

Изграждане на индекси

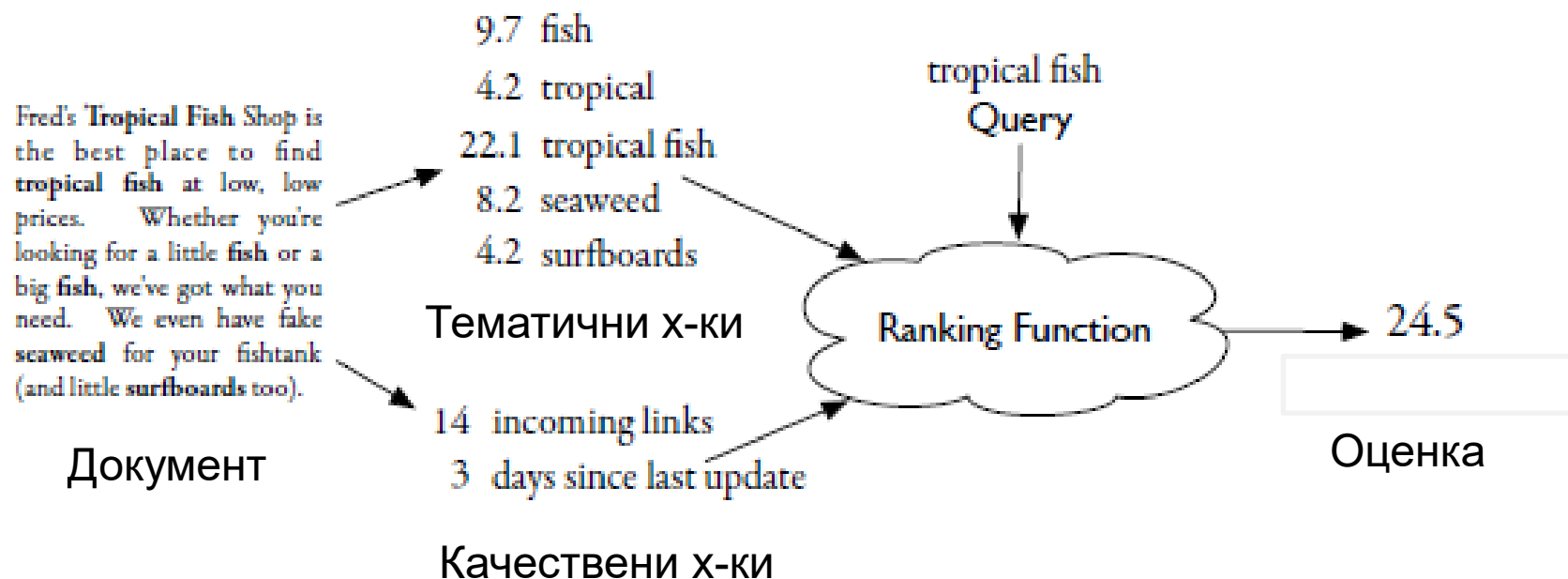
проф. д-р инж. Христо Вълчанов

<http://cs.tu-varna.bg>

Индекси

- Проектирани са да поддържат *търсене*.
- Търсещите машини използва специфична форма на търсене – *ranking*
 - Документите се извличат в подреден ред в зависимост от оценката на документа.

Абстрактен модел на рейтинговане



Текст за индексиране

d1: The jaguar is a New World mammal of the Felidae family.

d2: Jaguar has designed four new engines.

d3: For Jaguar, Atari was keen to use a 68K family device.

d4: The Jacksonville Jaguars are a professional US football team.

d5: Mac OS X Jaguar is available at a price of US \$199 for Apple's new "family pack".

d6: One such ruling family to incorporate the jaguar into their name is Jaguar Paw.

d7: It is a big cat.

Отделяне на думите

d1: the₁ jaguar₂ is₃ a₄ new₅ world₆ mammal₇ of₈ the₉ felidae₁₀ family₁₁

d2: jaguar₁ has₂ designed₃ four₄ new₅ engines₆

d3: for₁ jaguar₂ atari₃ was₄ keen₅ to₆ use₇ a₈ 68k₉ family₁₀ device₁₁

d4: the₁ jacksonville₂ jaguars₃ are₄ a₅ professional₆ us₇ football₈ team₉

d5: mac₁ os₂ x₃ jaguar₄ is₅ available₆ at₇ a₈ price₉ of₁₀ us₁₁ \$199₁₂ for₁₃ apple's₁₄ new₁₅ family₁₆ pack₁₇

d6: one₁ such₂ ruling₃ family₄ to₅ incorporate₆ the₇ jaguar₈ into₉ their₁₀ name₁₁ is₁₂ jaguar₁₃ paw₁₄

d7: it₁ is₂ a₃ big₄ cat₅

Нормализиране на думи

d1: the₁ jaguar₂ be₃ a₄ new₅ world₆ mammal₇ of₈ the₉ felidae₁₀ family₁₁

d2: jaguar₁ have₂ design₃ four₄ new₅ engine₆

d3: for₁ jaguar₂ atari₃ be₄ keen₅ to₆ use₇ a₈ 68k₉ family₁₀ device₁₁

d4: the₁ jacksonville₂ jaguars₃ be₄ a₅ professional₆ us₇ football₈ team₉

d5: mac₁ os₂ x₃ jaguar₄ be₅ available₆ at₇ a₈ price₉ of₁₀ us₁₁ \$199₁₂ for₁₃ apple₁₄ new₁₅ family₁₆ pack₁₇

d6: one₁ such₂ rule₃ family₄ to₅ incorporate₆ the₇ jaguar₈ into₉ their₁₀ name₁₁ be₁₂ jaguar₁₃ paw₁₄

d7: it₁ be₂ a₃ big₄ cat₅

Премахване на общите термини

d1: jaguar₂ new₅ world₆ mammal₇ felidae₁₀ family₁₁

d2: jaguar₁ design₃ four₄ new₅ engine₆

d3: jaguar₂ atari₃ keen₅ 68k₉ family₁₀ device₁₁

d4: jacksonville₂ jaguars₃ professional₆ us₇ football₈ team₉

*d5: mac₁ os₂ x₃ jaguar₄ available₆ price₉ us₁₁ \$199₁₂ apple₁₄ new₁₅
family₁₆ pack₁₇*

