MSTCZJU GIP
Series No. 0, 4th Event

Introduction to Information Retrieval
Chapter 5, Index Compression

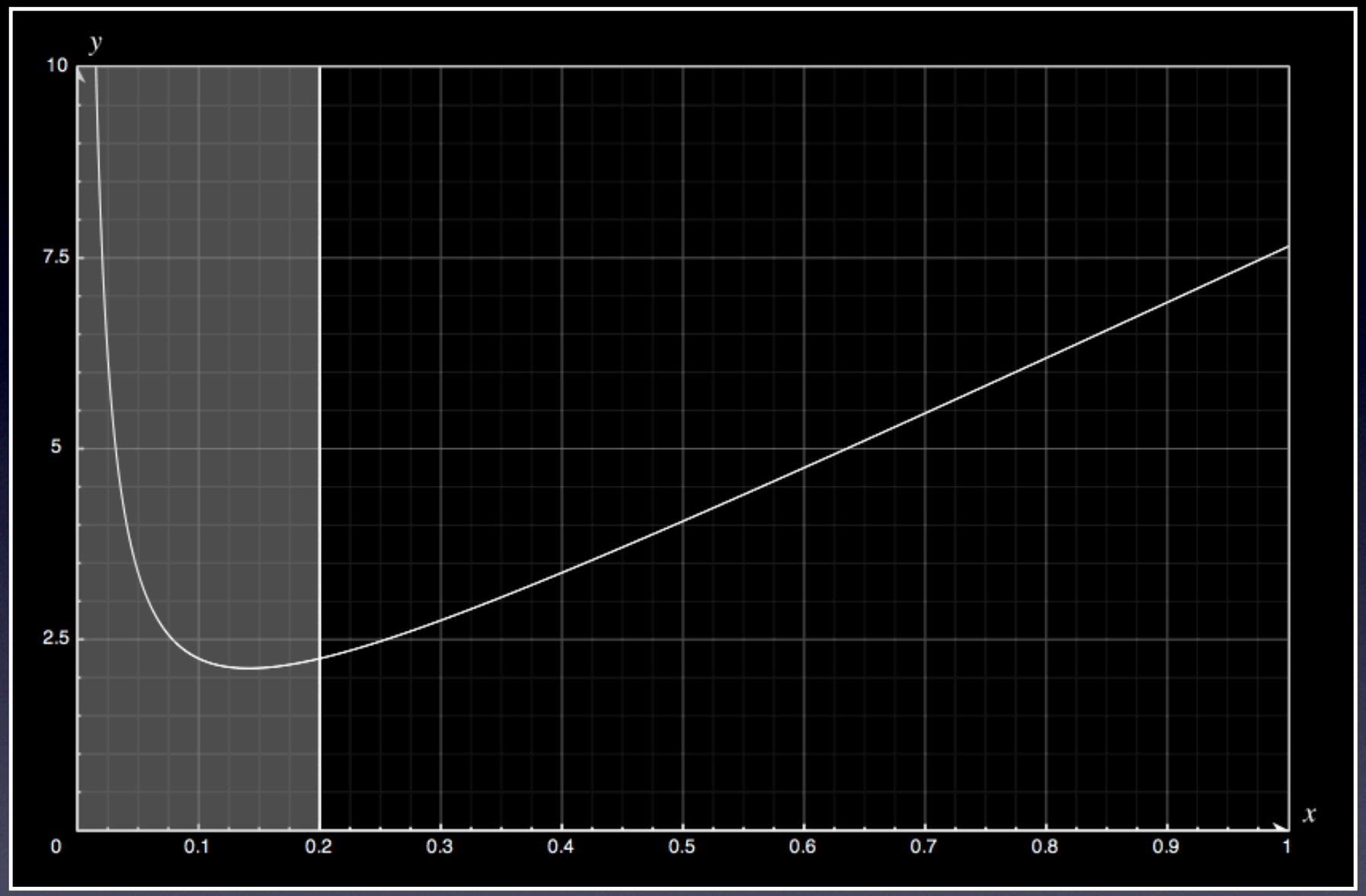
itsuhane@MSTCZJU
Nov. 22, 2008

Index Compression

ItIR C5

Benefit

- Less Space Cost
- Caching
- Less I/O



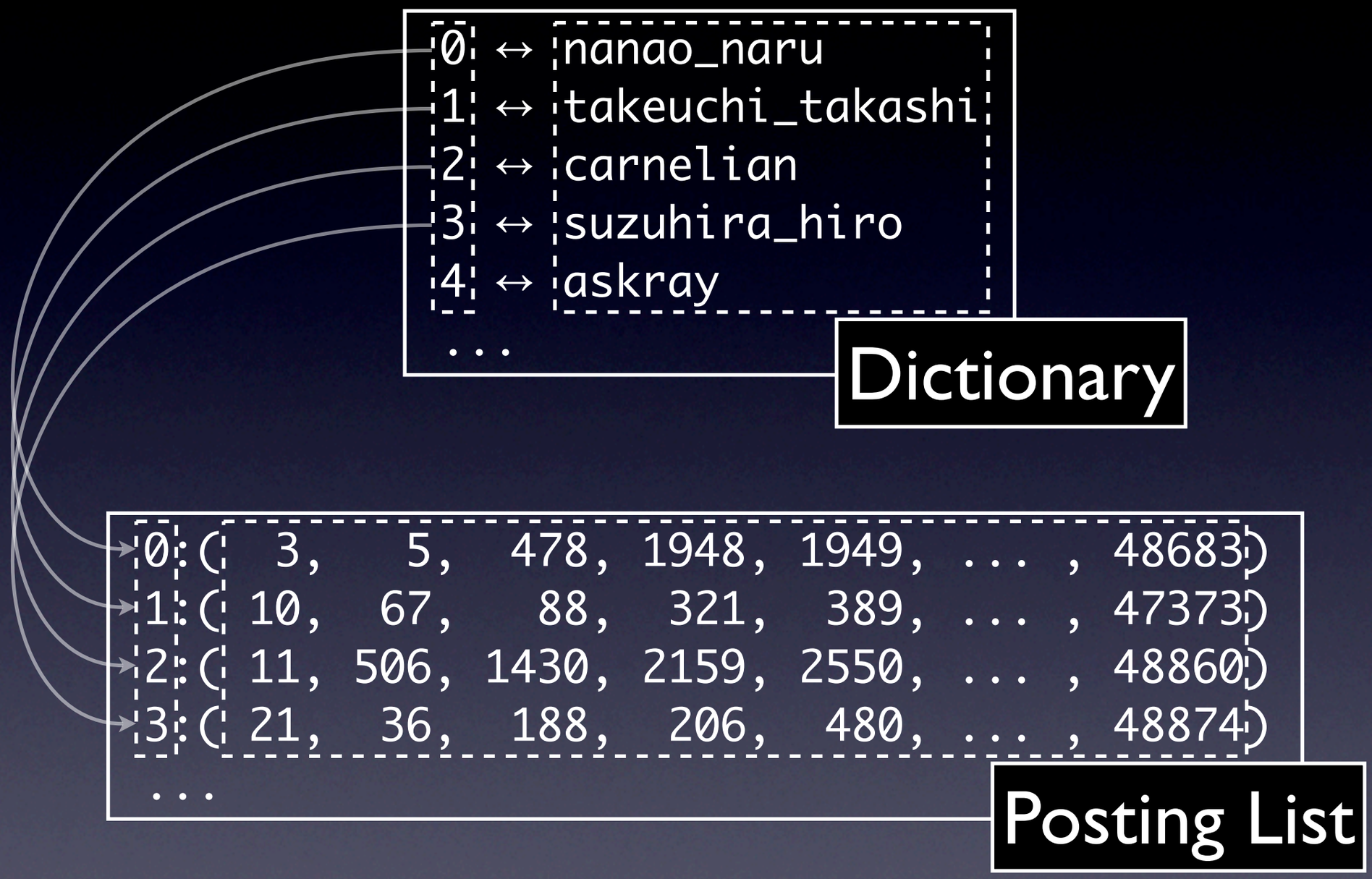
