Processamento e Recuperação de Informação

Basic Concepts

Index Construction

Memory-Based Inversion

Sort-based Inversion

Multiway Merging

In-place Merging

Comparison of Inversion Methods

Processamento e Recuperação de Informação

Efficient Index Construction

Departamento de Engenharia Informática
Instituto Superior Técnico

1º Semestre
2018/2019

Outline

1. Basic Concepts
2. Index Construction
3. Memory-Based Inversion
4. Sort-based Inversion
5. Multiway Merging
6. In-place Merging
7. Comparison of Inversion Methods

Bibliography

Processamento e Recuperação de Informação

Basic Concepts

Index Construction

Memory-Based Inversion

Sort-based Inversion

Multiway Merging

In-place Merging

Comparison of Inversion Methods

Managing Gigabytes: Compressing and Indexing Documents and Images - 2nd edition Ian H. Witten, Alistair Moffat, Timothy C. Bell Morgan Kaufmann 2000

Outline

1. Basic Concepts
2. Index Construction
3. Memory-Based Inversion
4. Sort-based Inversion
5. Multiway Merging
6. In-place Merging
7. Comparison of Inversion Methods

Text Indexes

- An index is a mechanism to locate a given term in the document collection
- Index types:
  - inverted files
  - signature files
  - bitmaps

Basic Concepts
Index Construction
Memory-Based Inversion
Sort-based Inversion
Multiway Merging
In-place Merging
Comparison of Inversion Methods

Definitions

- A document collection is a set of separate documents
- Documents are composed of terms
- Indexes provide efficient access to the documents given one or more terms
- Indexes can have different granularities:
  - set of documents $\rightarrow$ document $\rightarrow \cdots \rightarrow$ paragraph $\rightarrow$ word

A Document Collection

Doc.	Text
1	That government is best which governs least
2	That government is best which governs not at all
3	The mass of men serve the state not as men, but as machines
4	Wooden men can be manufactured that will serve the purpose as well
5	Government is at best but an expedient
6	But most governments are usually inexpedient

An Inverted File

Lexicon

Num.	Term
1	best
2	expedient
3	government
4	governs
5	inexpedient
6	least
7	machines
8	manufactured
9	mass
10	men
11	purpose
12	serve
13	state
14	wooden

Inverted file

Num.	Inverted list
1	$\langle 3; 1, 2, 5\rangle$
2	$\langle 1; 5\rangle$
3	$\langle 4; 1, 2, 5, 6\rangle$
4	$\langle 2; 1, 2\rangle$
5	$\langle 1; 6\rangle$
6	$\langle 1; 1\rangle$
7	$\langle 1; 3\rangle$
8	$\langle 1; 4\rangle$
9	$\langle 1; 3\rangle$
10	$\langle 2; 3, 4\rangle$
11	$\langle 1; 4\rangle$
12	$\langle 2; 3, 4\rangle$
13	$\langle 1; 3\rangle$
14	$\langle 1; 4\rangle$

Inverted file with word granularity

Inverted File

Num.	Inverted list
1	$\langle 3; (1;4), (2;4), (5;4) \rangle$
2	$\langle 1; (5;7) \rangle$
3	$\langle 4; (1;2), (2;2), (5;1), (6;3) \rangle$
4	$\langle 2; (1;6), (2;6) \rangle$
5	$\langle 1; (6;6) \rangle$
6	$\langle 1; (1;7) \rangle$
7	$\langle 1; (3;13) \rangle$
8	$\langle 1; (4;5) \rangle$
9	$\langle 1; (3;2) \rangle$
10	$\langle 3; (3;4), (3;10), (4;2) \rangle$
11	$\langle 1; (4;10) \rangle$
12	$\langle 2; (3;5), (4;8) \rangle$
13	$\langle 1; (3;7) \rangle$
14	$\langle 1; (4;1) \rangle$

Query Processing

- Query processing involves searching the index and manipulating the inverted lists

