



# Digital Libraries

Jun.-Prof. Dr. Tobias Schreck  
WS 2014/15

# Planned Lecture Timetable

Date	Lecture
20.10.2014	Introduction
<b>27.10.2014</b>	<b>Text Documents</b>
03.11.2014	Images of Pages
10.11.2014	Knowledge Representation
17.11.2014	Collections, Digitization
24.11.2014	Preservation
01.12.2014	Survey of Digital Libraries
08.12.2014	Multimedia Retrieval
15.12.2014	Research Data
22.12.2014	Visual Search and Exploration
12.01.2014	Personalization
19.01.2014	Evaluation
26.01.2014	Economic Aspects
02.02.2014	Current Research Topics
09.02.2015	Exam (take one)

Date	Exercise
07.11.2014	t.b.a.
21.11.2014	t.b.a.
05.12.2014	t.b.a.
19.12.2014	t.b.a.
09.01.2015	t.b.a.
23.01.2015	t.b.a.
06.02.2015	t.b.a.
All exercises are from 13:30-15:00 in G227	



## 2. Text Documents

# Learning Goals

- Understand important role of textual documents for information representation, as compared to image-based representation of text.
- Distinguish between text encoding, text structure markup, text presentation; familiarize with selected standards.
- Understand basic text searching methods: Boyer-Moore algorithm, Hash tables, Tries; be able to discuss main advantages and disadvantages.

## 2. Text Documents

2.1 Text Documents in Digital Libraries

2.2 Markup, Page Description & Style Sheets

2.3 Searching Text

2.4 Language processing

2.5 Summary

## 2.1 Text documents in DLs

- Text materials have a special place in digital libraries
- Text in digital format may come from different sources:
  - recently - generated documents are available in digital formats (e-books, e-journals, digital image, video, etc.)
  - additional efforts have been made to convert traditional documents into digital formats as either a *sequence of characters* or an *image of the page*

## 2.1 Text documents in DLs

- Two basic methodologies that support digital libraries:
  - The ability of the computer to manipulate *images*
    - Scanning
    - Formatting
    - Displaying
    - Indexing
  - The ability of the computer to manipulate *text (sequence of characters)*
    - Formatting
    - Searching
    - Language processing
    - Document conversion

# 2.1 Text documents in DLs

## Code format

### ➤ **ASCII** (7-bit, 1-bit unused)

won out over alternatives for language that use only the 26 letters of the Latin alphabet

<div><div><div>b<sub>7</sub></div><div>b<sub>6</sub></div><div>b<sub>5</sub></div><div>b<sub>4</sub></div><div>b<sub>3</sub></div><div>b<sub>2</sub></div><div>b<sub>1</sub></div></div><div>Bits</div></div> <div><div>Column</div><div>Row</div></div>					0	0	0	0	1	0	1	1	0	1	1	0	1
					0	1	2	3	4	5	6	7					
0	0	0	0	0	0	NUL	DLE	SP	0	@	P	`	p				
0	0	0	1	1	1	SOH	DC1	!	1	A	Q	a	q				
0	0	1	0	2	2	STX	DC2	"	2	B	R	b	r				
0	0	1	1	3	3	ETX	DC3	#	3	C	S	c	s				
0	1	0	0	4	4	EOT	DC4	\$	4	D	T	d	t				
0	1	0	1	5	5	ENQ	NAK	%	5	E	U	e	u				
0	1	1	0	6	6	ACK	SYN	&	6	F	V	f	v				
0	1	1	1	7	7	BEL	ETB	'	7	G	W	g	w				
1	0	0	0	8	8	BS	CAN	(	8	H	X	h	x				
1	0	0	1	9	9	HT	EM	)	9	I	Y	i	y				
1	0	1	0	10	10	LF	SUB	*	:	J	Z	j	z				
1	0	1	1	11	11	VT	ESC	+	;	K	[	k	{				
1	1	0	0	12	12	FF	FC	,	<	L	\	l					
1	1	0	1	13	13	CR	GS	-	=	M	]	m	}				
1	1	1	0	14	14	SO	RS	.	>	N	^	n	~				
1	1	1	1	15	15	SI	US	/	?	O	_	o	DEL				

### ➤ **Unicode** (16-bit)

for languages with additional symbols, cover the characters for all major languages



