



ТЕХНОСФЕРА

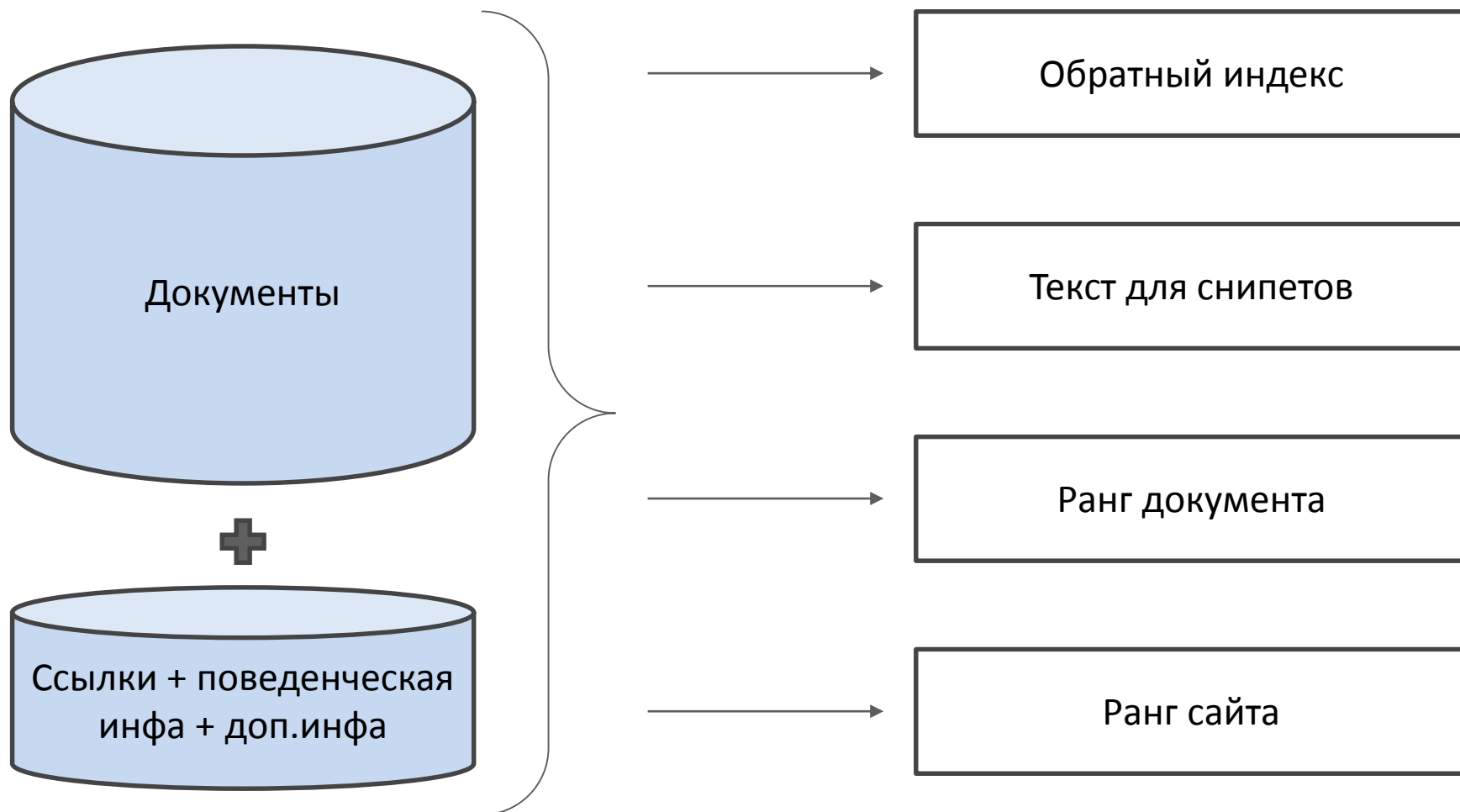
Индексация

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

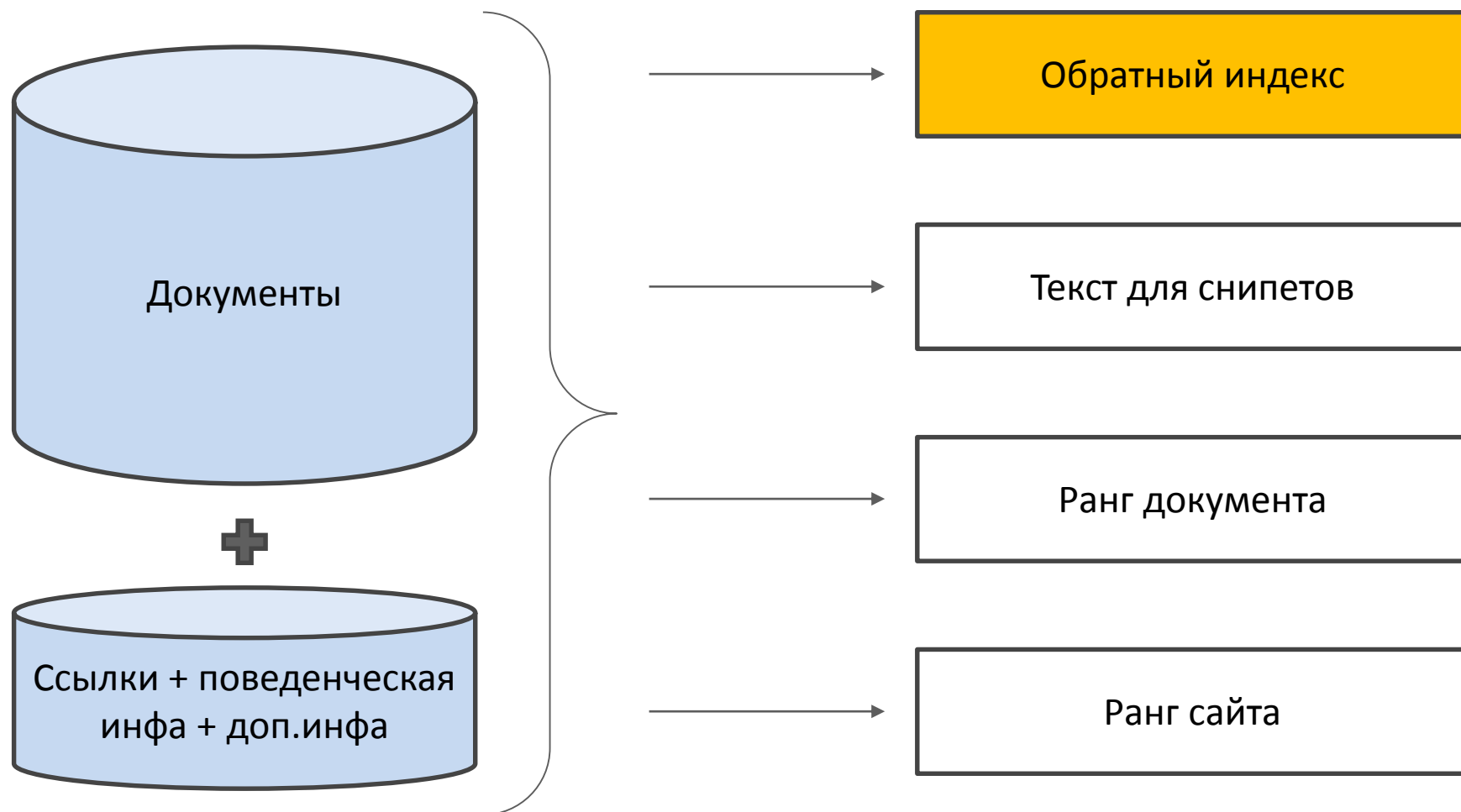
План лекции:

- 1. Обратный индекс**
- 2. Поиск по обратному индексу, пересечение блоков**
- 3. Сжатие индекса**

Что мы получаем из документов



Что мы получаем из документов



От документа к индексу

1. Документ → текст

От документа к индексу

1. Документ \rightarrow текст
2. Текст \rightarrow слова

От документа к индексу

1. Документ → текст
2. Текст → слова
3. Слова → леммы (зачем?)

От документа к индексу

1. Документ → текст
2. Текст → слова
3. Слова → леммы (зачем?)