*d6: one₁ such₂ rule₃ family₄ incorporate₆ jaguar₈ their₁₀ name₁₁ jaguar₁₃
paw₁₄*

d7: big₄ cat₅

Индекси

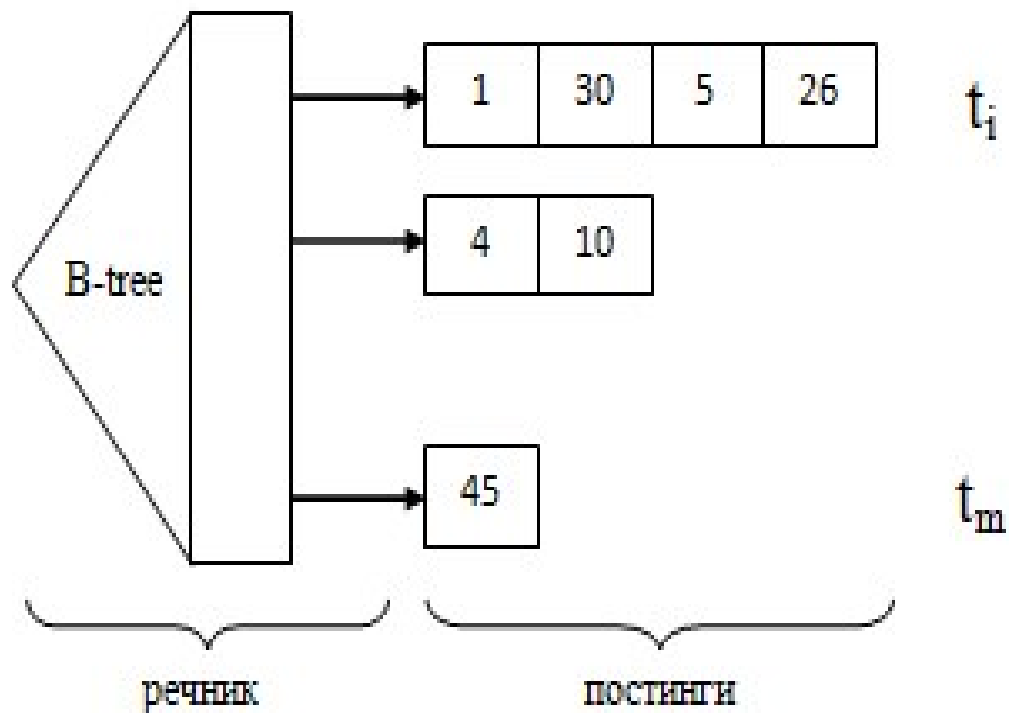
Forward index – списък от документи, във всеки документ се съдържат думи.

документи -> към -> думи

Inverted index – списък от думи и документи в които те се съдържат.

думи -> към -> документи

Изграждане на индекс



Инвертен индекс - пример

d1: jaguar₂ new₅ world₆ mammal₇ felidae₁₀ family₁₁

d2: jaguar₁ design₃ four₄ new₅ engine₆

d3: jaguar₂ atari₃ keen₅ 68k₉ family₁₀ device₁₁

d4: jacksonville₂ jaguars₃ professional₆ us₇ football₈ team₉

d5: mac₁ os₂ x₃ jaguar₄ available₆ price₉ us₁₁ \$199₁₂ apple₁₄
new₁₅ family₁₆ pack₁₇

d6: one₁ such₂ rule₃ family₄ incorporate₆ jaguar₈ their₁₀
name₁₁ jaguar₁₃ paw₁₄

d7: big₄ cat₅

family
football
jaguar
new
rule
us
world
...

d_1, d_3, d_5, d_6

d_4

$d_1, d_2, d_3, d_4, d_5, d_6$

d_1, d_2, d_5

d_6

d_4, d_5

d_1

Позиция на думата

<i>family</i>	$d_1/11, d_3/10, d_5/16, d_6/4$
<i>football</i>	$d_4/8$
<i>jaguar</i>	$d_1/2, d_2/1, d_3/2, d_4/3, d_5/4, d_6/8$
<i>new</i>	$d_1/5, d_2/5, d_5/15$
<i>rule</i>	$d_6/3$
<i>us</i>	$d_4/7, d_5/11$
<i>world</i>	$d_1/6$
<i>...</i>	

Оценка релевантността на документите

- Релевантността се измерва чрез присвояване на тегло към срещанията на термина в документа.
- Често срещан термин в документа е повече релевантен за индексирание на документа.
- Рядко срещан термин в колекция засилва релевантността на документ.
- Често срещан термин в много документи е по-малко отличителен.