Less I/O

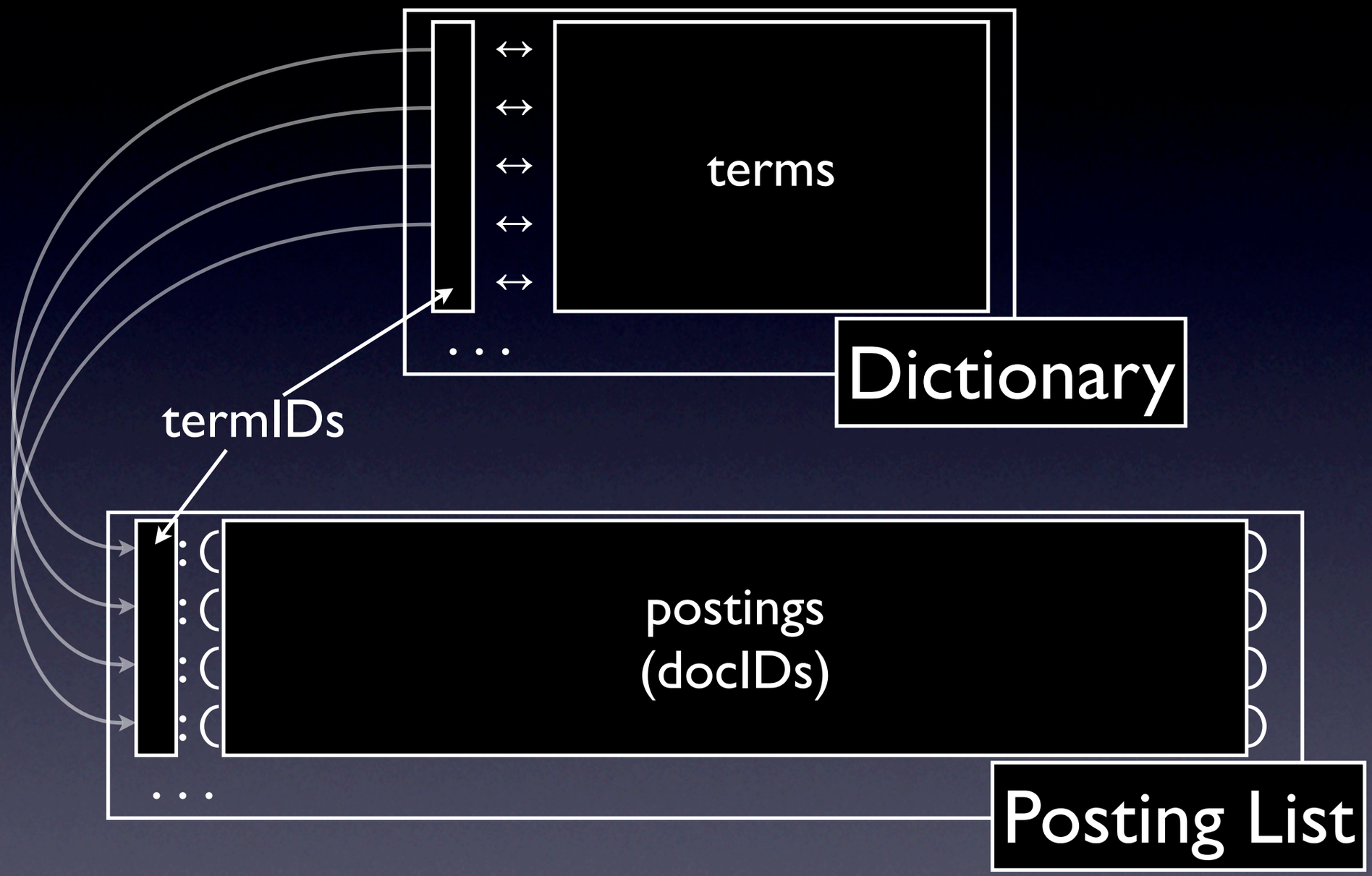
Motivator

- Caching - Prime Motivator
- Less I/O
- Less Space Cost

Terminology

- term, termID
- posting, docID
- dictionary
- posting list, postings


0 ↔ nanao_naru
1 ↔ takeuchi_takashi
2 ↔ carnelian
3 ↔ suzuhira_hiro
4 ↔ askray
...
Dictionary
0: (3, 5, 478, 1948, 1949, ... , 48683)
1: (10, 67, 88, 321, 389, ... , 47373)
2: (11, 506, 1430, 2159, 2550, ... , 48860)
3: (21, 36, 188, 206, 480, ... , 48874)
...
Posting List


terms
Dictionary
termIDs
postings
(docIDs)
Posting List

Los?

