и $8 \leqslant N \leqslant 15$
- Дельта-код короче при $N \geqslant 32$; можно ли еще оптимизировать?

Другие способы улучшения гамма-кода

- Можно итерировать идею дельта-кода: закодировать гамма-кодом не только номер диапазона кодируемого числа, но и номер диапазона номера диапазона, и т.д.
  - код, построенный на этой идее, называется омега-кодом Элиаса
  - омега-код имеет небольшое асимптотическое преимущество перед дельта-кодом, но в большинстве диапазонов с малыми номерами дельта-коды экономичнее

Другие способы улучшения гамма-кода

- Можно итерировать идею дельта-кода: закодировать гамма-кодом не только номер диапазона кодируемого числа, но и номер диапазона номера диапазона, и т.д.
  - код, построенный на этой идее, называется омега-кодом Элиаса
  - омега-код имеет небольшое асимптотическое преимущество перед дельта-кодом, но в большинстве диапазонов с малыми номерами дельта-коды экономичнее
- ★ Более продуктивная идея — посчитать частоту встречаемости диапазонов и кодировать номера диапазонов кодами Хаффмана
  - ⋆ фиксированный дополнительный расход памяти в $\approx \log N \cdot \log\log N$ бит на явное выписывание кодов диапазонов, где N — максимальное кодируемое число
  - ⋆ пропорциональная длине кодируемого списка чисел экономия на длине записи экспонент за счет оптимальности кода Хаффмана

Предложение

Любое натуральное число представимо единственным образом в виде суммы различных чисел Фибоначчи, никакие два из которых не являются соседними.

Доказательство: для представления числа N возьмем наибольшее число Фибоначчи Φ_k, не превосходящее N, и рекурсивно построим представление для $N - \Phi_k$. Так как $N < \Phi_{k+1}$ влечет $N - \Phi_k < \Phi_{k-1}$, в полученном представлении не будет соседних чисел Фибоначчи. Представление единственно, так как по индукции легко показать, что для любого k $\Phi_{k+1} > \Phi_k + \Phi_{k-2} + \Phi_{k-4} + \cdots$. □

Предложение

Любое натуральное число представимо единственным образом в виде суммы различных чисел Фибоначчи, никакие два из которых не являются соседними.

Доказательство: для представления числа N возьмем наибольшее число Фибоначчи Φ_k, не превосходящее N, и рекурсивно построим представление для $N - \Phi_k$. Так как $N < \Phi_{k+1}$ влечет $N - \Phi_k < \Phi_{k-1}$, в полученном представлении не будет соседних чисел Фибоначчи. Представление единственно, так как по индукции легко показать, что для любого k $\Phi_{k+1} > \Phi_k + \Phi_{k-2} + \Phi_{k-4} + \cdots$. □

- Любое натуральное N можно записать двоичной позиционной записью в системе счисления Фибоначчи:
  - k-й бит равен единице $\Leftrightarrow$ Φ_k присутствует в представлении N
  - старший бит, всегда равный 1, записывается справа

  Пример: $1024 = 987 + 34 + 3 = \Phi_{15} + \Phi_8 + \Phi_3 = [001000010000001]_{fib}$

Предложение

Любое натуральное число представимо единственным образом в виде суммы различных чисел Фибоначчи, никакие два из которых не являются соседними.

Доказательство: для представления числа N возьмем наибольшее число Фибоначчи Φ_k, не превосходящее N, и рекурсивно построим представление для $N - \Phi_k$. Так как $N < \Phi_{k+1}$ влечет $N - \Phi_k < \Phi_{k-1}$, в полученном представлении не будет соседних чисел Фибоначчи. Представление единственно, так как по индукции легко показать, что для любого k $\Phi_{k+1} > \Phi_k + \Phi_{k-2} + \Phi_{k-4} + \cdots$. □

- Любое натуральное N можно записать двоичной позиционной записью в системе счисления Фибоначчи:
  - k-й бит равен единице $\Leftrightarrow$ Φ_k присутствует в представлении N
  - старший бит, всегда равный 1, записывается справа

  Пример: $1024 = 987 + 34 + 3 = \Phi_{15} + \Phi_8 + \Phi_3 = [001000010000001]_{fib}$

★ Фибоначчиева запись никогда не содержит двух единиц подряд и заканчивается на 1 $\Rightarrow$ если после каждого числа дописывать бит 1, то получится подстрока 11, которая будет “запятой”, отделяющей число от следующего