Оценка релевантността на документите

term frequency - inverse document frequency (tf-idf)

- **term frequency** – честотата на срещанията на термин към общия брой термини в документ

$$tf(t, d) = \frac{n_{t,d}}{\sum_{t'} n_{t',d}}$$

Оценка релевантността на документите

term frequency - inverse document frequency (tf-idf)

- **inverse document frequency** – важността на термин в колекция от документи

$$idf(t) = \log \frac{|D|}{|\{d' \in D | n_{t,d'} > 0\}|}$$

Оценка релевантността на документите

term frequency - inverse document frequency (tf-idf)

tf - idf

$$tfidf(t, d) = \frac{n_{t,d}}{\sum_{t'} n_{t',d}} \cdot \log \frac{|D|}{|\{d' \in D | n_{t,d'} > 0\}|}$$

Модифициран индекс

family	$d_1/11/.13, d_3/10/.13, d_5/16/.7, d_6/4/.8$
football	$d_4/8/.47$
jaguar	$d_1/2/.04, d_2/1/.04, d_3/2/.04, d_4/3/.04, d_5/4/.04, d_6/8/.04$
new	$d_1/5/.20, d_2/5/.24, d_5/15/.10$
rule	$d_6/3/.28$
us	$d_4/7/.30, d_5/11/.15$
world	$d_1/6/.47$
...	

Опростен индексер

```
procedure BUILDINDEX( $D$ )  
   $I \leftarrow \text{HashTable}()$   
   $n \leftarrow 0$   
  for all documents  $d \in D$  do  
     $n \leftarrow n + 1$   
     $T \leftarrow \text{Parse}(d)$   
    Remove duplicates from  $T$   
    for all tokens  $t \in T$  do  
      if  $I_t \notin I$  then  
         $I_t \leftarrow \text{Array}()$   
      end if  
       $I_t.\text{append}(n)$   
    end for  
  end for  
  return  $I$   
end procedure
```

Размер на индекса

Пример:

Колекция от е-мейли. Всеки мейл е с размер средно 1Kb и всеки мейл съдържа средно по 100 думи.

Колекцията съдържа 1 000 000 мейла в обем 1Gb.

Брой думи: 100×10^6

След парсване и токенизиране: 200 000 различни термина

1. Индексът съдържа 200 000 списъка
2. При допускане, че 20% от термините в документ се срещат 2 пъти, то всеки документ се среща в 80 списъка
3. Всеки списък съдържа средно 400 записа
4. При представяне на ID на документ с 4byte integer, то средния размер на списък е 1600bytes
5. Целият индекс ще съдържа $400 \times 200\,000 = 80\,000\,000$ записа
6. **Размерът на индекса е 320 Mb**

Размер на индекса

Добавяне на допълнителна информация към елементите в списъците:

- Позиция
- Тегло

Допълнително заета памет: 8 bytes

Целият индекс ще съдържа $80 \times 12 \times 10^6 = 960 \text{ Mb}$

Групиране на индекси (Merging)

Адресира ограниченията в обема памет.:

- Изграждане на индексните структури докато има налична памет;
- Запис на частичния индекс във файл;
- Започване изграждане на нов индекс в паметта;
- В края на процеса дискът съдържа множество частични индекси, които се групират заедно.

Частичните индекси трябва да се изграждат така, че да се групират в малки части (най-често в азбучен ред).

Групиране на индекси (Merging)

Index A

aardvark	2	3	4	5	apple	2	4
----------	---	---	---	---	-------	---	---

Index B

aardvark	6	9	actor	15	42	68
----------	---	---	-------	----	----	----

Index A

aardvark	2	3	4	5								apple	2	4
----------	---	---	---	---	--	--	--	--	--	--	--	-------	---	---