- Lossy - Discards some Information
  - MP3
- Hybrid - Preserves some, Redirect some
  - WavPack
- Lossless - Preserves all Information
  - Monkey's Audio

Statistical Properties

How many terms?

- sometimes: certain size.
- OED: more than 600,000 words.
- Actually: ?

Actually

- OED discarded some meaningful words!
  - PostgreSQL, setrlimit, BD5HLI, itsuhane
  - toold, tooold, toooold, ...
  - A lucky word: google
  - A disgusting word: baidu

How many terms?

- sometimes: certain size.
- OED: more than 600,000 words.
- Actually: MANY MANY!

No. of Terms

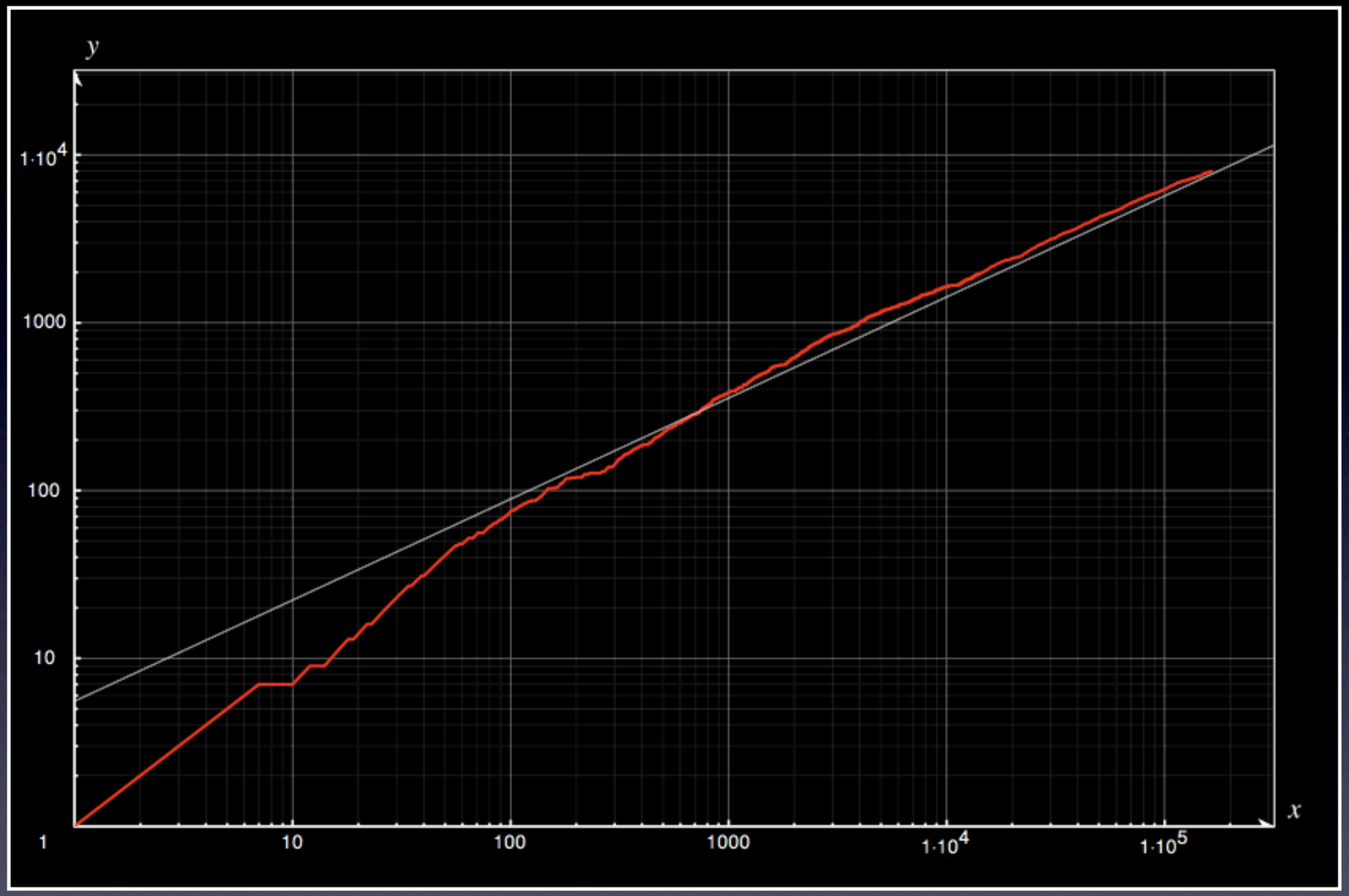
- As collection size growing, (known/concerned)vocabulary size grows.

