ТЕХНОСФЕРА

Сжатие индекса и словаря

Сергукова Юлия,
программист отдела инфраструктуры проекта
Поиск@Mail.Ru

План лекции:

1. **Обратный индекс – пересказ прошлого занятия**
2. **Сжатие индекса**
3. **Сжатие чисел**
4. **Сжатие последовательностей**

ТЕХНОСФЕРА

Обратный индекс

ID

Term <-> TermID

документ <-> URL <-> DocID

Быстрый и компактный

1. Быстрый:
    1. Больше нагрузка – все запросы
    2. Пользователь не будет ждать!
2. Компактный:
    1. Завязано на скорость – можем хранить в RAM

\+

Гибкий:

- Хранить разные данные (зонные индексы)
- Масштабируемый / разделяемый

ТЕХНОСФЕРА

Память: как правильно с ней работать?

1. Меньше позиционируемся – больше читаем
2. Меньший объем данных – меньше читать

ТЕХНОСФЕРА

Зачем сжимать?

1. Очевидное: индекс в RAM
2. Ускоряем сам факт передачи данных
3. «Прочитать сжатое + распаковать» иногда быстрее, чем «прочитать несжатое»
4. Кэшируем бОльшие объемы данных

Виды сжатия

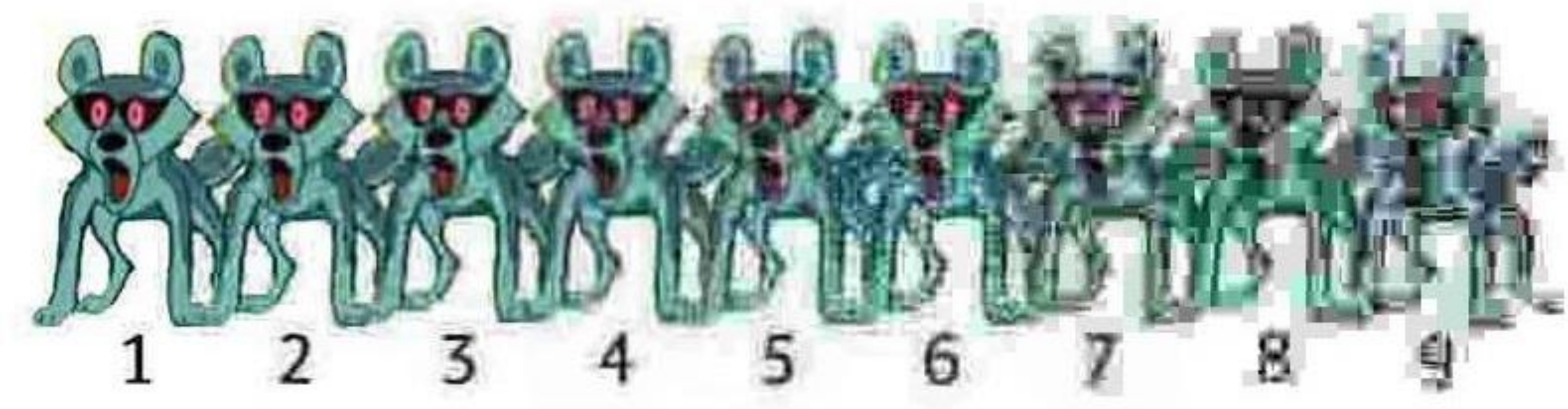
В ИП в основном сжатие без потерь (почему?)

1. gzip/rar/lzo
2. png
3. и т.д.

ТЕХНОСФЕРА

Виды сжатия

Сжатие с потерями



Виды сжатия

Сжатие с потерями:

1. Архиватор Попова ☺
2. В ИП: удаление капитализации, удаление стоп-слов, лемматизация
3. Координаты для дальних позиций слов

Что можно сжать?