## 2. Text Documents

2.1 Text Documents in Digital Libraries

**2.2 Markup, Page Description & Style Sheets**

2.3 Searching Text

2.4 Language processing

2.5 Summary

## 2.2 Markup, Page Description and Style Sheets

- Methods for storing textual materials must represent two different aspects of a document

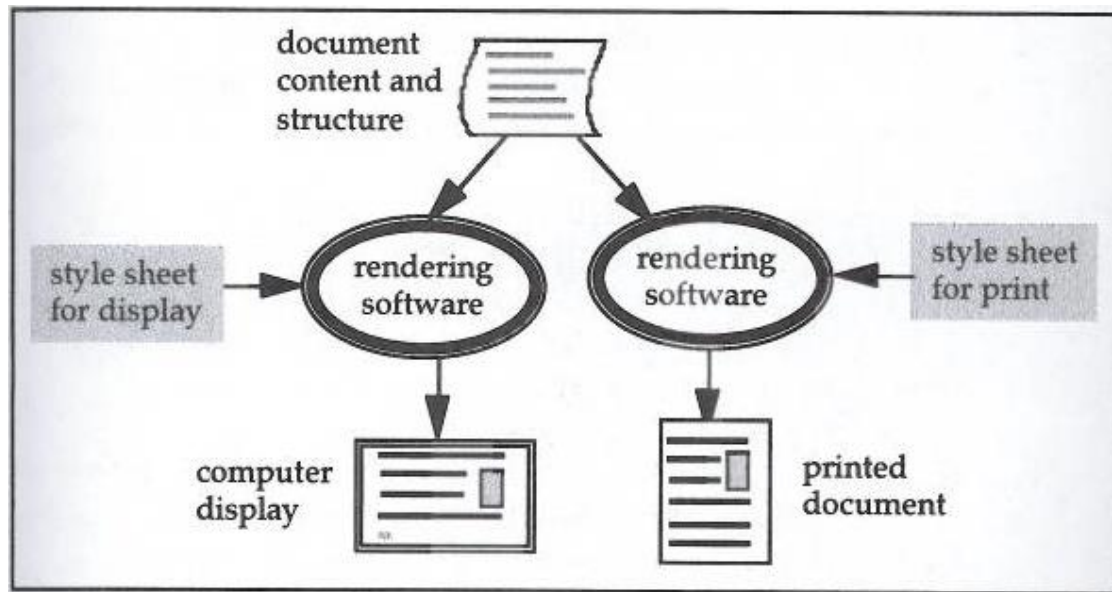
**Structure** – describes the division of text into elements, identify parts of the documents that are emphasized, materials placed in table or footnotes, and everything that relates one to another  
Often represented by a *markup specification* (e.g. SGML, HTML, and MARC)

**Appearance** – how the text document looks when displayed on screen or printed on paper  
Often defined by a *style sheet*

*Page description languages* (e.g. Tex, PostScript, and PDF) encode both structure and appearance

## 2.2 Markup, Page Description and Style Sheets

**Good practice is to separate content from format:** typically, markup is used to define the semantic content and structure, and a style sheet is used to define the visual layout.



Alternative rendering of a single document

## 2.2 Markup, Page Description and Style Sheets

### 2.2.1 Markup

- SGML
- HTML
- MARC

### 2.2.2 Style Sheets

### 2.2.3 Page Description

## **Standard Generalized Markup Language**

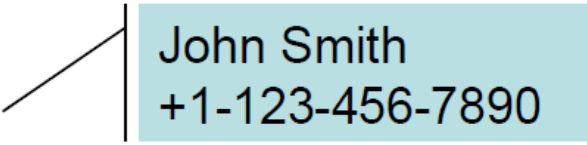
- International Organization for Standardization (ISO) Standard
- “Meta” language used for defining markup languages
- A markup language defined using SGML has a specific vocabulary and a declared syntax.
- Provides exchange of information
  - Any level of complexity
  - Understood by people, software, storage (database), presentation systems
  - Without regard to manufacturer
  - Authority of international standards

