ТЕХНОСФЕРА

Сжатие индекса и словаря

План лекции

- Сжатие индекса
- Приемы увеличения сжатия
- ДЗ
- Вопросы обработки документов в ПС

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

Память: как правильно с ней работать?

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

Сжатие индекса

Зачем сжимать

- Экономим место
  - Особенно если RAM
- Больше помещается в память
  - Быстрее передача данных
  - {Прочитать сжатое, распаковать} может быть быстрее чем {прочитать несжатое}
  - Больше можно закешировать

Виды сжатия

- Сжатие без потерь: вся информация остаётся как есть
  - gzip/rar/…
  - png
  - Обычно используем её в ИП.

Сжатие с потерями

Сжатие с потерями

- Jpeg/архиватор Попова
- Что-то считаем возможным убрать
- Понижение капитализации, стоп-слова, морф. нормализация – может рассматриваться как сжатие с потерями.
- Ещё – удаление координат для позиций, которые вряд ли будут вверху на ранжировании.

Сжатие координатных блоков

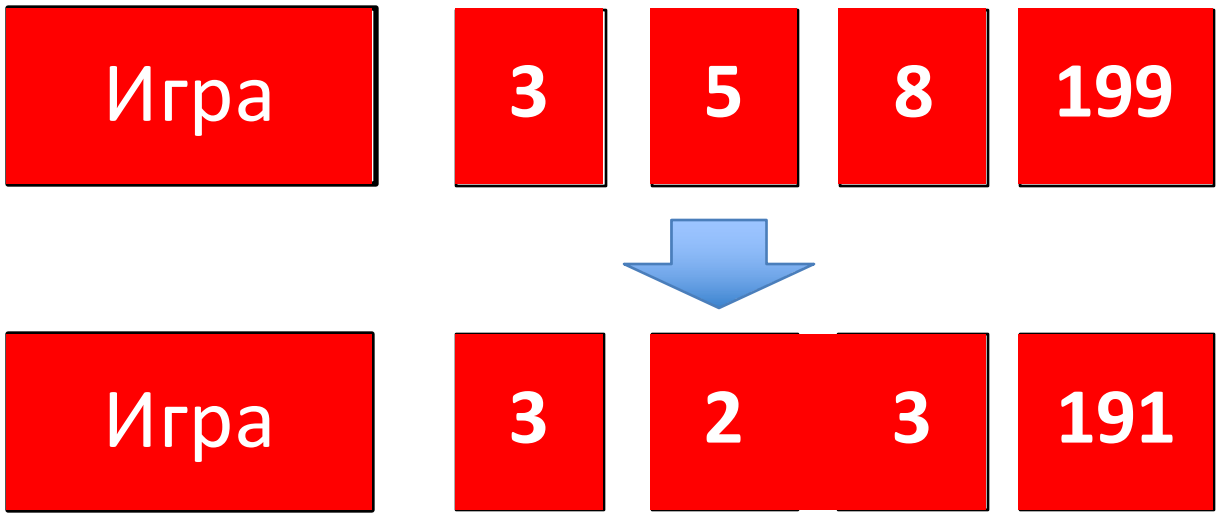
- Координатные блоки – существенная часть обратного индекса
- Будем сжимать каждый постинг
- В булевском индексе – это docID

Цель

- Предположим, мы индексируем 1 МЛН документов
  - можем использовать log2 100,000
  - 20 битов на DocID
- Наша задача: значительно меньше, чем 20 битов

Подготовка к компрессии

- Список документов выгодно хранить по возрастанию DocID
- Следовательно, можем кодировать промежутки