Basic Concepts
Index Construction
Memory-Based Inversion
Sort-based Inversion
Multiway Merging
In-place Merging
Comparison of Inversion Methods

Query Processing

- Query processing involves searching the index and manipulating the inverted lists
- For instance:

Basic Concepts
Index Construction
Memory-Based Inversion
Sort-based Inversion
Multiway Merging
In-place Merging
Comparison of Inversion Methods

Query Processing

- Query processing involves searching the index and manipulating the inverted lists
- For instance:
  - Query "government and men"

Query Processing

- Query processing involves searching the index and manipulating the inverted lists
- For instance:
  - Query "government and men" $\Rightarrow$ intersection between inverted lists 3 and 10 ($= \emptyset$);

Query Processing

- Query processing involves searching the index and manipulating the inverted lists
- For instance:
  - Query "government and men" $\Rightarrow$ intersection between inverted lists 3 and 10 ($= \emptyset$);
  - Query "men or machines"

Query Processing

- Query processing involves searching the index and manipulating the inverted lists
- For instance:
  - Query “government and men” $\Rightarrow$ intersection between inverted lists 3 and 10 $(= \emptyset)$;
  - Query “men or machines” $\Rightarrow$ union between inverted lists 10 and 7 $(= \{3, 4\})$;

Query Processing

Basic Concepts
Index Construction
Memory-Based Inversion
Sort-based Inversion
Multiway Merging
In-place Merging
Comparison of Inversion Methods

- Query processing involves searching the index and manipulating the inverted lists
- For instance:
  - Query "government and men" $\Rightarrow$ intersection between inverted lists 3 and 10 ($= \emptyset$);
  - Query "men or machines" $\Rightarrow$ union between inverted lists 10 and 7 ($= \{3, 4\}$);
  - Query "men and not machines"

Query Processing

- Query processing involves searching the index and manipulating the inverted lists
- For instance:
  - Query "government and men" $\Rightarrow$ intersection between inverted lists 3 and 10 $(= \emptyset)$;
  - Query "men or machines" $\Rightarrow$ union between inverted lists 10 and 7 $(= \{3, 4\})$;
  - Query "men and not machines" $\Rightarrow$ inverted list 10 except inverted list 7 $(= \{4\})$;

Query Processing

- Query processing involves searching the index and manipulating the inverted lists
- For instance:
  - Query "government and men" $\Rightarrow$ intersection between inverted lists 3 and 10 ($= \emptyset$);
  - Query "men or machines" $\Rightarrow$ union between inverted lists 10 and 7 ($= \{3, 4\}$);
  - Query "men and not machines" $\Rightarrow$ inverted list 10 except inverted list 7 ($= \{4\}$);

More on this later...

Outline

1 Basic Concepts
2 Index Construction
3 Memory-Based Inversion
4 Sort-based Inversion
5 Multiway Merging
6 In-place Merging
7 Comparison of Inversion Methods

An example collection

Text size	B	5×10^9 bytes
N. of Documents	N	5×10^6
N. of distinct terms	n	1×10^6
Total n. of terms	F	800×10^6
N. of index pointers	f	400×10^6
Size of compressed inverted file	I	400×10^6 bytes
Size of lexicon	L	3×10^6 bytes
Disk seek time	t_s	10×10^{-3} sec
Disk transfer time per byte	t_r	0.5×10^{-6} sec
Coding time per byte	t_d	5×10^{-6} sec
Time to compare and swap 10 byte records	t_c	10^{-6} sec
Time to parse, stem, look up one term	t_p	20×10^{-6} sec
Amount of main memory available	M	40×10^6 bytes

A naive solution

1. Create a document×term frequency matrix

doc \ term	1	2	3	4	...
1	2	1	-	-	...
2	-	-	-	2	...
3	1	1	-	3	...
4	6	1	-	-	...
...					