$$M = kT^b$$

No. of Terms

No. of Tokens

Heaps' Law



Heaps' Law

Heaps' Law

- Dictionary will grow.
  - Space will not enough.
    - Compress is necessary.
      - There is no perfect hash!

Dist. of Terms

- Help characterize the compress algorithm

$$cf_i \propto \frac{1}{i}$$

Zipf's Law

Zipf's Law

- Frequency of appearance
  - Convert postings to gaps
- cf[i] is not a constant

Dict. Compression

- Dictionary is growing...
- Dictionary is frequently used...

Simplest Scheme

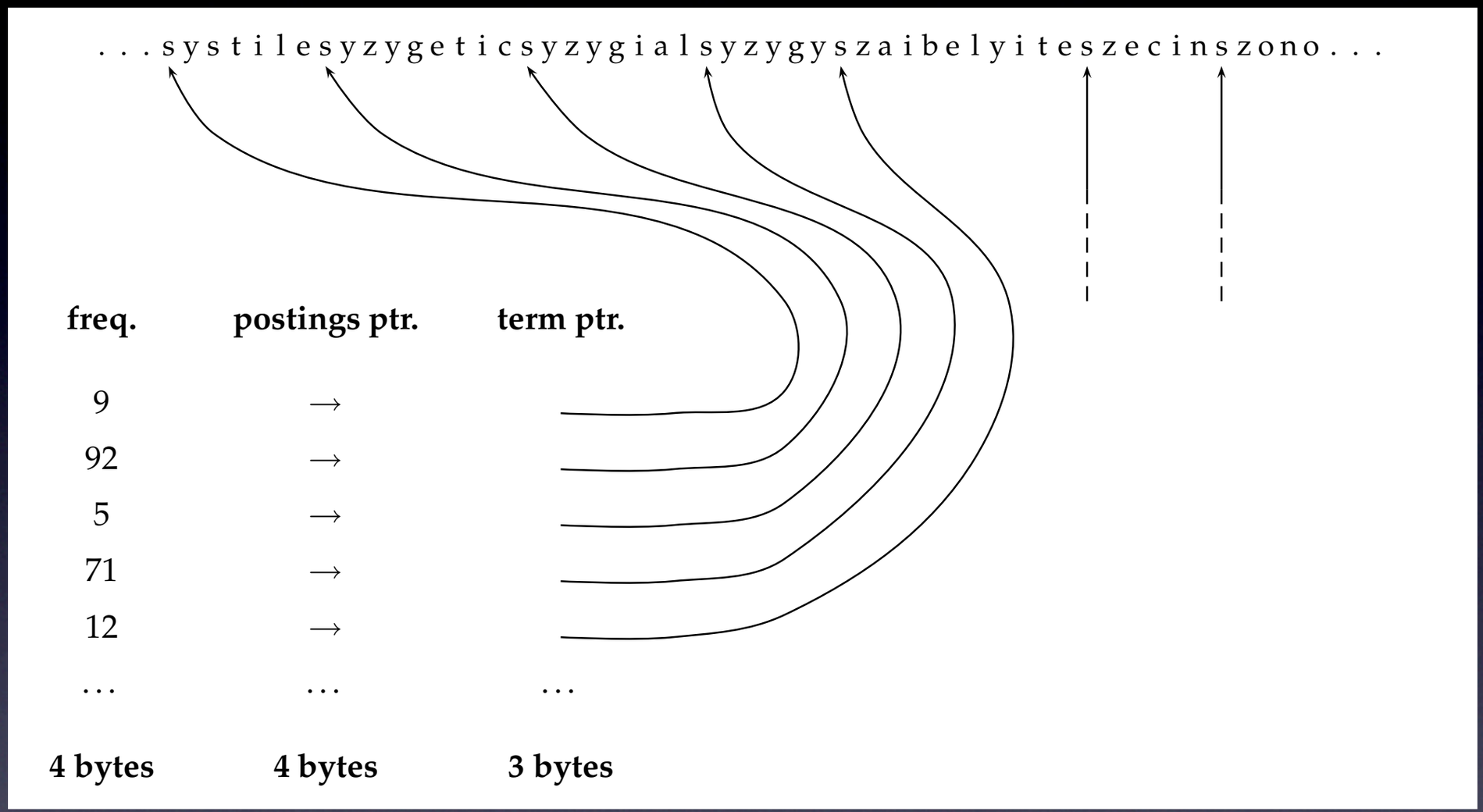
- Sorted fixed-width entries
  - fast for binary search
  - but waste lot of space
  - insert or remove terms is expensive