# SGML – structure and contents

## ■ SGML encodes two kinds of content

### – Content (data)

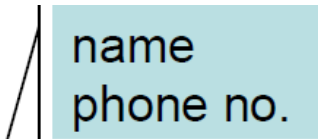
- Data about the subject



John Smith  
+1-123-456-7890

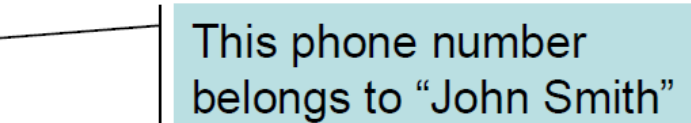
### – Information about the content (markup)

- Information about what the content represents



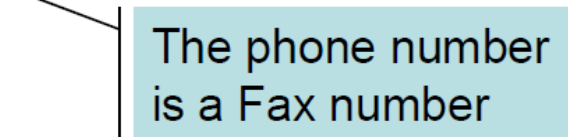
name  
phone no.

- Hierarchical structure



This phone number  
belongs to “John Smith”

- Modifiers to the content



The phone number  
is a Fax number

# How SGML works

- Elements

- The building block of SGML
- Each element has a:

- start-tag
- content
- end-tag

`<name>John Smith </name>`

`<phone>+1-123-456-7890</phone>`

- Tags may contain other information:

`<authortype = pseudonym>`

`<figure graphicfile = "fig1.gif">`

# How SGML works

- Nested elements create structure
- Attributes
  - Attributes are name/value pairs that express more information about an element
  - May modify the content in some way
  - May be used as pointers to other places

```
<customer id="123">  
  <name>  
    <first>John</first>  
    <last>Smith</last>  
  </name>  
  <phone type="fax" countryCode="001">  
    123-456-7890</phone>  
  <referredBy customerId="456"/>  
</customer>
```



# Markup Languages

- SGML
- **HTML**
- MARC

## Hyper Text Markup Language

- SGML instance
- The markup language used on the web
- Syntactically very similar to SGML
- Supports hypertext links

`<A HREF = “http://www.w3.org/TR/html4/strict.dtd”>`

# HyperText Markup Language

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
    "http://www.w3.org/TR/html4/strict.dtd">
<HTML>
  <HEAD>
    <TITLE>My first HTML document
    </TITLE>
  </HEAD>
  <BODY>
    <P>Hello world!
  </BODY>
</HTML>
```

# Markup Languages

- SGML
- HTML
- **MARC**

## **MA**chine-**R**eadable **C**atalog

- Started in the late 60s and developed by the Library of Congress to facilitate catalog sharing
- Provides a machine readable representation of a catalog card
- Based on a system of numbers, letters and symbols to identify fields in the record
- Many “national” versions (UKMARC, CANMARC, AUSMARC, DANMARC, ANNAMARC, INTERMARC, etc.), UNIMAR (Universal MARC) as standard format for exchange of information
  - e.g. USMARC to UNIMARC to AUSMARC
- **Does not** support many features needed for full documents (equations, tables, footnotes...)

## Catalog Card

- Name: Arnosky, Jim.
- Title proper: Raccoons and ripe corn.
- Statement of responsibility: Jim Arnosky
- Edition statement: 1st ed.
- Place of publication: New York
- Name of publisher: Lothrop, Lee & Shepard Books