2. Write matrix to disk one column at a time

A naive solution

Processamento e Recuperação de Informação

Basic Concepts

Index Construction

Memory-Based Inversion

Sort-based Inversion

Multiway Merging

In-place Merging

Comparison of Inversion Methods

1. Create a document×term frequency matrix

doc \ term	1	2	3	4	...
1	2	1	-	-	...
2	-	-	-	2	...
3	1	1	-	3	...
4	6	1	-	-	...
...					

2. Write matrix to disk one column at a time

Memory requirements (using 4 bytes/entry):

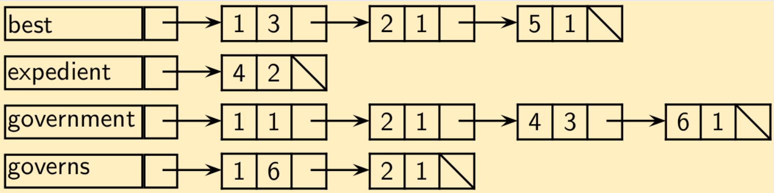
$$4 \times 1\,000\,000 \times 5\,000\,000 = 18\,Tbytes$$

Outline

1. Basic Concepts
2. Index Construction
3. Memory-Based Inversion
4. Sort-based Inversion
5. Multiway Merging
6. In-place Merging
7. Comparison of Inversion Methods

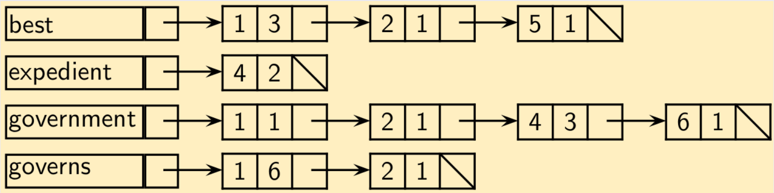
Memory-Based Inversion

Using a dictionary structure



Memory-Based Inversion

Using a dictionary structure



Time requirements

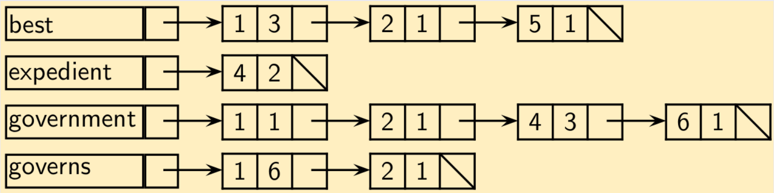
$$T = \underset{\substack{\uparrow \\ \text{parse text}}}{Bt_r + Ft_p} + \underset{\substack{\uparrow \\ \text{write file}}}{I(t_d + t_r)}$$

$\approx 6h$

B = text size
F = n. terms
t_r = transfer rate
t_p = parse time
I = file size
t_d = coding time

Memory-Based Inversion

Using a dictionary structure



Time requirements

$$T = \underset{\substack{\uparrow \\ \text{parse text}}}{Bt_r + Ft_p} + \underset{\substack{\uparrow \\ \text{write file}}}{I(t_d + t_r)}$$

$\approx 6h$

Space requirements

$$S = \underset{\substack{\uparrow \\ \text{pointer size}}}{(4 + 4 + 2)} \times \underset{\substack{\uparrow \\ \text{n. of pointers}}}{400\,000\,000}$$

$\approx 4Gb$

Outline

1. Basic Concepts
2. Index Construction
3. Memory-Based Inversion
4. Sort-based Inversion
5. Multiway Merging
6. In-place Merging
7. Comparison of Inversion Methods

Sort-based Inversion

- If virtual memory was used, the dictionary-based algorithm would take about 6 weeks to index the 5 Gbyte collection
- Mainly due to disk seek time
- Solution: allow only sequential disk access

Sort-based Inversion Algorithm (I)

1. Create an empty dictionary structure S;
   Create an empty temporary file T on disk;
2. For each document D_d $(1 \leq d \leq N)$:
   1. Read and parse D_d;
   2. For each triple $\langle t, d, f_{d,t} \rangle$
      1. Store t in S;
      2. Store $\langle t, d, f_{d,t} \rangle$ in T;

S	
that	1
government	2
best	3
governs	4
least	5
...	