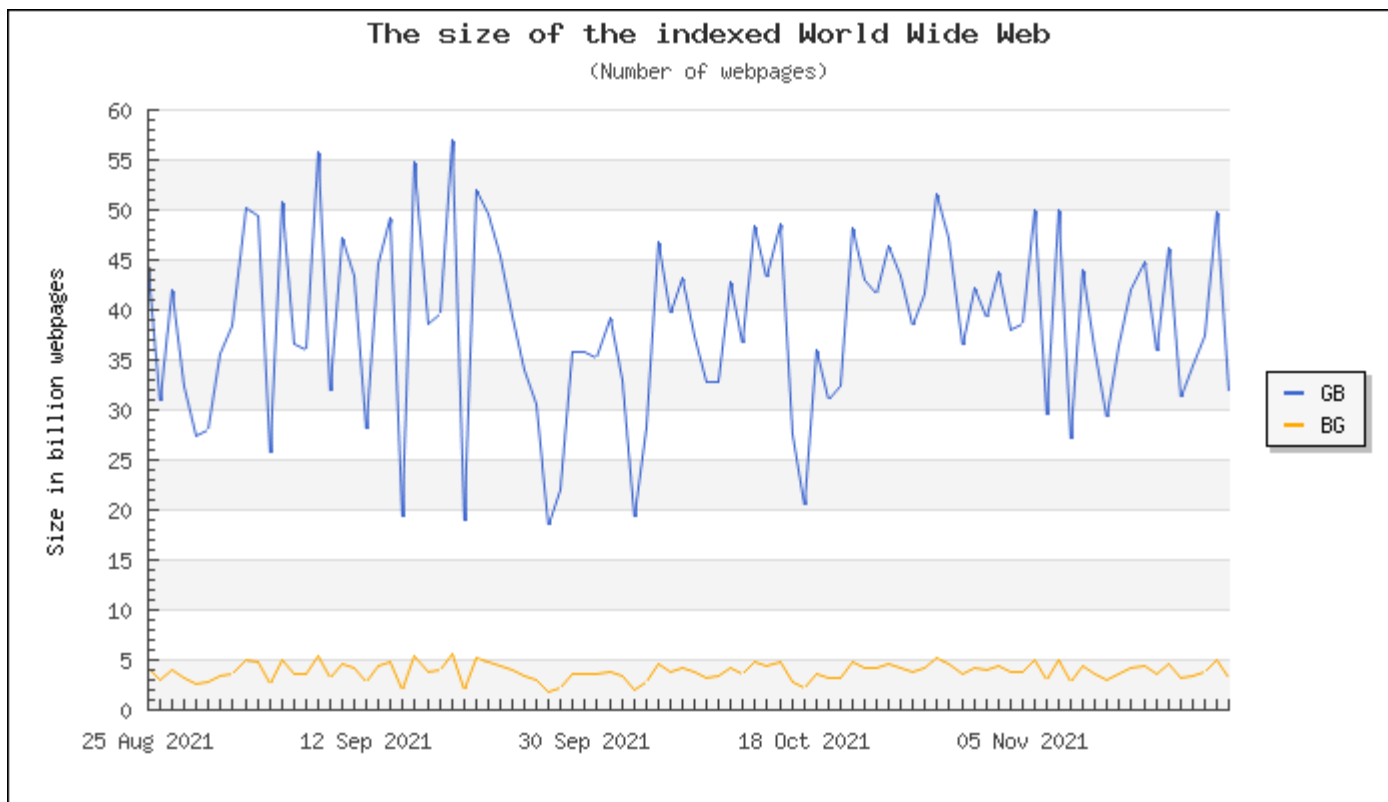
Index B

aardvark						6	9	actor	15	42	68			
----------	--	--	--	--	--	---	---	-------	----	----	----	--	--	--

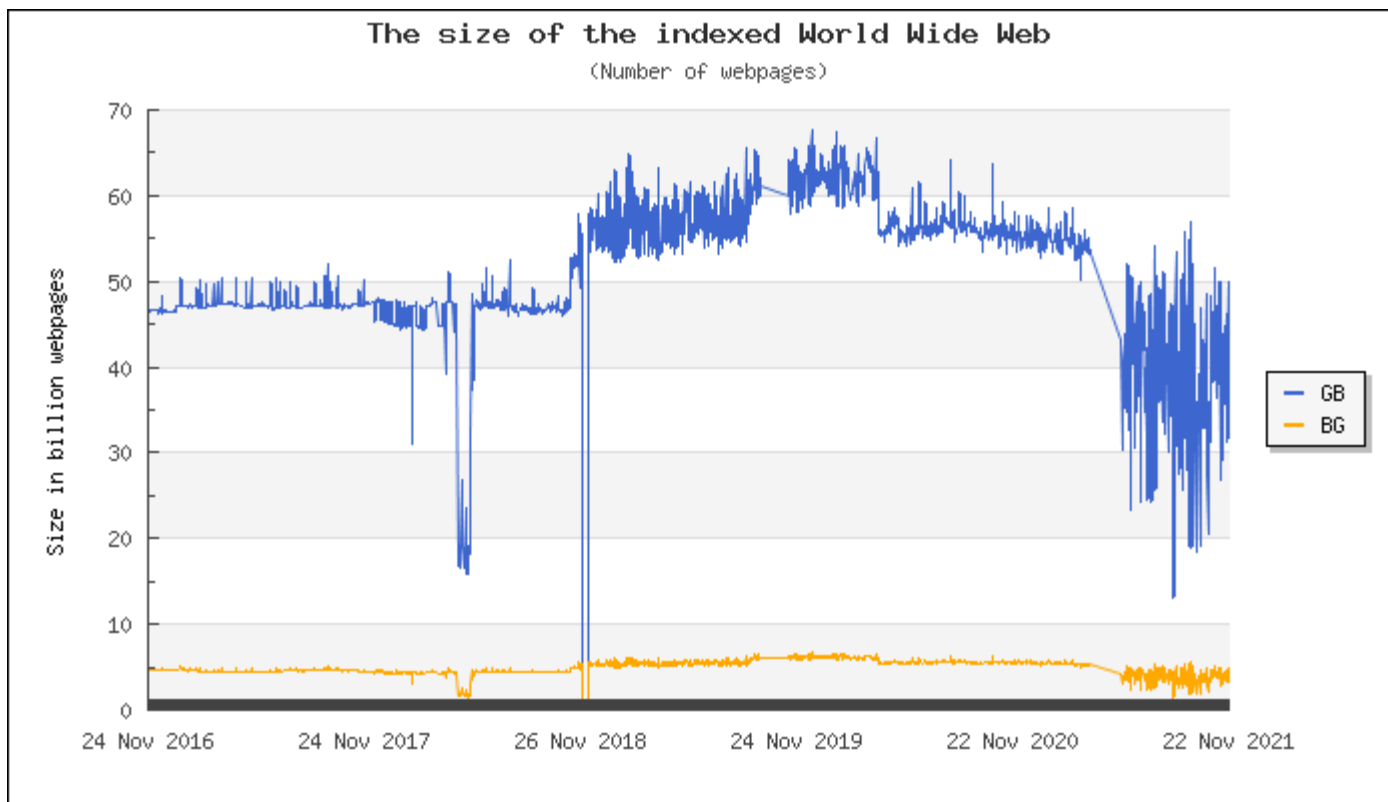
Combined index

aardvark	2	3	4	5	6	9	actor	15	42	68	apple	2	4
----------	---	---	---	---	---	---	-------	----	----	----	-------	---	---