- Date of publication: c1987
- Pagination: 25 p.
- Illustrative matter: col. ill.
- Size: 26 cm
- Summary: Hungry raccoons feast at night in a field of ripe corn
- Topical subject: Raccoons
- Local call number: 599.74 ARN
- Local barcode number: 8009
- Local price: \$15.00

## MARC representation

"SIGNPOSTS"	DATA
1001#\$a	Arnosky, Jim.
24510\$a	Raccoons and ripe corn /
\$c	Jim Arnosky.
250##\$a	1st ed.
260##\$a	New York:
\$b	Lothrop, Lee & Shepard Books,
\$c	c1987.
300##\$a	25 p. :
\$b	col. ill. ;
\$c	26 cm.
520##\$a	Hungry raccoons feast at night in a field of ripe corn.
650#1\$a	Raccoons.
900##\$a	599.74 ARN
901##\$a	8009
903##\$a	\$15.00

See also definition by Library of Congress, <http://www.loc.gov/marc/>

# MARC Structure

- Each bibliographic record is divided into **fields**. These fields are subdivided into one or more "**subfields**."
- Each field is associated with a 3-digit number called a "**tag**". A tag identifies the field (i.e. the kind of data ) that follows.
- Two character positions follow each tag (with the exception of Fields 001 through 009). One or both of these character positions may be used for "**indicators**".
- Some indicators are empty. The empty indicators are marked by a "placeholder".
- Most fields contain several related pieces of data, each subfield is preceded by a subfield code. Subfield codes are one lowercase letter (occasionally a number) preceded by a delimiter.

# Example of MARC

100 1# \$a Gregory, Ruth, W. \$q (Ruth Wilhelme) \$d 1910-

100 Main entry -- Personal name -- (primary author)

Indicator 1: Type of personal name entry element

- 0 -- Forename
- 1 -- Surname (this is the most common form)
- 3 -- Family name

Indicator 2 undefined.

Subfields used most often:

- \$a -- Personal name
- \$b -- Numeration
- \$c -- Titles and other words associated with a name (R)
- \$q -- Fuller form of name
- \$d-- Dates associated with a name ( generally, year of birth)



## 2.2.2 Style Sheets

- Markup
- **Style Sheets**
- Page Description

## 2.2.2 Style Sheets

- Style sheets are used to store and apply formatting to text (*e.g., font and color for 2<sup>nd</sup> level headings...*)
- A style sheet can be very complex yet still fail to be satisfactory for complex documents (*e.g. equations*)
- With HTML there is no formal concept of style sheets. The markup combines the structural elements with formatting tags. Individual browsers interpret and adapt to the computer display in use.
- Methods (e.g. CSS Cascading Style Sheet and XSL Extensible Style Language) have been developed to control the appearance of documents

# CSS Example

- Inline - affects individual HTML tag