Simplest Scheme

- How to decide the width?
  - 8? Too short:
    - kumo_no_mukou_yakusoku_no_basho
  - 32? Too long:
    - ARIA

dictionary-as-a-string

- Store all the terms as a single string, without spacing for alignment.
- Pointers indicates term start and end.



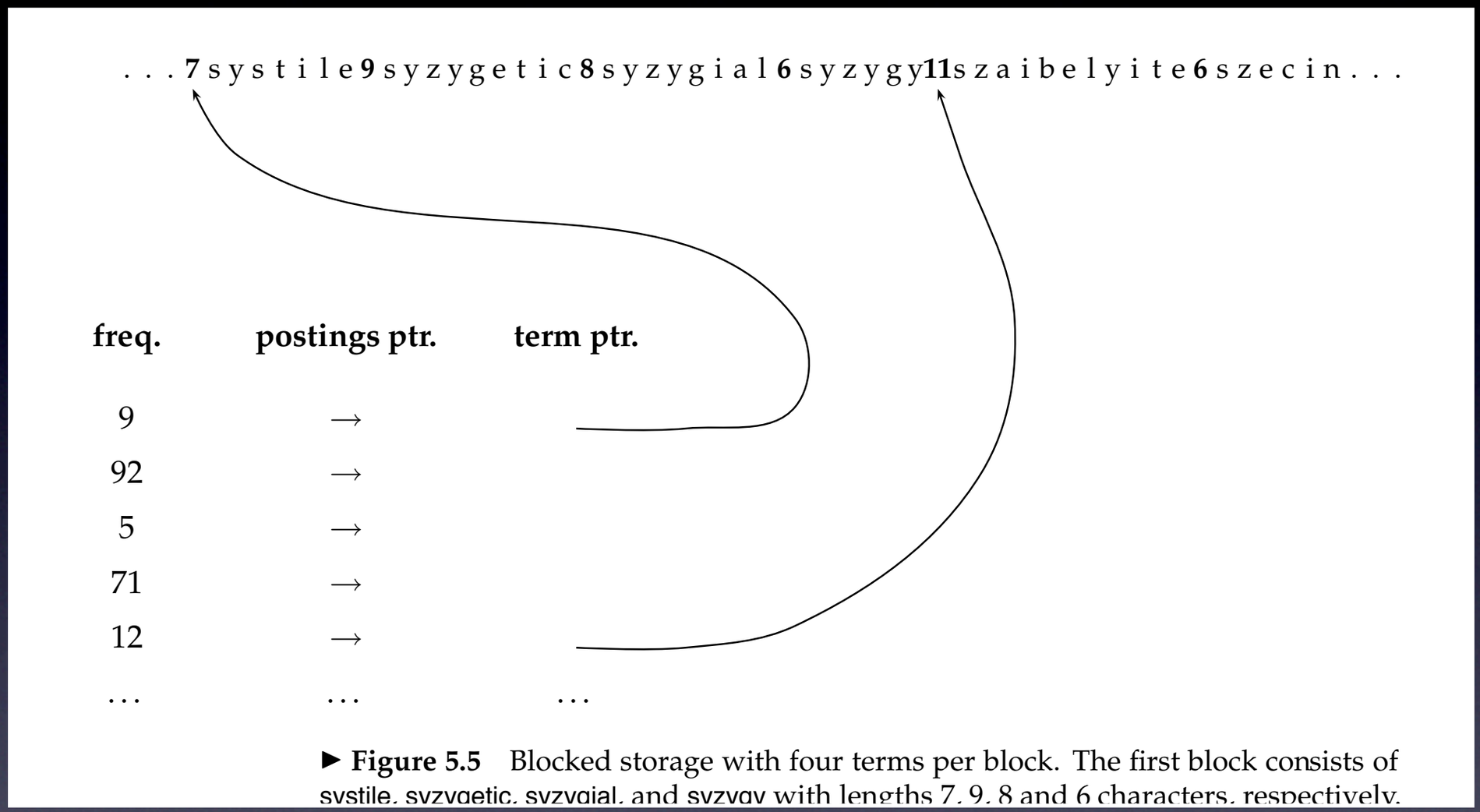
dictionary-as-a-string

dictionary-as-a-string

- More natural in C-like programming.
- Use less space.

Blocked storage

- A variant of dictionary-as-a-string
- Group terms into blocks, each term is represented in Pascal-strings
- A pointer for each block, but not term.



► **Figure 5.5** Blocked storage with four terms per block. The first block consists of systile, syzygetic, syzygial, and syzygy with lengths 7, 9, 8 and 6 characters, respectively.