Лемма – лингв. нормализованная, основная форма слова, вместе с информацией о построении других форм

От документа к индексу

1. Документ → текст
2. Текст → слова
3. Слова → леммы (зачем?)
4. Лемма → список документов, в которых встречается:
«posting list», «инвертированный список» и т.д.

От слов к ~~действию~~ числам

Документ <-> URL

От слов к ~~действию~~ числам

Документ <-> URL <-> docID

От слов к ~~действию~~ числам

Документ \leftrightarrow URL \leftrightarrow docID

Слово \leftrightarrow termID (например, hash)

Обратный индекс



Чем дополнить обратный индекс?



Чем дополнить обратный индекс?

1. Ранг термина
2. Координаты

Нужные данные для поиска и ранжирования
Мы ограничены размером и скоростью

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

Физические ограничения

1. RAM быстрее HDD на 2-3 порядка
2. SSD? Быстро, но пока недостаточно надежно для ВНС
3. Гибриды – дорого и малый объем
4. RAM ограничена по объему

Скорость меряется в IOpS – Input/Output per Second

Память: как правильно с ней работать?

1. Считать блок:

- Спозиционироваться
- Считать

2. Что быстрее?

1. Считать 1GB, записанный непрерывно
2. Считать 1024 блока по 1MB
3. Считать 1024×1024 блока по 1KB

Память: как правильно с ней работать?

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

Наша конфигурация

Компонент	тестовый кластер
CPU	Xeon: 2x8 core, HT*; 2.4 Ghz
RAM	48 Gb
Диски	1Tb+ SATA
Скорость	~10ms

* Hyper Threading – технология, «честно» реализующая параллельную работу 2 тредов

Размер индекса

1. 10кМ документов
2. 1 документ \sim 70Kb
3. 1 лемма \sim 8b

Проблема:

очень большой словарь

Разделяемый индекс

Как поделить большой индекс между несколькими серверами?



Разделяемый индекс

1. Сервер \leftrightarrow терм
2. Сервер \leftrightarrow документ

Бинарный поиск

$A \& B$

$A || B$

$\neg B$

Пересечение блоков





pathfinder kingmaker


×


Найти


 Везде

 Картинки

 Видео

 Приложения


 Новости


 Ответы


Пересечение блоков


 × Найти


Везде

 Картинки

 Видео

 Приложения

 Новости

 Ответы

pathfinder

94

7

12

55

57

43

kingmaker

94

1

7

Пересечение блоков

 × Найти

Везде

Картинки

Видео

Приложения

Новости

Ответы

pathfinder

94

7

12

55

57

43

kingmaker

94

1

7

Пересечение блоков

 × Найти[Везде](#)[Картинки](#)[Видео](#)[Приложения](#)[Новости](#)[Ответы](#)

pathfinder

7

12

43

55

57

94

kingmaker

1

7

94

Пройти по всему списку – долго

Пересечение блоков

 × Найти

Везде

Картинки

Видео

Приложения

Новости

Ответы

pathfinder



kingmaker

Jump Tables!

Пересечение блоков

 × Найти

Везде

Картинки

Видео

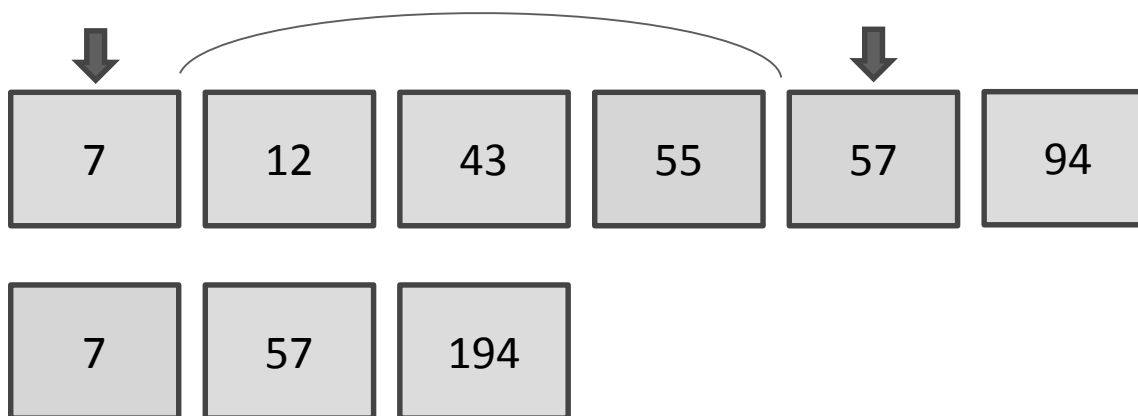
Приложения

Новости

Ответы

pathfinder

IT








Jump Tables!