```
<html>
```

```
...
```

```
<body>
```

```
...
```

```
<p style="font-family: Arial, sans-  
serif; ">some text</p>
```

```
</body>
```

```
</html>
```

# CSS Example

- Embedded - affects individual document

```
styles.css
```

```
p{
```

```
    font-family: Arial, Sans-serif;
```

```
}
```

```
<html>
```

```
    <head>
```

```
        ...
```

```
        <link href="styles.css"
              rel="stylesheet" type="text/css">
```

```
    </head>
```

```
    <body>
```

```
        ...
```

```
        <p>some text</p>
```

```
    </body>
```

```
</html>
```

# CSS - Advantages and Disadvantages

- Advantages:
  - Greater typography and page layout controls
  - Easier site maintenance
  - The style sheet information can be stored separate from your HTML document, and shared among many HTML files.  
Change one style sheet file, and the appearance of the entire site is updated.
- Disadvantages
  - Browser compatibility must be ensured

## 2.2.3 Page Description

- Markup
- Style Sheets
- **Page Description**

## 2.2.3 Page Description Languages

- Specify the appearance of a document directly without a structural markup
- Render textual materials with the same graphical quality and precision
- Most well-known page description languages

**PostScript** – programming language to create graphical output for printing

**PDF** –format for storing page images in a portable format that is independent of any particular computer

## 2. Text Documents

2.1 Text Documents in Digital Libraries

2.2 Markup, Page Description & Style Sheets

**2.3 Searching Text**

2.4 Language processing

2.5 Summary



## 2.3 Searching Text

It is vital for digital libraries to implement effective **search engines** such that the reader can find what is wanted from a large collection of documents

### Questions to ask while designing a search engine:

1. What vocabulary to use? (text itself, synonyms, indexing, or other form of content labeling?)
  2. How to handle multi-term queries? (Boolean logic?)
  3. What search algorithms to use? (binary search?)
  4. How to rank the items in the search result? (rank-order?)
- ...

## 2.3 Searching Methods

### 2.3.1 Linear Searching

search from beginning until the end

### 2.3.2 Inverted Files

index the terms, retrieve only the blocks with right matches

### 2.3.3 Hash Tables

guess the location

### 2.3.4 Data structures to improve linear searching

## 2.3.1 Linear Searching

### Linear Searching

- Search for exact string “abc”, “xyz”
- Search for string with operators - “a?c”, “a|b” - **regular expression search**

Operator	Meaning
.	Any character
A*	Match any numbers of a(s)
A+	Match one or more a(s)
A?	Either nothing or a single A
[a-d]	Matches any of a, b, c, or d
(a)	Matches expression a
a   b	Matches either a or b

## 2.3.1 Linear Search

- Standard linear search

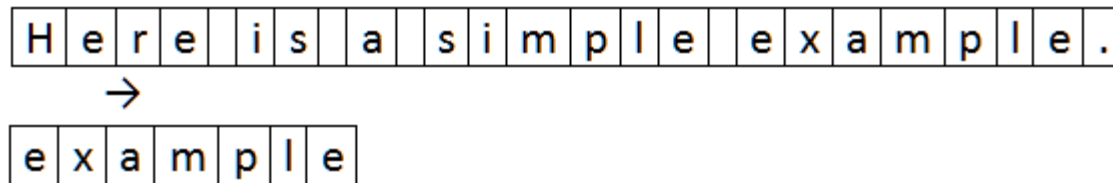
H	e	r	e		i	s		a		s	i	m	p	l	e		e	x	a	m	p	l	e	.
---	---	---	---	--	---	---	--	---	--	---	---	---	---	---	---	--	---	---	---	---	---	---	---	---

→

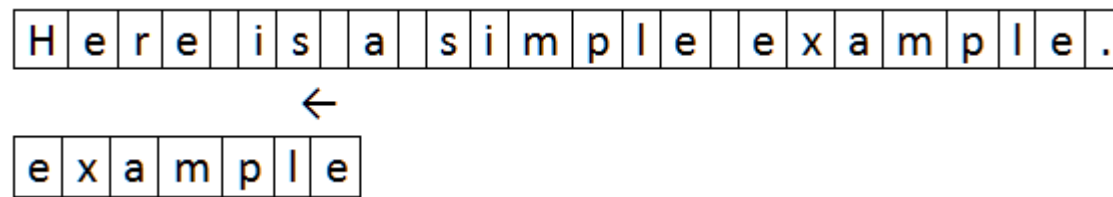
e	x	a	m	p	l	e
---	---	---	---	---	---	---

## 2.3.1 Linear Search

- Standard linear search



- The **Boyer-Moore algorithm** (BM) is the practical method of choice for exact matching. It is especially suitable if
  - the alphabet is large (as in natural language)
  - the pattern is long (as often in bio-applications)



# Boyer-Moore algorithm

## BM is based on three main ideas:

- Longer shifts are based on examining  $P$  right-to-left, in order  $P[n], P[n-1], \dots$