T
$\langle 1, 1, 1 \rangle$
$\langle 2, 1, 1 \rangle$
$\langle 3, 1, 1 \rangle$
$\langle 4, 1, 1 \rangle$
$\langle 5, 1, 1 \rangle$
$\langle 2, 2, 1 \rangle$
$\langle 3, 2, 1 \rangle$
$\langle 4, 2, 1 \rangle$
$\langle 5, 3, 1 \rangle$
$\langle 6, 3, 2 \rangle$
...

Sort-based Inversion Algorithm (II)

3. Le k be the number of records that fit in memory
   1. Read k records from T;
   2. Sort according to t and d;
   3. Write the sorted run into T;
   4. Repeat until all runs are sorted;
4. Pairwise merge the sorted runs;

T
$\langle 1,1,1\rangle$
$\langle 2,1,1\rangle$
$\langle 3,1,1\rangle$
$\langle 4,1,1\rangle$
$\langle 5,1,1\rangle$
$\langle 2,2,1\rangle$
$\langle 3,2,1\rangle$
$\langle 4,2,1\rangle$
$\langle 5,3,1\rangle$
$\langle 6,3,2\rangle$
...

$\rightarrow$

T
$\langle 1,1,1\rangle$
$\langle 2,1,1\rangle$
$\langle 3,1,1\rangle$
$\langle 2,2,1\rangle$
$\langle 4,1,1\rangle$
$\langle 5,1,1\rangle$
$\langle 3,2,1\rangle$
$\langle 4,2,1\rangle$
$\langle 5,3,1\rangle$
$\langle 6,3,2\rangle$
...

Sort-based Inversion Algorithm (II)

3. Le k be the number of records that fit in memory
   1. Read k records from T;
   2. Sort according to t and d;
   3. Write the sorted run into T;
   4. Repeat until all runs are sorted;
4. Pairwise merge the sorted runs;

T
$\langle 1,1,1\rangle$
$\langle 2,1,1\rangle$
$\langle 3,1,1\rangle$
$\langle 2,2,1\rangle$
$\langle 4,1,1\rangle$
$\langle 5,1,1\rangle$
$\langle 3,2,1\rangle$
$\langle 4,2,1\rangle$
$\langle 5,3,1\rangle$
$\langle 6,3,2\rangle$
...

$\rightarrow$

T
$\langle 1,1,1\rangle$
$\langle 2,1,1\rangle$
$\langle 2,2,1\rangle$
$\langle 3,1,1\rangle$
$\langle 3,2,1\rangle$
$\langle 4,1,1\rangle$
$\langle 4,2,1\rangle$
$\langle 5,1,1\rangle$
$\langle 5,3,1\rangle$
$\langle 6,3,2\rangle$
...

Sort-based Inversion Algorithm (III)

Processamento e Recuperação de Informação

Basic Concepts

Index Construction

Memory-Based Inversion

Sort-based Inversion

Multiway Merging

In-place Merging

Comparison of Inversion Methods

5. For each term t $(1 \leq t \leq n)$:
   1. Start a new inverted file entry;
   2. Read all triples $\langle t, d, f_{d,t} \rangle$ from T and form the inverted list of t;
   3. Compress the inverted list;
   4. Append the list to the inverted file.

T
$\langle 1, 1, 1 \rangle$
$\langle 2, 1, 1 \rangle$ $\langle 2, 2, 1 \rangle$
$\langle 3, 1, 1 \rangle$ $\langle 3, 2, 1 \rangle$
$\langle 4, 1, 1 \rangle$ $\langle 4, 2, 1 \rangle$
$\langle 5, 1, 1 \rangle$ $\langle 5, 3, 1 \rangle$
$\langle 6, 3, 2 \rangle$
...

$\rightarrow$

t	Inverted list
1	$\langle 1; (1; 1) \rangle$
2	$\langle 2; (1; 1), (2; 1) \rangle$
3	$\langle 2; (1; 1), (2; 1) \rangle$
4	$\langle 2; (1; 1), (2; 1) \rangle$
5	$\langle 2; (1; 1), (3; 1) \rangle$
6	$\langle 1; (3; 2) \rangle$
...	