Пересечение блоков



x

Найти

Везде Картинки Видео Приложения Новости Ответы

pathfinder

7

12

43

55

57

94

JT

7

57

194

JT2

7

243

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

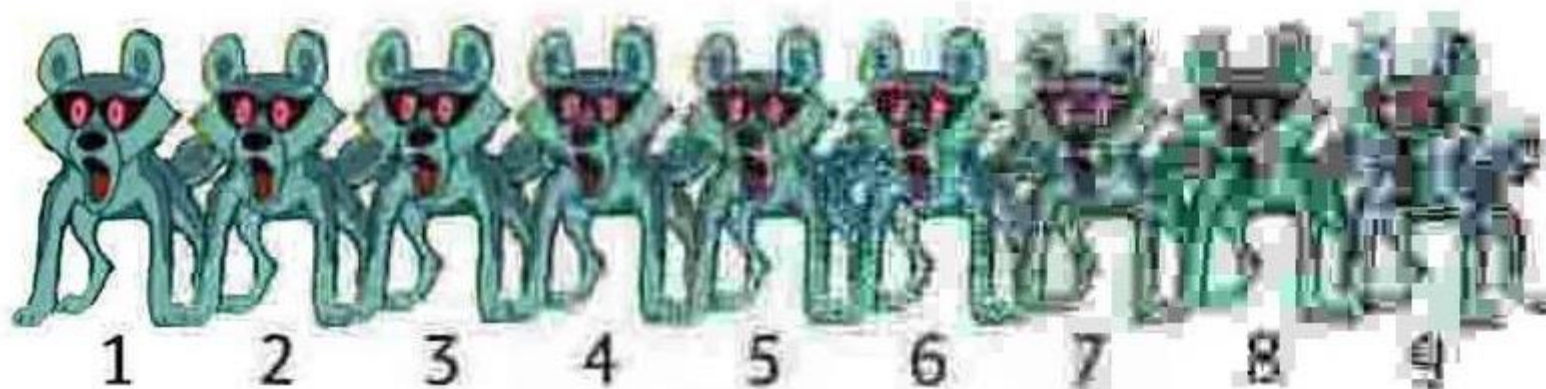
Виды сжатия

В ИП: сжатие без потерь (почему?)

1. gzip/rar/lzo
2. png
3. и т.д.

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?