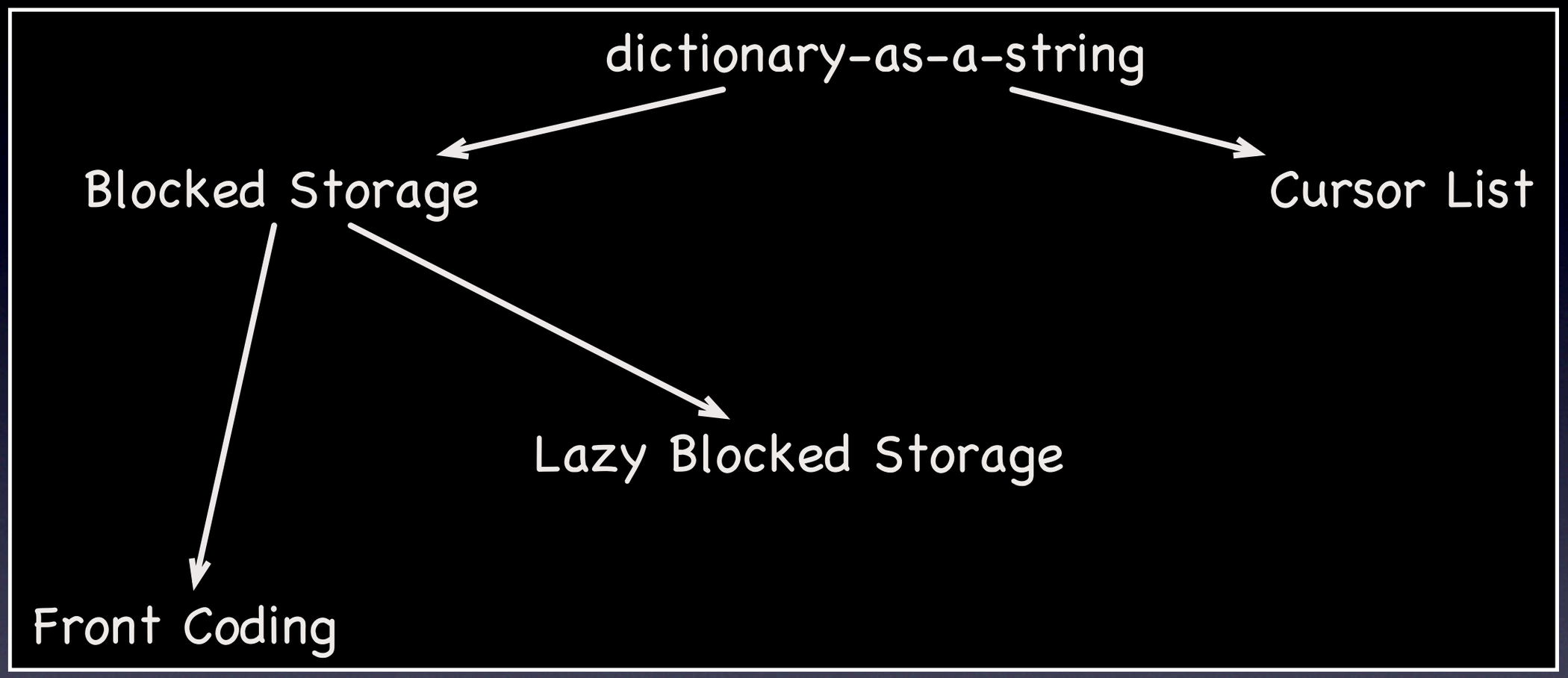
Blocked storage

Blocked storage

- Further compact.
- Need more time to lookup a term.

Front Coding

- (Blocked storage).upgrade();



More Considerations

Postings Compression

- Just because they are huge...

Thinking in Gaps

- The key point is to store postings by recording the gaps, instead of using the docID it self.
- For a “famous” term, gaps between two occurrences is a relatively small number.

Thinking in Gaps

```
0:(    3,    5,  478, 1948, 1949, ... , 48683)
1:(   10,   67,   88,  321,  389, ... , 47373)
2:(   11,  506, 1430, 2159, 2550, ... , 48860)
3:(   21,   36,  188,  206,  480, ... , 48874)
...
```

```
0:(    3,    2,  473, 1470,    1, ... )
1:(   10,   57,   21,  233,   68, ... )
2:(   11,  495,  924,  729,  391, ... )
3:(   21,   15,  152,   18,  274, ... )
...
```

Thinking in Gaps

```
...
19:( ... , 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 15, 7, 15)
...
```

Most frequently appeared

Thinking in Gaps

- The more times a code appears, it is more expected to appear again.

$$H(P) = -\sum_{x \in X} P(x) \log_2 P(x)$$

Entropy

$$
\begin{aligned}
H(P) &= -\sum_{x \in X} P(x) \log_2 P(x) \\
&= \sum_{x \in X} (-\log_2 P(x)) P(x) \\
&= \sum_{x \in X} L(x) P(x) \\
&= E(L)
\end{aligned}
$$

Entropy

Entropy

- Information Theory suggest that if we could encode X into a code space such that each code for x is exactly of length L(x), then the total encoded data length is the minimal length, for which, the average code length is expressed by H(P).