Длина кода Фибоначчи числа N равна $1 + \lfloor \log_\phi N + c \rfloor$, где $\phi = \frac{1+\sqrt{5}}{2}$ — золотое сечение, а $c \approx 2/3$

Превышение над длиной двоичной записи N примерно $0.44 \log N + 2/3$

Альтернатива: троичное кодирование

Представим число N в троичной системе:

$$N = [i_k i_{k-1} \ldots i_1 i_0]_3 = i_k \cdot 3^k + i_{k-1} \cdot 3^{k-1} + \cdots + i_1 \cdot 3^1 + i_0, \;\; i_k \neq 0$$

Альтернатива: троичное кодирование

Представим число N в троичной системе:

$$N = [i_k i_{k-1} \dots i_1 i_0]_3 = i_k \cdot 3^k + i_{k-1} \cdot 3^{k-1} + \dots + i_1 \cdot 3^1 + i_0, \;\; i_k \neq 0$$

Закодируем N последовательностью из $2k+1$ бит:

- i_k кодируется одним битом, равным $i_k - 1$
- каждое из чисел $i_{k-1}, \dots, i_0$ кодируется своим бинарным кодом из двух бит
- после i_0 записывается "запятая" 11 для разделения чисел

Пример: $42 = 3^3 + 3^2 + 2 \cdot 3^1$ кодируется строкой 0 01 10 00 11

Длина троичного кода числа N равна $3 + 2\lfloor \log_3 N \rfloor$
Превышение над длиной двоичной записи N примерно $0.26 \log N + 2$

Представим число N в троичной системе:

$$N = [i_k i_{k-1} \dots i_1 i_0]_3 = i_k \cdot 3^k + i_{k-1} \cdot 3^{k-1} + \cdots + i_1 \cdot 3^1 + i_0, \;\; i_k \neq 0$$

Закодируем N последовательностью из $2k+1$ бит:

- i_k кодируется одним битом, равным $i_k - 1$
- каждое из чисел $i_{k-1}, \dots, i_0$ кодируется своим бинарным кодом из двух бит
- после i_0 записывается "запятая" 11 для разделения чисел

Пример: $42 = 3^3 + 3^2 + 2 \cdot 3^1$ кодируется строкой 0 01 10 00 11

Длина троичного кода числа N равна $3 + 2\lfloor \log_3 N \rfloor$
Превышение над длиной двоичной записи N примерно $0.26 \log N + 2$

★ оба альтернативных кодирования достаточно эффективны (см. следующую страницу), но работают медленнее кодирования Элиаса, поскольку построение фибоначчиева или троичного разложения числа N из машинного двоичного представления требует выполнения порядка $\log N$ арифметических операций

Сравнение длин кодов: табличка для медитации

Диапазон	γ	δ	Ф	З
1	1	1	2	3
2	3	4	3	3
3	3	3	4	5
4	5	5	4	5
5-7	5	5	5	5
8	7	8	6	5
9-12	7	8	6	7
13-15	7	8	7	7
16-20	9	9	7	7
21-26	9	9	8	7
27-31	9	9	8	9
32-33	11	10	8	9
34-54	11	10	9	9
55-63	11	10	10	9
64-80	13	11	10	9
81-88	13	11	10	11
89-127	13	11	11	11
128-133	15	14	11	11
134-222	15	14	12	11
223-242	15	14	13	11
243-255	15	14	13	13
256-376	17	15	13	13
377-511	17	15	14	13
512-609	19	16	14	13
610-728	19	16	15	13

Диапазон	γ	δ	Ф	З
729-986	19	16	15	15
987-1023	19	16	16	15
1024-1596	21	17	16	15
1597-2047	21	17	17	15
2048-2186	23	18	17	15
2187-2583	23	18	17	17
2584-4095	23	18	18	17
4096-4180	25	19	18	17
4181-6560	25	19	19	17
6561-6764	25	19	19	19
6765-8191	25	19	20	19
8192-10945	27	20	20	19
10946-16383	27	20	21	19
16384-17710	29	21	21	19
17711-19682	29	21	22	19
19683-28656	29	21	22	21
28657-32767	29	21	23	21
32768-46367	31	24	23	21
46368-59048	31	24	24	21
59049-65535	31	24	24	23
...	...	...	...	...
1000000	39	28	30	27
10000000	45	32	35	31
100000000	51	35	39	35
1000000000	57	38	44	39