Sort-based Inversion Requirements

Time requirements

$$\begin{aligned} T = \quad & Bt_r + Ft_p + 10ft_r & \leftarrow \quad & \text{read,parse,write} \\ & +20ft_r + R(1.2k \log k)t_c & \leftarrow \quad & \text{sort runs} \\ & +\lceil \log R \rceil (20ft_r + ft_c) & \leftarrow \quad & \text{merge runs} \\ & +10ft_r + I(t_d + t_r) & \leftarrow \quad & \text{write inverted file} \end{aligned}$$

$\approx 20h$

B = text size
t_r = transfer rate
F = total n. terms
t_p = parse time
f = n. index pointers
R = n. runs
k = size of run
t_c = compare+swap
I = file size
t_d = coding time

Sort-based Inversion Requirements

Time requirements

$$\begin{aligned} T = \quad & Bt_r + Ft_p + 10ft_r && \leftarrow \quad \text{read,parse,write} \\ & +20ft_r + R(1.2k\log k)t_c && \leftarrow \quad \text{sort runs} \\ & +\lceil \log R \rceil (20ft_r + ft_c) && \leftarrow \quad \text{merge runs} \\ & +10ft_r + I(t_d + t_r) && \leftarrow \quad \text{write inverted file} \end{aligned}$$

$\approx 20h$

Space requirements

$$S = \underbrace{10 \times 400\,000\,000}_{\text{temp. file size}} \times \underbrace{2}_{\text{plus one copy}}$$

$\approx 8Gb$

Compression of the Temporary File

- The $\langle t, d, f_{d,t} \rangle$ triples can be compressed
  - For $\langle d, f_{d,t} \rangle$ we can use methods appropriate for index compression (which is another subject altogether)
  - E.g., Elias-δ and unary coding

Compression of the Temporary File

Processamento e Recuperação de Informação

Basic Concepts

Index Construction

Memory-Based Inversion

Sort-based Inversion

Multiway Merging

In-place Merging

Comparison of Inversion Methods

- The $\langle t, d, f_{d,t} \rangle$ triples can be compressed
  - For $\langle d, f_{d,t} \rangle$ we can use methods appropriate for index compression (which is another subject altogether)
  - E.g., Elias-δ and unary coding
- How to code t?

Using Gaps

- Term numbers can be coded as sequences of *t*-gaps within each sorted run

$$\left|\begin{array}{c} \ldots \\ \langle 2,2,1\rangle \\ \langle 4,1,1\rangle \\ \langle 5,1,1\rangle \\ \ldots \end{array}\right| \rightarrow \left|\begin{array}{c} \ldots \\ \langle 2,2,1\rangle \\ \langle 3,1,1\rangle \\ \langle 2,1,1\rangle \\ \ldots \end{array}\right|$$

- A t-gap x is encoded as value $x+1$, using unary coding
- The same can be applied for d, within the term triples

Compression Requirements

Time requirements

- Runs must be sorted *before writing to disk*
- Must share space with dictionary $\Rightarrow$ shorter initial runs $\Rightarrow$ more runs to merge
- Requirements:

$$\begin{array}{rlcl} T = & Bt_r + Ft_p & \leftarrow & \text{read,parse} \\ & +R(1.2k\log k)t_c + I'(t_r + t_d) & \leftarrow & \text{sort,compress,write} \\ & \lceil \log R \rceil (2I'(t_r + t_d) + ft_c) & \leftarrow & \text{merge runs} \\ & (I' + I)(t_r + t_d) & \leftarrow & \text{recompress} \end{array}$$

$\approx 26h$

B = text size
t_r = transfer rate
F = total n. terms
t_p = parse time
R = n. runs
k = size of run
t_c = compare+swap
I' = temp. file size
t_d = coding time
f = n. index pointers
I = file size