Цель кодирования

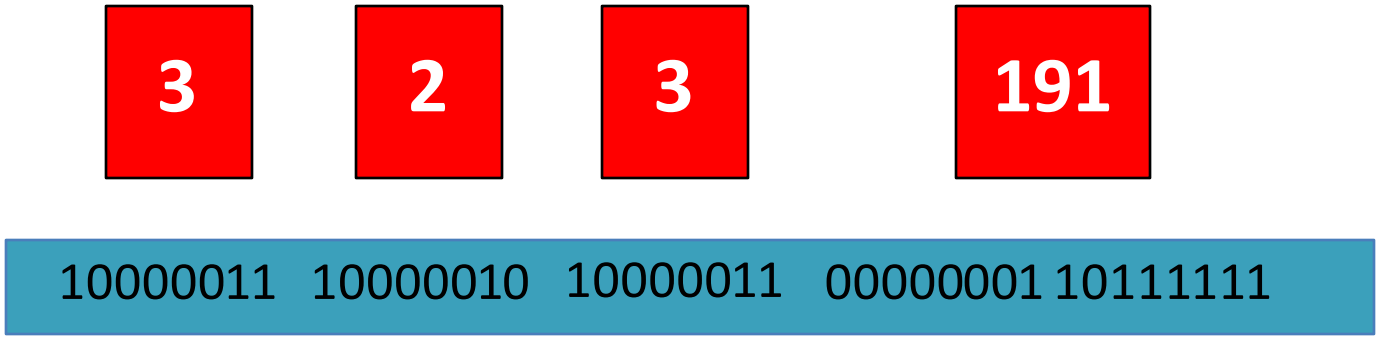
- Если средний промежуток размера G, мы хотим использовать $\sim\log_2 G$ битов на промежуток.
- Главное: кодировать каждое целое число минимальным количеством битов.
- Требуется код с переменной длиной.
- Будем достигать желаемого тем, что будем назначать короткие коды небольшим промежуткам

Код Variable Byte (VB)

- Храним признак окончания числа
- Число $G < 128$ кодируется одним байтом
- Иначе берем остаток, и кодируем его тем же алгоритмом
- Для последнего байта c=1, для остальных c=0

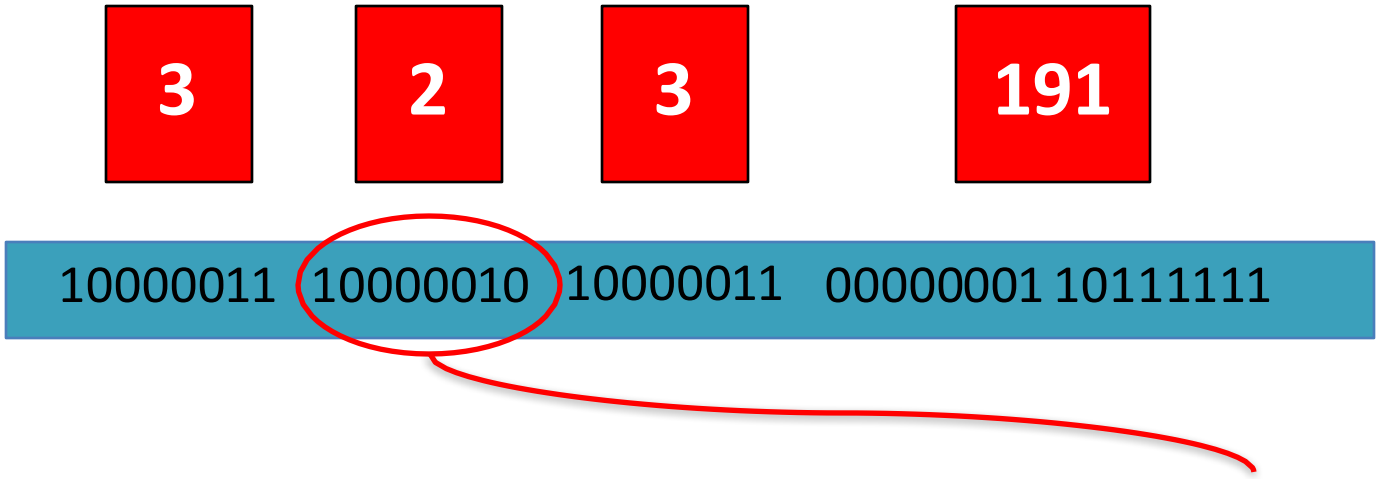
Пример кодирования varbyte

Записываем в виде непрерывной строки бит



+: Простота реализации
+: Хорошая скорость
+: Эффективно для CPU

Но есть и минусы



+: Простота реализации

+: Хорошая скорость

+: Эффективно для CPU

-: Гранулярность = 1 байт

От байт к битам

- Кодирование по байтам избыточно для малых промежутков
- Будем использовать битовое кодирование