```

123456789012345678901234567890
T: maistuko kaima maisemaomaloma?
P: maisemaomaloma    (legend: match/mismatch)
    ←

```

# Boyer-Moore algorithm

BM is based on three main ideas:

1. Longer shifts are based on examining  $P$  right-to-left, in order  $P[n], P[n-1], \dots$

1 2 3  
123456789012345678901234567890  
T: maistuko kaima maisemaomaloma?  
P: maisemaomaloma (legend: match/mismatch)

←

2. “bad character shift rule” - avoids repeating unsuccessful comparisons against a target character

1 2 3  
123456789012345678901234567890  
T: maistuko kaima maisemaomaloma?  
P: maisemaomaloma

^

-

# Boyer-Moore algorithm

BM is based on three main ideas:

1. Longer shifts are based on examining  $P$  right-to-left, in order  $P[n], P[n-1], \dots$

1 2 3  
123456789012345678901234567890  
T: maistuko kaima maisemaomaloma?  
P: maisemaomaloma (legend: match/mismatch)



2. “bad character shift rule” - avoids repeating unsuccessful comparisons against a target character

1 2 3  
123456789012345678901234567890  
T: maistuko kaima maisemaomaloma?  
P: maisemaomaloma



3. “good suffix shift rule” - aligns only matching pattern characters against target characters already successfully matched



# Boyer-Moore algorithm - example

Text:     H   E   R   E       I   S       A       S   I   M   P   L   E       E   X   A   M   P   L   E  
Pattern:   E   X   A   M   P   L   E

Step 1:																									
Text:		H	E	R	E		I	S		A		S	I	M	P	L	E		E	X	A	M	P	L	E
		E	X	A	M	P	L	E																	
Bad Character Rule												E	X	A	M	P	L	E							
Good Suffix Rule			E	X	A	M	P	L	E																

Step 2:																									
Text:		H	E	R	E		I	S		A		S	I	M	P	L	E		E	X	A	M	P	L	E
										E	X	A	M	P	L	E									
Bad Character Rule												E	X	A	M	P	L	E							
Good Suffix Rule										E	X	A	M	P	L	E									

### Step 3:

Text:	H	E	R	E	I	S	A	S	I	M	P	L	E	E	X	A	M	P	L	E
								E	X	A	M	P	L	E						
Bad Character Rule										E	X	A	M	P	L	E				
Good Suffix Rule													E	X	A	M	P	L	E	

### Step 4:

Text:	H	E	R	E	I	S	A	S	I	M	P	L	E	E	X	A	M	P	L	E
													E	X	A	M	P	L	E	
Bad Character Rule														E	X	A	M	P	L	E
Good Suffix Rule														E	X	A	M	P	L	E

### Step 5:

Text:	H	E	R	E	I	S	A	S	I	M	P	L	E	E	X	A	M	P	L	E
														E	X	A	M	P	L	E

# Boyer-Moore algorithm - complexity

BM takes  $O(N*M)$  time, but the speed of BM comes from shifting the pattern  $P[1 \dots n]$  to the right in longer steps. Typically less than  $m$  chars (often about  $m=n$  only) of  $T[1 \dots m]$  are examined

## 2.3 Searching Methods

### 2.3.1 Linear Searching

search from beginning until the end