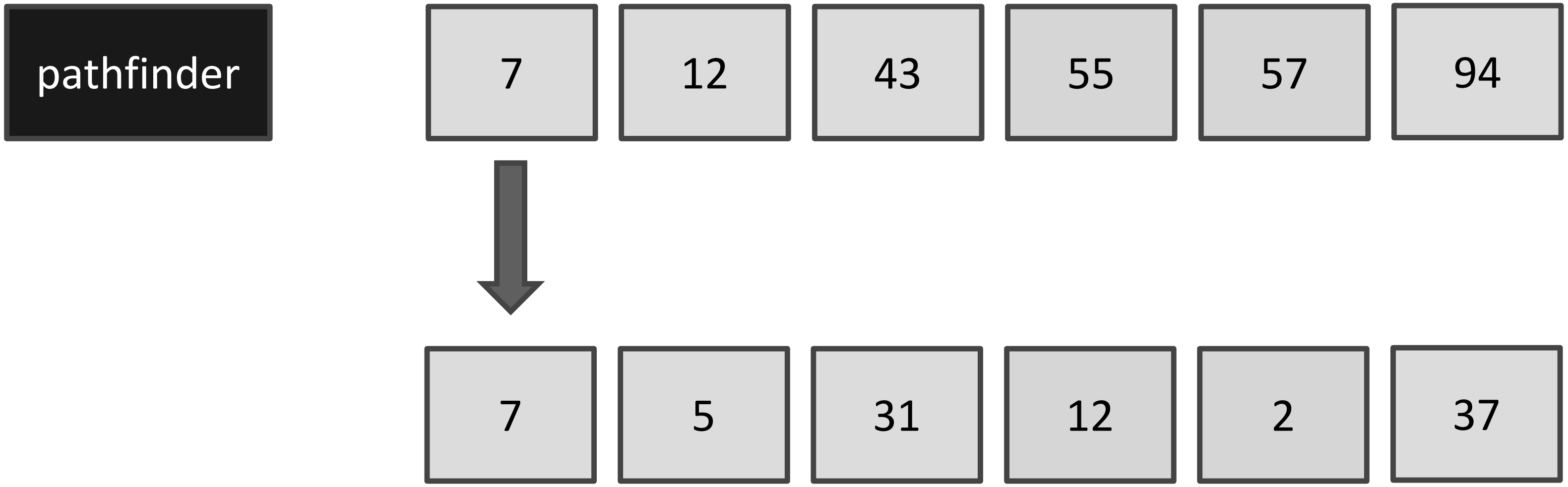
1. Постинг-листы
2. Словарь

Цель

- Индексируем 1M документов
  - docID – int – 32b
  - но 1M можно записать 20b
- Наша цель: использовать меньше или хотя бы 20b

Подготовка к компрессии

- Упорядочить docID по возрастанию
- Можем кодировать на docID, а delta



- важно помнить: delta > 0

Кодирование чисел

- Нет смысла использовать фиксированное количество бит для всех чисел – «лишние» ведущие нули
- Нужен код переменной длины

Побитовая гранулярность

- Как определять конец последовательности?
  - недопустимая последовательность
  - кодировать длину

Коды Элиаса

Разработал в 1960-х годах Питер Элиас.

гамма-, дельта-, омега- коды

все – для положительных целых чисел

Основной принцип: двоичное представление + кодируем нулями длину записи (т.н. универсальный код)

Elias-gamma

1. получаем двоичное представление числа: N → Nb
2. считаем количество значимых бит: bit_len(Nb) = B
3. результат: <B-1 нулей><Nb>

Elias-gamma

1. получаем двоичное представление числа: N → Nb
2. считаем количество значимых бит: bit_len(Nb) = B
3. результат: <B-1 нулей><Nb>

Пример:

N = 13

Elias-gamma

1. получаем двоичное представление числа: N → Nb
2. считаем количество значимых бит: bit_len(Nb) = B
3. результат: <B-1 нулей><Nb>

Пример:

N = 13

Nb = 1101 → B = 4

Eg(13) = 0001101

Elias-gamma

1. получаем двоичное представление числа: N → Nb
2. считаем количество значимых бит: bit_len(Nb) = B
3. результат: <B-1 нулей><Nb>

Пример:

N = 1

Nb = 1 → B = 1

Eg(1) = 1

Elias-gamma

000010101001111000010111

Elias-gamma

0000101010011111000010111

000010101   0011111000010111

000010101   00111   1000010111

000010101   00111   1   000010111

000010101   00111   1   000010111

Elias-gamma

+

- для небольших чисел используем меньше 32 бит

-

- для больших чисел – оверхэд на нулях

Elias-delta

Используется для заведомо больших чисел

1. получаем двоичное представление числа: N → Nb
2. считаем количество значимых бит: bit_len(Nb) = B
3. кодируем B с помощью Elias-gamma
4. результат: <Eg(B)><Nb>

Elias-delta

1. получаем двоичное представление числа: N → Nb
2. считаем количество значимых бит: bit_len(Nb) = B
3. кодируем B с помощью Elias-gamma
4. результат: <Eg(B)><Nb>

Пример:

N = 1057

Elias-delta

1. получаем двоичное представление числа: N → Nb
2. считаем количество значимых бит: bit_len(Nb) = B
3. кодируем B с помощью Elias-gamma
4. результат: <Eg(B)><Nb>

Пример:

N = 1057

Nb = 10000100001, B = 11

Eg(8) = 0001011

Ed(157) = 000101110000100001

Elias-delta

1. получаем двоичное представление числа: N → Nb
2. считаем количество значимых бит: bit_len(Nb) = B
3. кодируем B с помощью Elias-gamma
4. результат: <Eg(B)><Nb>

Пример:

Ed(157) = 000101110000100001

Eg(157) = 00000000010000100001

Коды Голомба

Энтропийные коды, изобретенные Соломоном Голомбом

Тезис: множество кодируемых чисел можно описать через геометрическое распределение, чьи характеристики можно использовать для кодирования последовательности

Код Голомба-Райса

Параметр кодирования k

1. частное $q = (N-1) / 2^k$
2. остаток от деления $r = N – q*2^k – 1$
3. q кодируем унарно, r – бинарно, используя k бит

Код Голомба-Райса

Параметр кодирования k

1. частное q = (N-1) / 2^k
2. остаток от деления r = N – q*2^k – 1
3. q кодируем унарно, r – бинарно, используя k бит

Пример:

GR(113, k=5) - ?

Код Голомба-Райса

Параметр кодирования k

1. частное q = (N-1) / 2^k – округляем вниз
2. остаток от деления r = N – q*2^k – 1
3. q кодируем унарно, 1 – разделитель, r кодируем бинарно, используя k бит

Пример:

GR(113, k=5) - ?

q = (113-1)/2^5 = 3.5 → 3 → 000

r = 113 – 3*2^5 – 1 = 16 → r2 = 10000

ТЕХНОСФЕРА

Код Голомба-Райса

Параметр кодирования k

1. частное q = (N-1) / 2^k – округляем вниз
2. остаток от деления r = N – q*2^k – 1
3. q кодируем унарно, 1 – разделитель, r кодируем бинарно, используя k бит

Пример:

GR(113, k=5) – 000110000

q = (113-1)/2^5 = 3.5 → 3 → 000

r = 113 – 3*2^5 – 1 = 16 → r2 = 10000

Код Голомба-Райса

Параметр кодирования k

1. частное q = $\lfloor {}^{N-1}/_{2^k} \rfloor$
2. остаток от деления r = $N - q * 2^k - 1$
3. q кодируем унарно, 1 – разделитель, r кодируем бинарно, используя k бит

Пример:

GR(113, k=4) – 00000001 0000

q = (113-1)/2^4 = 7 → 7 → 0000000

r = 113 – 7*2^4 – 1 = 0 → r2 = 0000 – используем k бит (!)

Код Голомба-Райса

+

- подходит для последовательностей с «ядром», вокруг которого сгруппированы значения

-

- сильно ухудшается во всех остальных случаях

???


A
LEONARDO FIBONACCI
INSIGNE MATEMATICO
PISANO
DEL
SECOLO XII

???