- Важное требование побитового сжатия:
  - Кодирование длинны
  - Или недопустимая последовательность

Код Фиббоначи

1 1 2 3 5 8 13 …

Код Фиббоначи

1 1 2 3 5 8 13 …

Представим основанием СС:

… 13 8 5 3 2 1 FIB

… 32 16 8 4 2 1 BIN

Код Фибоначчи

- 11 – недопустимая комбинация
- Алгоритм довольно простой
- Сжимает эффективней gzip и varbyte

Код Фибоначчи: пример

- Закодируем число 11
- ”11” => 8+3 => 1,0,1,0,0
- Как определить конец числа?

Код Фибоначчи: пример

- Закодируем число 11
- ”11” => 8+3 => 1,0,1,0,0
- Как определить конец числа?
- 001011

Код Фибоначчи: пример

- Закодируем число 11
- ”11” => 8+3 => 1,0,1,0,0
- Примеры

1 = 1 -> 1

2 = 2 -> 1 0

4 = 3+1 -> 1 0 1

19 = 13+5+1 -> 1 0 1 0 0 1

Гамма-код (Elias Gamma)

- Кодирование:
  1. Записываем число в 2ой форме
  2. Перед двоичным представлением числа дописать нули. Кол-во нулей на единицу меньше двоичного представления числа

Гамма-код (Elias Gamma)

- Кодирование:
  1. Записываем число в 2ой форме
  2. Перед двоичным представлением числа дописать нули. Кол-во нулей на единицу меньше двоичного представления числа

Примеры:

Число	2е предст.	Кодирование
1	$2^0 + 0$	1
2	$2^1 + 0$	010
3	$2^1 + 1$	011
4	$2^2 + 0$	00100
5	$2^2 + 1$	00101

Гамма-код (Elias Gamma)

- Декодирование:
  1. Считываем нули, пусть = N
  2. Первая единица это 2^N. Считываем оставшиеся разряды числа.

Гамма-код (Elias Gamma)

- Все гамма коды имеют нечётное количество битов
- В два раза хуже лучшего результата, $\log_2 G$

- Гамма коды – префиксные коды, как и VB
- Могут использоваться для любого распределения чисел.
- Не требует параметров.

Гамма-код (Elias Gamma)

- Нужно учитывать границы машинных слов – 8, 16, 32, 64 бит
  - Операции, затрагивающие границы машинных слов, значительно медленнее
- Работа с битами может быть медленной.
- VB кодировка выровнена по границам машинных слов и потенциально более быстрая.
- VB значительно проще в реализации.

Rice Encoding

- Рассмотрим среднее кодируемых чисел, $=g$
- Округлим g до ближайшей степени 2, $=b$
- Каждое число x будем представлять как
  - $(x-1)/b$ в унарном коде
  - $(x-1) \bmod b$ в бинарном коде

RiceEncoding пример

DocID: 34, 178, 291, 453

Промежутки: 34, 144, 113, 162

Среднее: $g = (34+144+113+162)/4 = 113{,}33$

Округляем: $b = 64$ (6 бит)

Число	Разложение	Кодирование	
34	64*0 + (34-1)	0	100001
144	64*2 + (144-1) & 63	110	001111
113	64*1 + (113-1) & 63	10	110000
162	64*2 + (162-1) & 63	110	100001

Свойства RiceEncoding

- Можно подобрать g как для всего индекса, так и для отдельного терма
- Более того – можно подобрать для отдельных промежутков

- Лучше сжимает, но медленнее VarByte

Также см. Golomb Encoding

Основные выводы

- VarByte – быстрый, но избыточный
- Fibonacci – компактный, но медленно
- Gamma – избыточны + нечетны по битам

Кодирование последовательностей

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

Simple9

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

Simple9

Маркерные биты описывают структуру значимых.