### 2.3.2 Inverted Files

index the terms, retrieve only the blocks with right matches

### 2.3.3 Hash Tables

guess the location

### 2.3.4 Data structures to improve linear searching

## 2.3.2 Inverted Files

- Elements to be searched extracted and ordered more readily accessible for multiple searches

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
n	o	w		i	s		t	h	e		t	i	m	e		f	o	r	
20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39
a	l	l		g	o	o	d		m	e	n		t	o		c	o	m	e
40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59
	t	o		t	h	e		a	i	d		o	f		t	h	e		p
60	61	62	63																
a	r	t	y																

## 2.3.2 Inverted Files

Word	Byte Position
now	0
is	4
the	7
time	11
for	16
all	20
good	24
men	29
to	33
come	36
to	41
the	44
aid	48
of	52
the	55
party	59

Sort the words



Word	Byte Position
aid	48
all	20
come	36
for	16
good	24
is	4
men	29
of	52
party	59
come	36
the	7
the	44
the	55
time	11
to	43
to	53

## 2.3.2 Inverted Files – documents level

Document	Text
<b>1</b>	Gold silver truck
<b>2</b>	Shipment of gold damaged in a fire
<b>3</b>	Delivery of silver arrived in a silver truck
<b>4</b>	Shipment of gold arrived in a truck

Number	Term	Times; Documents
<b>1</b>	a	<3; 2,3,4>
<b>2</b>	arrived	<2; 3,4>
<b>3</b>	damaged	<1; 2>
<b>4</b>	delivery	<1; 3>
<b>5</b>	fire	<1; 2>
<b>6</b>	gold	<3; 1,2,4>
<b>7</b>	of	<3; 2,3,4>
<b>8</b>	in	<3; 2,3,4>
<b>9</b>	shipment	<2; 2,4>
<b>10</b>	silver	<2; 1,3>
<b>11</b>	truck	<3; 1,3,4>

## 2.3.2 Inverted Files

### Strength

- permit fast searching (nothing else can manage files of gigabytes or terabytes with adequate response time)
- used by many search engines

### Weakness

- cannot search for arbitrary expressions
- list needs to be constructed and sorted before search
- difficult to update
- takes a lot of space



## 2.3.3 Hash Tables


- Hash storage system – take each input word, compute its hash function, and store the information (word) in the location pointed to by the hash function
- Hash functions need to be carefully designed

## 2.3.3 Hash Tables

e.g. Given the same sentence “now is time for all good men to come to the aid of the party”, assume value of letters  $a = 1$ ,  $b = 2, \dots$   
how to assign each word a value in such a way that the bucket sizes in the hash table are balanced?

$v = \text{sum}(\text{letter values})?$

Word	Value
now	52
is	28
the	33
time	47
for	39
all	25
good	41
men	32
to	35
come	36
to	35
the	33
aid	14
of	21
the	33
party	80



small dataset, big  
value range

## 2.3.3 Hash Tables

$v = \text{sum}(\text{letter values}) \% 19$  ?

Word	Value
now	14
is	9
the	14
time	9
for	1
all	6
good	3
men	13
to	16
come	17
to	16
the	14
aid	14
of	2
the	14
party	4