1 + 1 = 2
1 + 2 = 3
2 + 3 = 5
3 + 5 = 8
5 + 8 = 13
…

ТЕХНОСФЕРА

Код Фиббоначи

1	1	2	3	5	8	13

Код Фиббоначи

1	1	2	3	5	8	13

Используем числа Фиббоначи как значения разрядов в новой СИ

ТЕХНОСФЕРА

Код Фиббоначи

1	1	2	3	5	8	13

Используем числа Фиббоначи как значения разрядов в новой СИ:
… 21 13 8 5 3 2 1

1 = 1 → 1
2 = 2 → 1 0
4 = 3+1 → 1 0 1
19 = 13+5+1 → 1 0 1 0 0 1

Код Фиббоначи

Используем числа Фиббоначи как значения разрядов в новой СИ:
… 21 13 8 5 3 2 1

11 = 8 + 3 → 1 0 1 0 0
Как определить конец числа?

ТЕХНОСФЕРА

Код Фиббоначи

1 1 2 3 5 8 13

Используем числа Фиббоначи как значения разрядов в новой СИ:
… 21 13 8 5 3 2 1

11 = 8 + 3 → 1 0 1 0 0
Как определить конец числа?
001011

Код Фиббоначи

+

- выигрыш на небольших значениях

-

- лишние вычисления

Байтовая гранулярность

Проблема битовой гранулярности – нужно каждый раз вычислять величину записи следующего числа => нельзя перескакивать блоки чисел

Минимальная единица – байт

Variable Byte (VarByte)

1 байт = 8 бит = 1 маркерный бит + 7 значимых бит

Маркерный бит:

1 – конец числа

0 – «продолжение в следующем байте»

ТЕХНОСФЕРА

VarByte

- 3 → 11 → 10000011
- 2 → 10 → 10000010

VarByte

- 3 → 11 → 10000011
- 2 → 10 → 10000010
- 2018 → 11111100010

VarByte

- 3 → 11 → 10000011
- 2 → 10 → 10000010
- 2018 → 11111100010 → 1111 1100010

ТЕХНОСФЕРА

VarByte

- 3 → 11 → 10000011
- 2 → 10 → 10000010
- 2018 → 11111100010 → 00001111 11100010

VarByte

\+

- простота реализации
- хорошая скорость
- эффективно для CPU

\-

- гранулярность 1 байт (меньше не получится)

Кодирование последовательностей

Тезис: оставляем крупную гранулярность, но в каждый блок запаковываем несколько чисел

ТЕХНОСФЕРА

Simple9

Гранулярность 4 байта = 32 бита = 4 маркерных бита + 28 значимых битов

Simple9

Маркерные биты описывают структуру значимых.

0000 – 1x28бит

0001 – 2x14бит

0010 – 3x9бит

0011 – 4x7бит

…

Simple9

+

- можно очень компактно записать большую последовательность маленьких чисел

-

- выбор распределения зависит от наибольшего числа в группе

PForDelta (PFD)

Тезис: отложим «плохие» числа и закодируем их отдельно – патчинг

b – базис (Nmin – 1)
k – порог кодирования

ТЕХНОСФЕРА

PForDelta (PFD)

b – базис (Nmin – 1)
k – порог кодирования

- числа из [b, b + 2^k - 2]:
  - вычитаем b
  - результат кодируем по k бит на каждое число
- «исключения»:
  - заменяем маркером
  - складываем в отдельную последовательность
  - кодируем любым способом

PForDelta (PFD)

Пример:

[5, 17, 46, 31, 10, 12, 69], b = 4, k = 5 (2^k = 32)

PForDelta (PFD)

Пример:

[5, 17, 46, 31, 10, 12, 69], b = 4, k = 5 (2^k = 32)

[5, 17, *, 31, 10, 12, *][46, 69]

PForDelta (PFD)

Пример:

[5, 17, 46, 31, 10, 12, 69], b = 4, k = 5 (2^k = 32)

[5, 17, *, 31, 10, 12, *][46, 69]

[1, 13, *, 27, 6, 8, *][46, 69]

PForDelta (PFD)

Пример:

[5, 17, 46, 31, 10, 12, 69], b = 4, k = 5 (2^k = 32)

[5, 17, *, 31, 10, 12, *][46, 69]

[1, 13, *, 27, 6, 8, *][46, 69]

d5([1, 13, 31, 27, 6, 8, 31])d([46, 69])