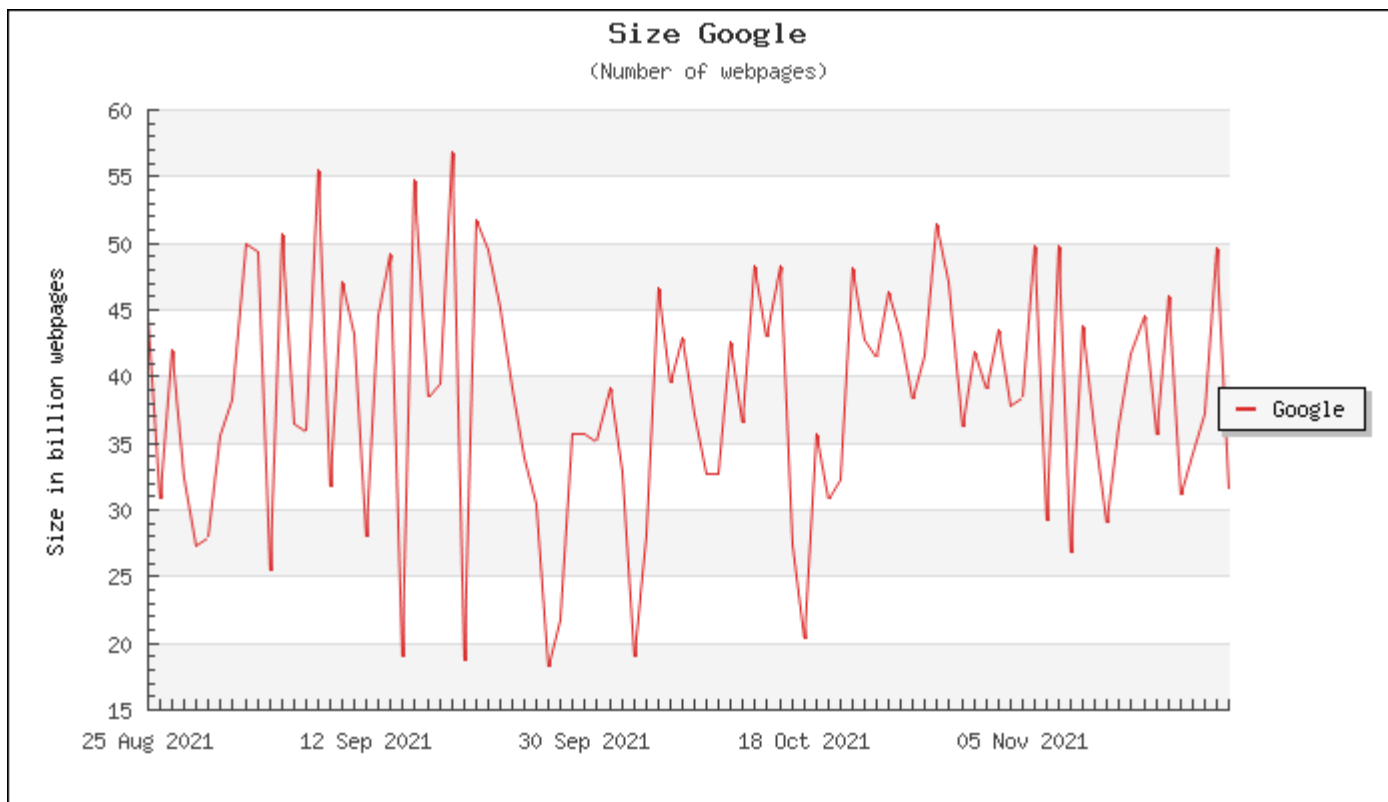
Индексиране в WWW



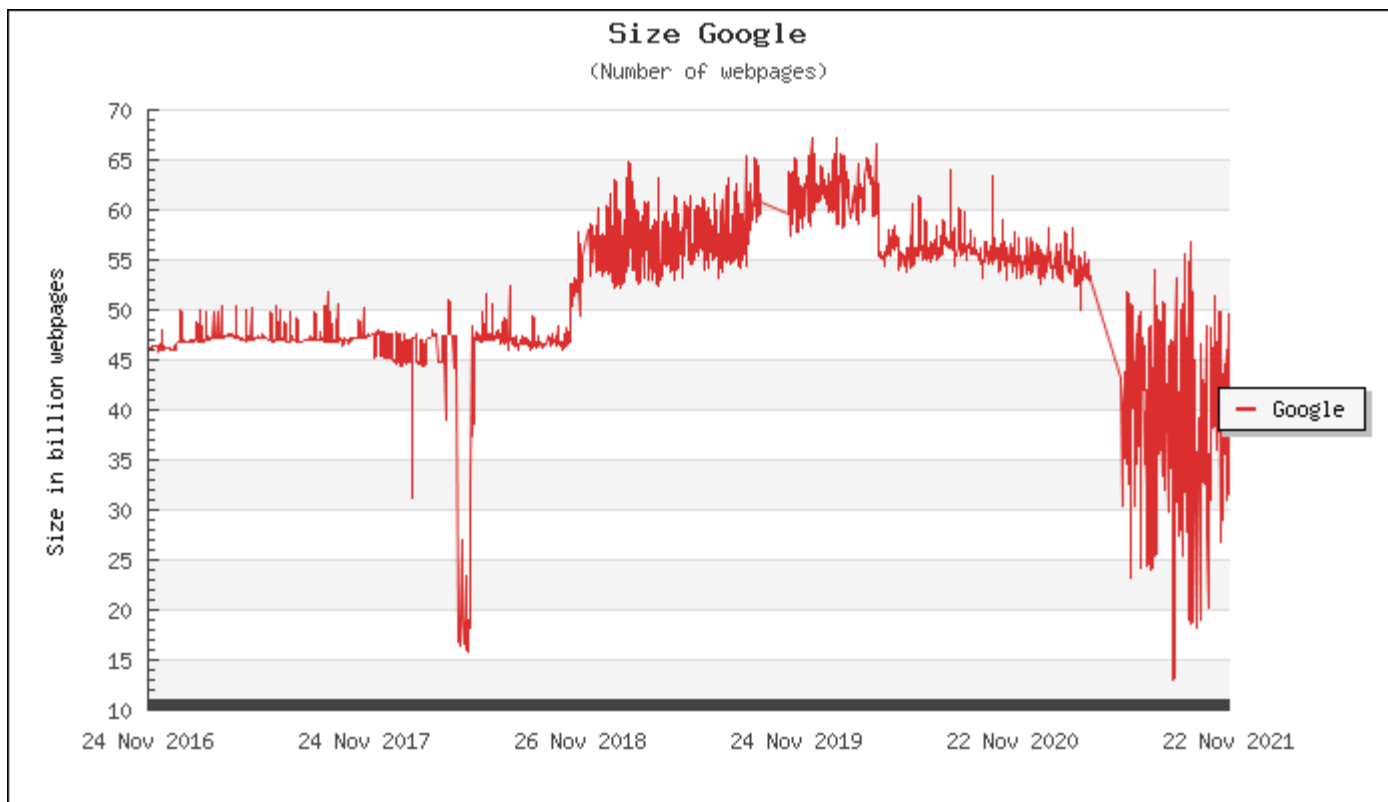
Индексиране в WWW



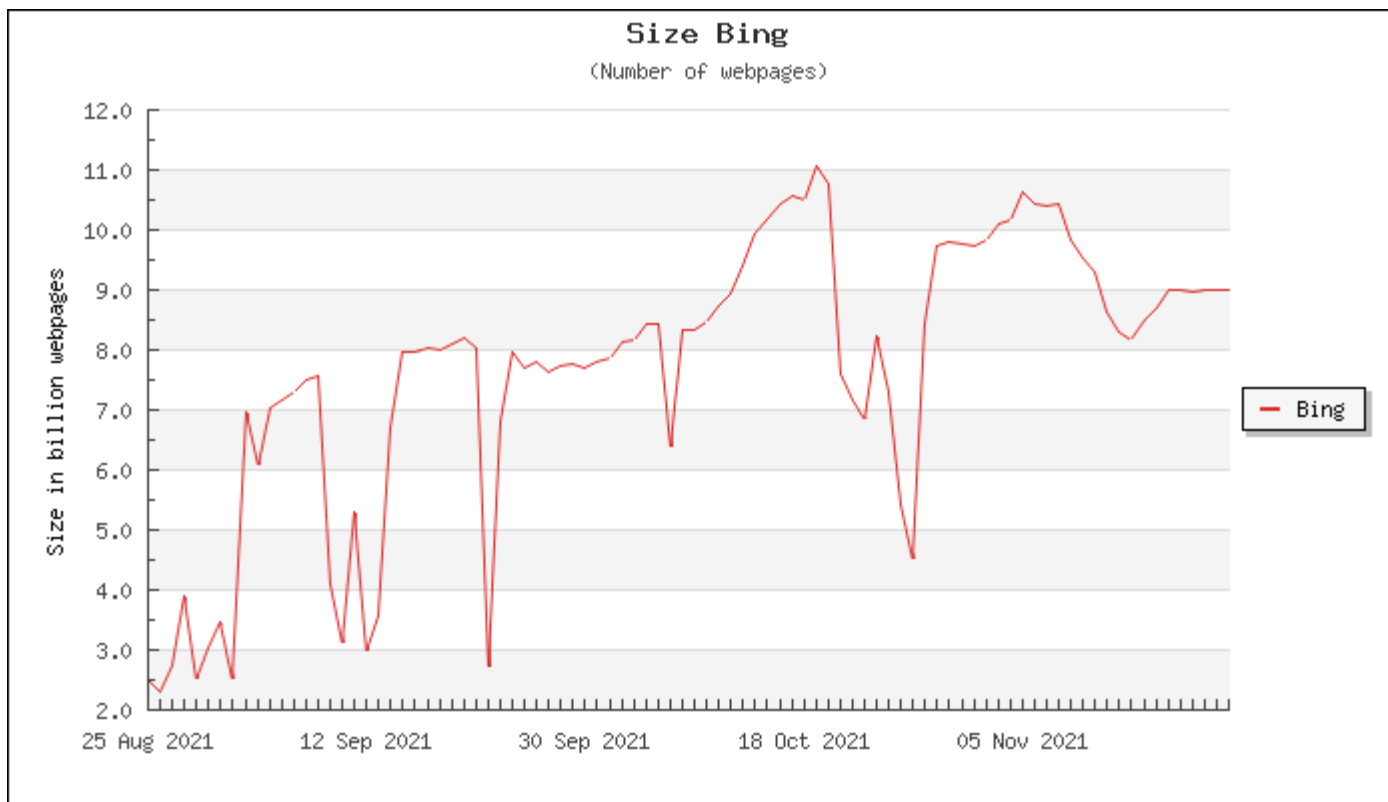
Индексиране в WWW



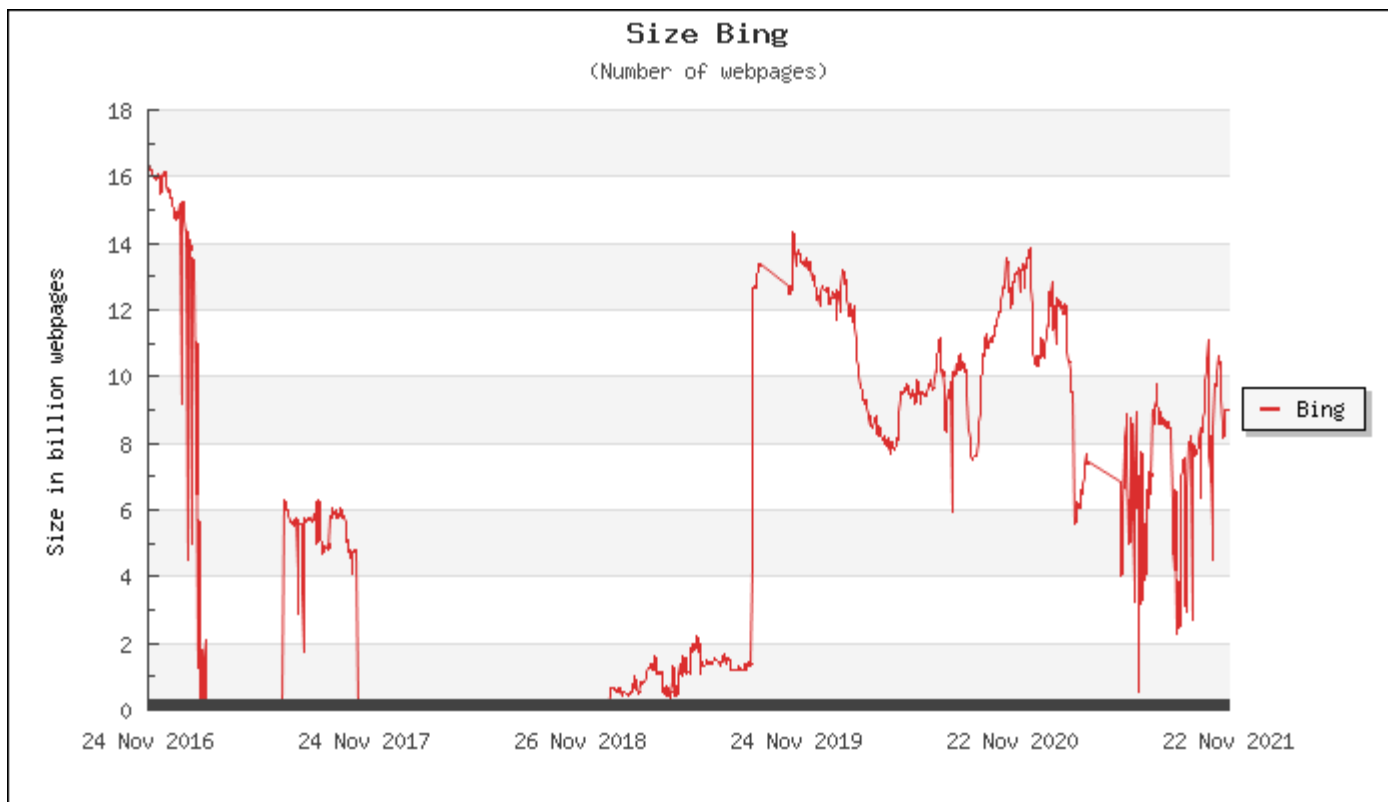
Индексиране в WWW



Индексиране в WWW



Индексиране в WWW



Разпределено изграждане на индексите

- Позволява обработка на огромни обеми данни (Web).
- Използване на голям брой икономически изгодни сървъри вместо скъпи машини.
- Разпределено средство за индексирание – ***MapReduce***.

Пример

- Даден е голям текстов файл, съдържащ данни за картови транзакции:
 - Всеки ред от файла съдържа номер на кредитна карта и обем валута;
 - Да се определи броят на уникалните номера на кредитни карти.

Пример – решение 1

Ако файлът е с малък обем:

1. Чете се всеки ред.
2. Парсва се номерът на кредитната карта.
3. Съхранява се номерът в хеш-таблица.

В края на обработката хеш-таблицата съдържа по един запис за всяка кредитна карта. Броят на редовете в таблицата е решението.

➤ При голям обем на файла, хеш-таблицата е прекалено голяма да се поддържа в паметта.

Пример – решение 2

Ако транзакциите във файла са подредени по номера на картата:

1. Чете се всеки ред.
2. Парсва се номерът на кредитната карта.
3. Ако намереният номер е различен от този в предния ред, увеличава се брояч.

В края на обработката броячът съдържа броят на уникалните кредитни карти във файла.

➤ Не е необходима хеш-таблица.

Пример – разпределено решение

Файлът съдържа неопределено множество от транзакции:

1. Разделя се файла на малки пакети от транзакции.
2. Препброяват се уникалните номера на карти във всяка част.

➤ Как да се комбинират резултатите?

➤ Не може да се събере броя на кредитните карти от всяка част.

✓ Поддържане на списък от уникални номера на карти във всяка част и групиране на тези списъци заедно. Размерът на финалния списък е решението.