0000 – 1x28бит

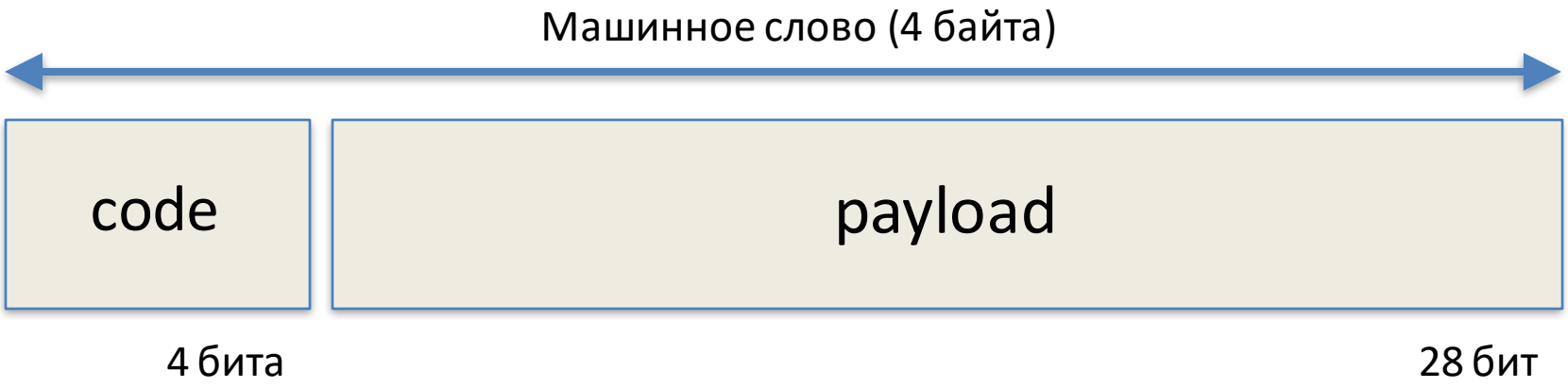
0001 – 2x14бит

0010 – 3x9бит

0011 – 4x7бит

…

Simple9



Simple9: payload

- 1 28-и битное число
- 2 14-и битных числа
- 3 9-и битных числа (и теряем 1 бит)
- 4 7-и битных числа
- 5 5-и битных чисел (и теряем 3 бита)
- 7 4-х битных чисел
- 9 3-х битных чисел (и теряем 1 бит)
- 14 2-х битных чисел
- 28 1 битных чисел

Simple9

- Плюсы:
  - Очень быстр при распаковке ($2*10^9$ чисел/сек)
  - Компактен
- Минус:
  - Избыточен для одного числа

Подведем итог

- Можем быстро объединять списки
- Применяя сжатие значительно уменьшаем размер индекса (-400%)

Домашнее задание

Дано: дамп lenta.ru (10k документов)

Документы доступны по адресу:
https://cloud.mail.ru/public/FnMq/qCNif6bFG/dataset

Комментарий:

- Разбор документов на слова лучше организовать с помощью regexp-а r'\w+'
- Необходимая нормализация: приводим все слова к нижнему регистру.

Необходимо:

- Создать индекс
- Реализовать булев поиск

Домашнее задание

Дано: дамп lenta.ru (10k документов)

Необходимо:

- Создать индекс
- Реализовать булев поиск

Если у вас **все работает корректно**, то вы сможете претендовать на оценку **10 баллов**.