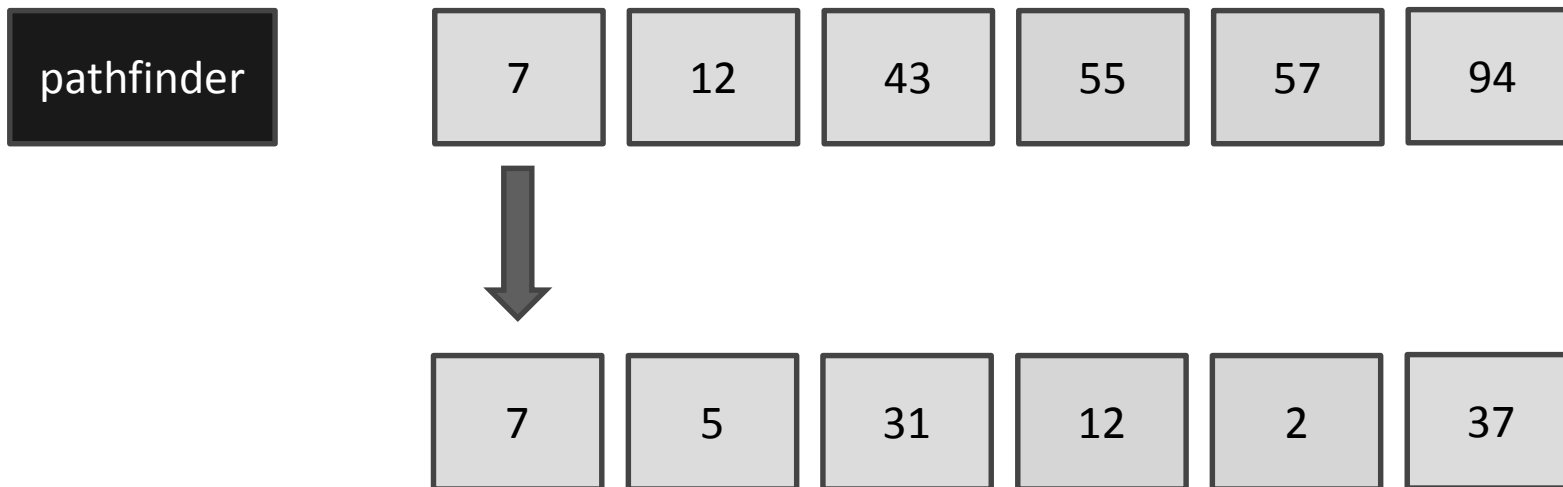
1. Постинг-листы
2. Словарь

Цель

- Индексируем 1М документов
 - docID – int – 32b
 - но 1М можно записать 20b
- Наша цель: заиспользовать меньше 20b

Подготовка к компрессии

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



Цель кодирования

- Если средний промежуток равен G , то мы хотим использовать $\log_2 (G)$ для кодирования
- Нужен код переменной длины!
- Т.о. маленьким промежуткам будет требоваться меньше бит

VarByte

- Храним признак окончания числа (1 в старшем разряде)
- $G < 128$ кодируется 1 байтом
- $G \geq 128$ кодируется 2+ байтами («дополнительные» с 0 в старшем разряде)

VarByte

- $3 \rightarrow 1000\ 0011$
- $2 \rightarrow 1000\ 0010$

VarByte

- $3 \rightarrow 1000\ 0011$
- $2 \rightarrow 1000\ 0010$
- $d2(2018) = 11111100010$

VarByte

- $3 \rightarrow 1000\ 0011$
- $2 \rightarrow 1000\ 0010$
- $d2(2018) = 1111\ 1100010$

VarByte

- 3 → 1000 0011
- 2 → 1000 0010
- 2018 → 00001111 11100010

VarByte

- 3 → 1000 0011
- 2 → 1000 0010
- 2018 → 00001111 11100010

- + простота реализации
- + хорошая скорость
- + эффективно для CPU
- гранулярность 1 байт (меньше не получится)

Побитовая гранулярность

- Как определять конец последовательности?

Побитовая гранулярность

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

???



???

$$1 + 1 = 2$$

$$1 + 2 = 3$$

$$2 + 3 = 5$$

$$3 + 5 = 8$$

$$5 + 8 = 13$$

...



Код Фиббоначи

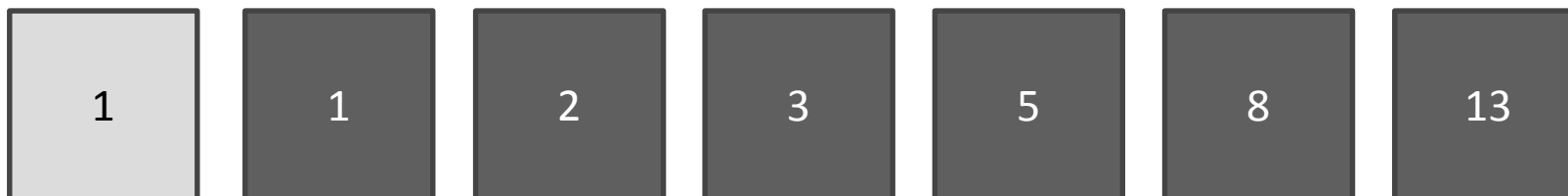


Код Фибоначчи



Используем числа Фибоначчи как значения разрядов в новой СИ

Код Фибоначи



Используем числа Фибоначи как значения разрядов в новой СИ:

... 21 13 8 5 3 2 1

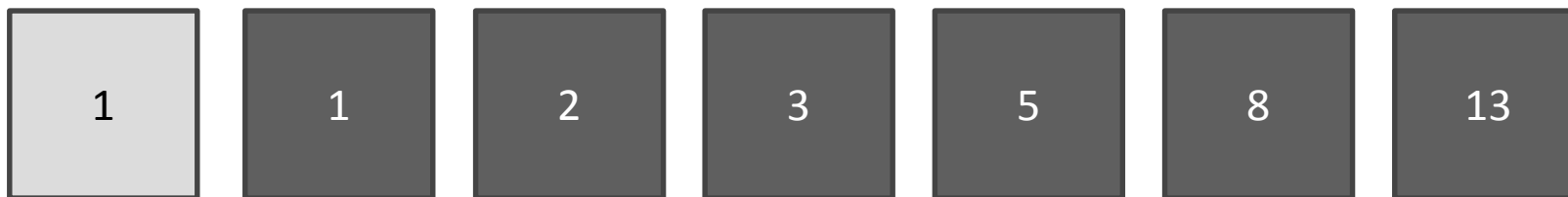
$$1 = 1 \rightarrow 1$$

$$2 = 2 \rightarrow 10$$

$$4 = 3+1 \rightarrow 101$$

$$19 = 13+5+1 \rightarrow 101001$$

Код Фибоначи



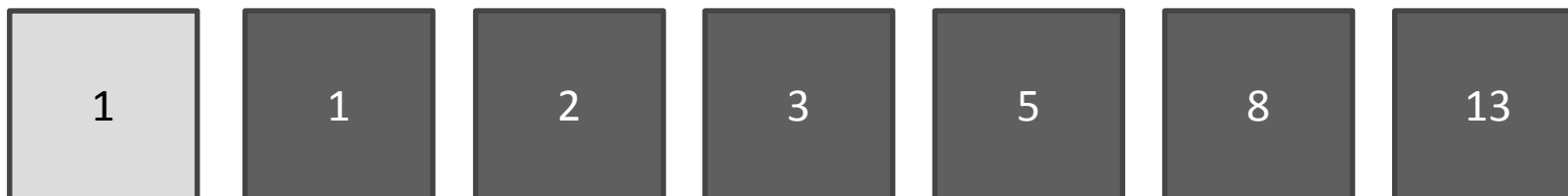
Используем числа Фибоначи как значения разрядов в новой СИ:

... 21 13 8 5 3 2 1

$$11 = 8 + 3 \rightarrow 1 0 1 0 0$$

Как определить конец числа?

Код Фибоначи



Используем числа Фибоначи как значения разрядов в новой СИ:

... 21 13 8 5 3 2 1

$$11 = 8 + 3 \rightarrow 10100$$

Как определить конец числа?

001011

Elias Gamma

Кодируем длину числа нулями:

- $G = 2^n$ или 2^{n+1}
- $EG(G) = \{0\}^n + \text{bin}(G)$

Elias Gamma

Кодируем длину числа нулями:

- $G = 2^n + m$
- $EG(G) = \{0\}^n + \text{bin}(G)$

Число	2^n	Elias Gamma
1	$2^0 + 1$	1
2	$2^1 + 0$	010
3	$2^1 + 1$	011
5	$2^2 + 1$	00101
192	$2^7 + 2^6 + 0$	000000011000000

Elias Gamma

Преимущества:

- удобно считывать (нули – N , первая единица – 2^N и т.д.)
- не требует внешних параметров
- префиксный
- подходит для любого распределения чисел

Недостатки:

- все гамма-коды – из нечетного количества бит
- в 2 раза хуже baseline ($\log_2(G)$)

Параметризованное кодирование:

Rice Encoding

- среднее число: g
- округлим до степени 2: b
- каждое число представим в виде записи из 2 частей:
 - $(x-1)/b$ – в унарном
 - $(x-1) \bmod b$ – в бинарном

Параметризованное кодирование: Rice Encoding

Пример:

docID :: 34, 178, 291, 453

delta :: 34, 144, 113, 162

$g = (34 + 144 + 113 + 162) / 4 = 113.3$

$b = 64 (2^6)$

Число	Разложение	Код
34	$64 * 0 + (34 - 1) \& 63$	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