Better, but...  
empty buckets:  
5-8,10-12,15

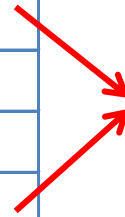
Bucket	Word
1	for
2	of
3	good
4	party
9	time, is
13	men
14	the, aid, now
16	to
17	come

collision

## 2.3.3 Hash Tables

multiply each letter by its position in the word?

Bucket	Word
2	time
3	good
4	all
8	men, of
9	is
11	party
12	to, aid
13	the
14	for
16	come
18	now

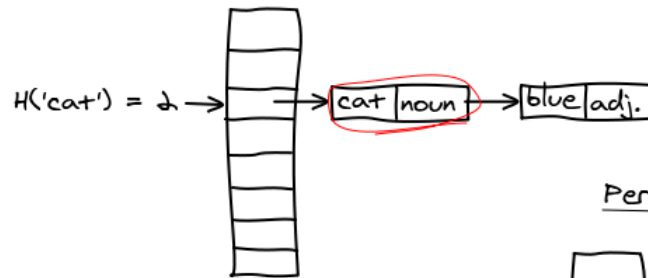


still two collisions

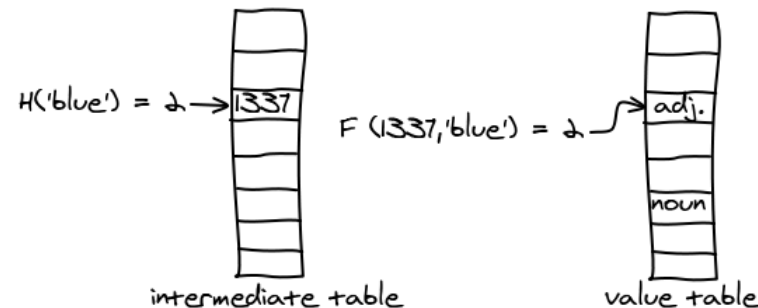
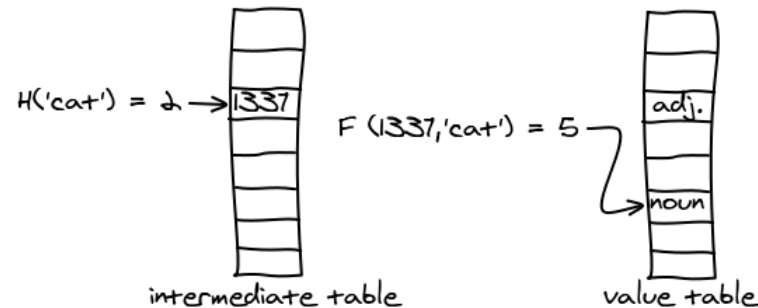
## 2.3.3 Hash Tables

**Perfect hashing** – an auxiliary table is made to adjust the hash function so that collisions never come up (Cormack et al., 1985)

Traditional Hash Table Lookup



Perfect Hash Table Lookup



## 2.3.3 Hash Tables

- Search in hash table
  - ✓ repeat the hash computation on the word sought and look in the result bucket
  - ✓ in general results are close to one-probe-per-item
- Search for collided items
  - ✓ buckets normally organized into a “collision chain” of cells, putting the items in other cells which are empty.
  - ✓ the system can use either binary search or a second hash function through the “collision chain” for additional items

## 2.3.3 Hash Tables

### Strength

- result close to “one-probe-per-item”, can provide faster retrieval than binary search

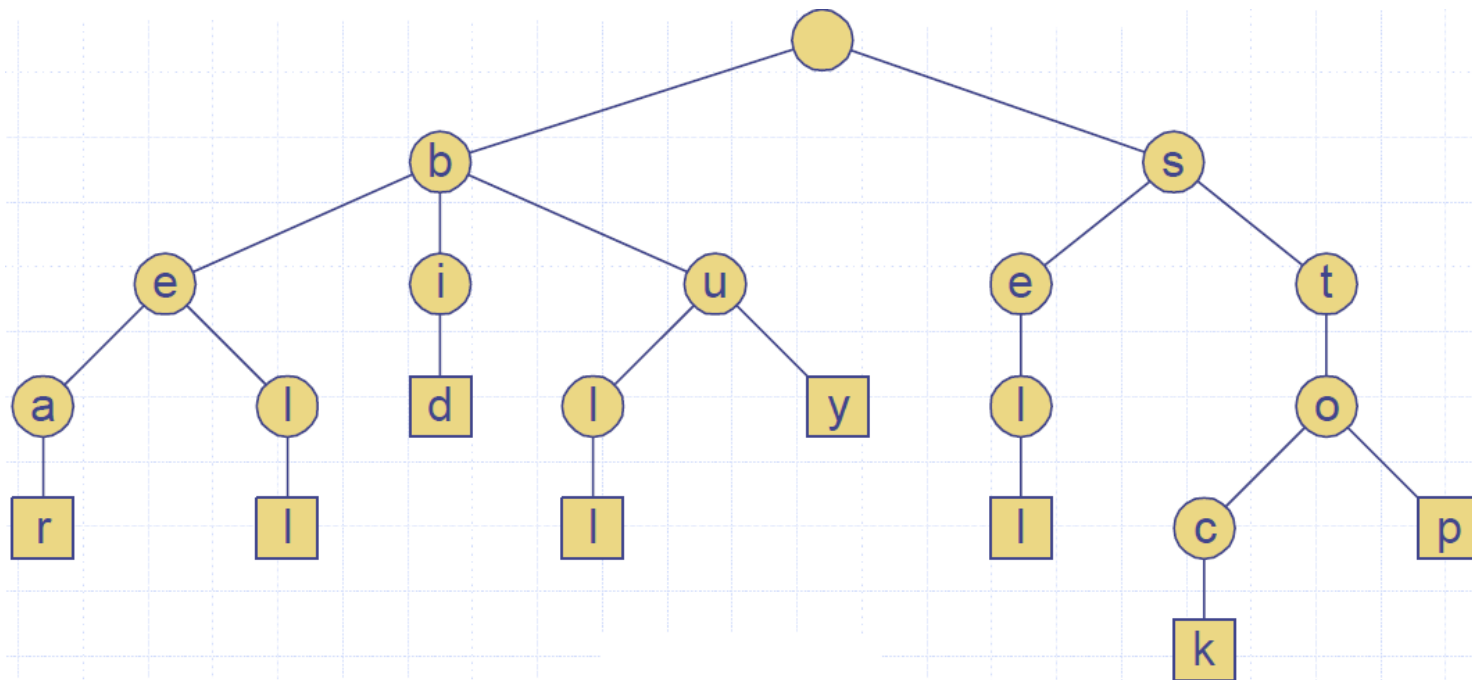
### Weakness

- similar words can be stored far apart, bad for searching for set of words with common beginnings
- hash table needs to be constructed and sorted before search
- need well designed hash function

## 2.3.4 Data structures to improve linear searching

- **Tries** - a multi-way tree structure useful for storing strings over an alphabet.

bear, bell, bid, bull, bug, sell, stock, stop





## 2. Text Documents

2.1 Text Documents in Digital Libraries

2.2 Markup, Page Description & Style Sheets

2.3 Searching Text

**2.4 Language processing**

2.5 Summary

# Thesauri

- Two types of retrieval errors:
  - errors of omission (recall failure), often occur when the same concept is expressed in two or more different words (**ship, boat**)
  - errors of inclusion (precision failure), can arise by the presence of ambiguous words (**rock- music or stone?**)
- Thesauri
  - Thesaurus gives a single label for each separate idea and then a list of equivalent terms
  - More details in section 3 (Knowledge representation)

## 2.5 Summary

- Text Documents in Digital Libraries  
source, format
- Markup, Page Description & Style Sheets  
structure, appearance
- Searching Text  
linear search, Boyer-Moore algorithm, inverted files, hash tables
- Language processing  
phrase detection, thesauri