$$L(x) = -\log_2 P(x)$$

Entropy

- From L(x), we could derive out a approximate solution: variable encoding.
- Frequent items are encoded into relatively shorter codes.

Thinking in Gaps

- Relatively frequent gaps have smaller values, and they should be encoded into shorter codes.

Thinking in Gaps

- Rice
- Variable Byte Code
- Huffman Code
- γ-code

Rice

- $0 \leftrightarrow 0$
- $1 \leftrightarrow 10$
- $2 \leftrightarrow 110$
- $3 \leftrightarrow 1110$
- ...

Rice

- Larger gaps?
  - 41732 ↔ 111...111...111...111...1110
- Fail...

Variable Byte Code

- Byte-scale compress
- Use leftmost bit as continue sign

Variable Byte Code

00001101

00001100

10110001

00011010001100011000

Huffman Code

- Need to know the frequency distribution.
- Too complex to decompress on-the-air.

γ-code

- length(unary) + offset(biased)

γ-code

x	length	offset	γ-code
0	-	-	-
1	0	/	0
2	10	0	10 0
5	110	01	110 01
10	1110	010	1110 010
29	11110	1011	11110 1011
60	111110	11100	111110 11100
99	1111110	100011	1111110 100011

γ-code

- Calculations?

Pitfall

- Boolean IR needs to merge postings by docIDs, but not gaps...

- If the time consumed for restoring gaps into docIDs is not important. Why not use RLE?

Discussion Time

MSTCZJU GIP
Series No. 0, 4th Event

Introduction to Information Retrieval
Chapter 5, Index Compression

itsuhane@MSTCZJU
Nov. 22, 2008