Дополнительно к предыдущему:

- имплементировать любой из алгоритм сжатия из лекции **+5 баллов**
- реализовать потоковую обработку дерева запроса **+5 баллов**
- сравнительный анализ и выводы **+5 баллов**

Разбор запроса

docker & запуск & (Ubuntu || убунту)

1. операции и термы
2. определяем приоритет и порядок операций
3. дерево строится от меньшего приоритета (корень, исполняется последним)

Общий workflow индексации

1. индексация входных данных (index.sh)

Наиболее затратна по времени (много данных + можем себе позволить время)
Можем индексировать по частям
Содержит ссылки на блоки
сохраняем соответствие url <-> docID (в выдаче нужны урлы)

Общий workflow индексации

1. индексация входных данных (index.sh)
2. оптимизация индекса

Общий workflow индексации

1. индексация входных данных (index.sh)
2. оптимизация индекса
3. построение словаря (make_dict.sh)

Общий workflow индексации

1. индексация входных данных (index.sh)
2. оптимизация индекса
3. построение словаря (make_dict.sh)
4. поиск (search.sh):
    1. строим Q-Tree
    2. ставим в соответствие блоки
    3. ищем конкретные docID → преобразуем в URL
    4. …
    5. PROFIT!

Домашнее задание

А конкретно:

- разобрать текстовый запрос простого формата
- вывести подходящие под булев запрос URL-ы

Формат вывода:

ИСХОДНЫЙ ЗАПРОС
КОЛ-ВО результатов
URL1
URL2
...

Пример:
Путин & Медведев
2
https://lenta.ru/news/2015/08/30/putin/
https://lenta.ru/photo/2015/08/30/medput/

Внимание: выводимые url-ы должны быть в пордке возрастания docid!

Домашнее задание

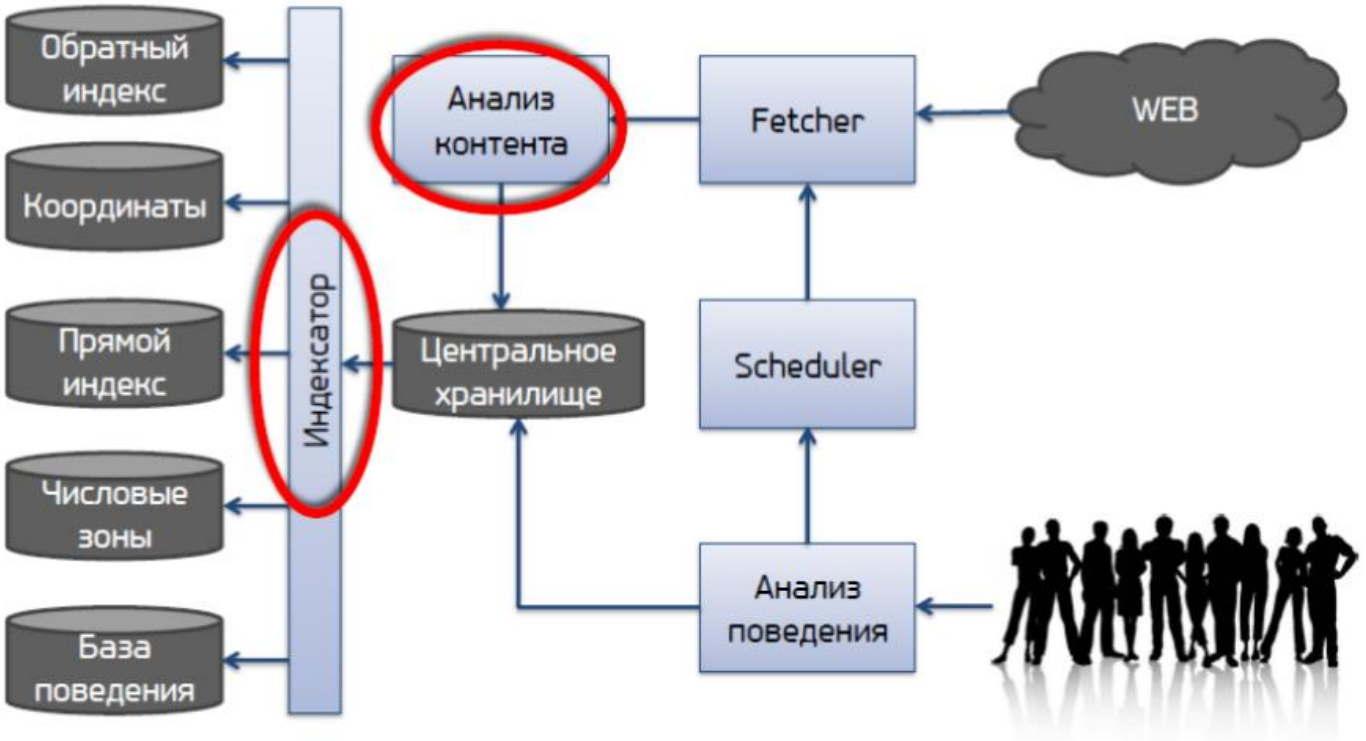
Куда отправлять код:

Код запакованный в .tgz отправляйте на gulin.vladimir.sfera@mail.ru
В теме письма обязательно указывайте. Формат: [Ir-ts] idx, Иван Иванов

В самом письме должно быть описание реализованной логики.
Как запустить? Как обработать запрос и т.д.

Срок сдачи: **14 октября 18:00.**

Этапы предобработки документа



Терминология

- *Токен* - экземпляр последовательности символов в документе, объединенных в семантическую единицу для обработки
- *Термин* - “нормализованный” токен (регистр, морфология, исправление ошибок и т.п.)

Нормализация

- Необходимо “нормализовывать” термины как в индексируем тексте, так и в запросе
- Например: Желательно считать одинаковыми термины
  U.S.A. и USA
- Обычно термины объединяются в классы эквивалентности
- Можно поступить наоборот, расширять:
  - window → window, windows
  - windows → Windows, windows
  - Windows (нет расширения)
- Такой подход более гибкий, но более ресурсоемкий

Документы

Делаем неявное предположение:

- Мы знаем, что такое документ
- Каждый документ доступен для автоматического разбора

Вопрос:

- Какие тут есть проблемы?

Лингвистика при обработке документов

- Определение формата документа (pdf, word, html и т.д.)
- Определение кодировки документа
- Определение языка документа
- Токенизация и сегментация
- Нормализация и лемматизация
- Выделение объектов и зон
- Вычисление текстовых факторов

Нормализация

Нормализация зависит от языка документа

- PETER WILL NICHT MIT. $\rightarrow$ MIT = mit
- He got his PhD from MIT. $\rightarrow$ MIT $\neq$ mit

Нормализация

Ударения и диакритика

- résumé vs. resume
- Умуляуты: Universität vs. Universitaet (заменяем на специальную последовательность «ae» или даже «æ»)

- Самый важный вопрос: как пользователи предпочитают писать запросы с этими словами?

Нормализация

Классы эквивалентности

- Soundex
  - фонетическая эквивалентность, Muller = Mueller
- Тезаурус
  - семантическая эквивалентность, car = automobile

Нормализация

Регистр

- Понизить регистр всех букв.
- Возможны исключения, например, для капитализированных слов внутри предложения.
  - MIT и mit
  - Fed и fed
  - КОТ и кот (Калининградская областная таможня)
- NB: немецкий → существительные с большой буквы
- Часто лучше понижать всё, потому что пользователи не заботятся о капитализации в запросах.

Токенизация

Проблемы токенизации

- Hewlett-Packard
- State-of-the-art
- co-education
- San Francisco
- York University vs. New York University

Проблемы токенизации

Числа

- 3/20/91
- 20/3/91
- Mar 20, 1991
- B-52
- 100.2.86.144
- (800) 234-2333
- 800.234.2333

Кодировки

- ASCII (ISO 646) - 7-битный стандарт
- ISO 8859
  - 8859-1 (ISO Latin-1)
  - ISO 8859-5
- Русские кодировки
  - CP1251 (windows)
  - 866 (dos)
  - KOI8-R (unix)
- Unicode
  - UTF-8
  - UTF-16
  - UTF-32