Пример – ново решение

Транзакциите са разделени в пакети, така, че в един пакет са всички транзакции с еднакви номера на карти:

1. Всеки пакет се обработва индивидуално и се определя броя на транзакциите в него.
2. В края на обработката се сумират броячите на отделните пакети.

✓ Няма необходимост от групиране на резултатите.

Решението

Подходящо групиране на данните

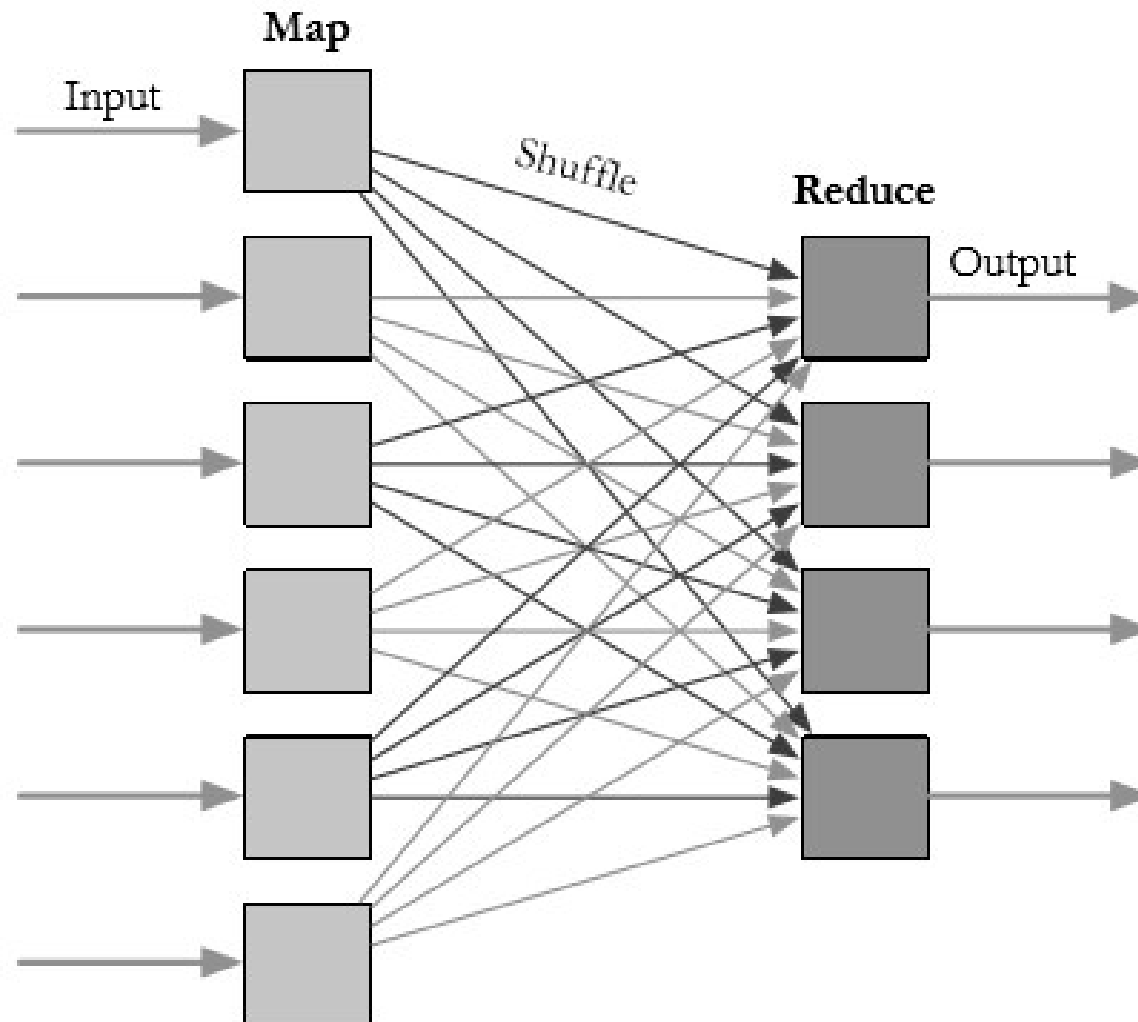
MapReduce

Разпределена програмна среда (framework).

Map – трансформира списък от позиции в друг списък с позиции със същата дължина.

Reduce – трансформира списък от позиции в единична позиция.

MapReduce



MapReduce

Данни – множество от записи.

1. Подават се на Mapper компонента.
2. *Mapper* трансформира записите в двойки <ключ:стойност>.
3. Следва операция “Разпределяне” (Shuffle). Чрез хеш-функция двойките с еднакъв ключ се групират за една и съща машина.
4. *Reduce* компонента обработва записите по пакети (всички двойки с еднакъв ключ се обработват наведнъж).

MapReduce

Операциите Map и Reduce са idempotent-ни:

- Многократното прилагане върху един и същи входни данни резултира в едни и същи изходни – fault tolerance.

MapReduce при индекси

```
procedure MAPDOCUMENTSTOPOSTINGS(input)
  while not input.done() do
    document ← input.next()
    number ← document.number
    position ← 0
    tokens ← Parse(document)
    for each word  $w$  in tokens do
      Emit( $w$ ,  $number:position$ )
      position = position + 1
    end for
  end while
end procedure
```

```
procedure REDUCEPOSTINGSTOLISTS(key, values)
  word ← key
  WriteWord(word)
  while not input.done() do
    EncodePosting(values.next())
  end while
end procedure
```

Въпроси?