Compression Requirements

Time requirements

- Runs must be sorted *before writing to disk*
- Must share space with dictionary ⇒ shorter initial runs ⇒ more runs to merge
- Requirements:

$$
\begin{array}{rlcl}
T = & Bt_r + Ft_p & \leftarrow & \text{read,parse} \\
 & +R(1.2k \log k)t_c + I'(t_r + t_d) & \leftarrow & \text{sort,compress,write} \\
 & \lceil \log R \rceil (2I'(t_r + t_d) + ft_c) & \leftarrow & \text{merge runs} \\
 & (I' + I)(t_r + t_d) & \leftarrow & \text{recompress}
\end{array}
$$

$\approx 26h$

Space requirements

$$
\begin{array}{lccccc}
S = & (1.1Mb & + & 0.25Mb) & \times & 400 \quad \times 2 \\
 & \uparrow & & \uparrow & & \uparrow \\
 & 10^6 \ \langle d, f_{d,t} \rangle \text{ pairs} & & \text{k+n bits} & & \text{runs} \\
 & & & \text{for } t\text{-gaps} & & (4\times \text{ more})
\end{array}
$$

$\approx 1Gb$

Outline

1. Basic Concepts
2. Index Construction
3. Memory-Based Inversion
4. Sort-based Inversion
5. Multiway Merging
6. In-place Merging
7. Comparison of Inversion Methods

Multiway Merging

- The merging process can be improved
- Instead of a 2-way merge, use an R-way merge
  - One pass $\Rightarrow$ pointers coded/decoded only once
  - Increased seek time (+1m, approximately)

Processamento e Recuperação de Informação

Basic Concepts

Index Construction

Memory-Based Inversion

Sort-based Inversion

Multiway Merging

In-place Merging

Comparison of Inversion Methods

Time Requirements

$$\begin{array}{rll} T = & Bt_r + Ft_p & \leftarrow \quad \text{read,parse} \\ & +R(1.2k\log k)t_c + I'(t_r + t_d) & \leftarrow \quad \text{sort,compress,write} \\ & f\lceil \log R \rceil t_c + I'(t_s/b + t_r + t_d) & \leftarrow \quad \text{merge runs} \\ & I(t_r + t_d) & \leftarrow \quad \text{recompress} \end{array}$$

$\approx 11h$

B = text size	t_r = transfer rate
F = total n. terms	t_p = parse time
R = n. runs	k = size of run
t_c = compare+swap	I' = temp. file size
t_d = coding time	f = n. index pointers
t_s = seek time	$b \leq M/R$ = input buffer
I = file size	

Multiway Merging

Processamento e Recuperação de Informação

Basic Concepts
Index Construction
Memory-Based Inversion
Sort-based Inversion
Multiway Merging
In-place Merging
Comparison of Inversion Methods

Time Requirements

$$
\begin{aligned}
T = \quad & Bt_r + Ft_p && \leftarrow \quad \text{read,parse} \\
& +R(1.2k \log k)t_c + I'(t_r + t_d) && \leftarrow \quad \text{sort,compress,write} \\
& f\lceil \log R \rceil t_c + I'(t_s/b + t_r + t_d) && \leftarrow \quad \text{merge runs} \\
& I(t_r + t_d) && \leftarrow \quad \text{recompress}
\end{aligned}
$$

$\approx 11h$

Space requirements

One copy of the temporary file: $540Mb$

Outline

1. Basic Concepts
2. Index Construction
3. Memory-Based Inversion
4. Sort-based Inversion
5. Multiway Merging
6. In-place Merging
7. Comparison of Inversion Methods

In-place Multiway Merging

Processamento e Recuperação de Informação

Basic Concepts

Index Construction

Memory-Based Inversion

Sort-based Inversion

Multiway Merging

In-place Merging

Comparison of Inversion Methods

- Assume that runs are split into blocks of *b* bytes (with added padding, if necessary)
- R blocks can be read into memory and overwritten by output blocks of *b* bytes
  - The output blocks contain the inverted lists
  - During the process all values can be recoded using the most efficient code
- If a block overruns the vacant space, it can be appended to the end of the file
- At the end of the merge, blocks must be sorted
  - Block-sorting can be done in linear time, using two *b*-byte buffers

In-place Multiway Merging

Processamento e Recuperação de Informação

Basic Concepts

Index Construction

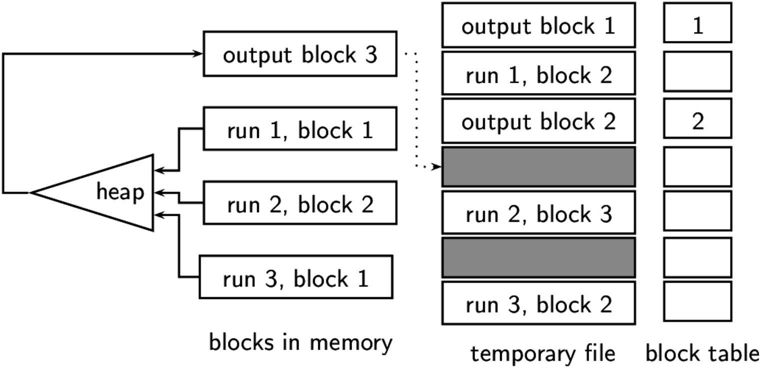
Memory-Based Inversion

Sort-based Inversion

Multiway Merging

In-place Merging

Comparison of Inversion Methods



Block-sorting algorithm

For $i \leftarrow 1$ to *nblocks*, if $i \neq blockTable[i]$ then:

1. Read block i into memory
2. Set $holding \leftarrow blockTable[i]$
3. Set $vacant \leftarrow i$
4. While $holding \neq vacant$ do
    1. Find j such that $blockTable[j] = vacant$
    2. Copy block j to *vacant*
    3. Set $blockTable[vacant] \leftarrow vacant$
    4. Set $vacant \leftarrow j$
5. Write block in memory to *vacant*
6. Set $blockTable[vacant] \leftarrow holding$

Multiway Merging Algorithm (I)

1. Initialization:
   Create an empty dictionary structure S;
   Create an empty temporary file T on disk;
   Set $L \leftarrow |S|$
   Set $k \leftarrow (M - L)/w$, where w is the number of bytes required to store one $\langle t, d, f_{d,t} \rangle$ record
   Set $b \leftarrow 50Kb$
   Set $R \leftarrow 0$

Multiway Merging Algorithm (II)

2. For each document D_d, $1 \leq d \leq N$:
   1. Read and parse D_d
   2. For each term $t \in D_d$
      1. Search S for t
      2. If t is not in S
         insert it
         set $L \leftarrow |S|$
         set $k \leftarrow (M - L)/w$
      3. Add a record $\langle t, d, f_{d,t} \rangle$ to the array of triples
   3. If, at any stage, the array of triples contains k items
      1. Sort the array (using quicksort)
      2. Write the array coding t-gaps in unary, d-gaps with δ and $f_{d,t}$ values in unary
      3. Add padding to complete a block of b bytes
      4. Set $R \leftarrow R + 1$
      5. If $(b \times (R + 1)) > M$, set $b \leftarrow b/2$

Multiway Merging Algorithm (III)

3. Merging:
    1. Read the first block from each run and add each block number to the free list
    2. Build heap with R candidates (one from each run)
    3. While the heap is non-empty
        1. Remove the root
        2. Add it to the output block, recoding
        3. Replace it by the next candidate from the same run
    4. Each time the output block is full
        1. Use the free list to find a vacant space; if there is none, append it to the end of the file
        2. Write the output block
        3. Update the free list and block table
    5. Each time an input block is empty
        1. Read the next block from this run
        2. Update the free list
4. Reorder the blocks
5. Truncate the inverted file

Algorithm Requirements

Time Requirements

$$
\begin{aligned}
T = \quad & Bt_r + Ft_p && \leftarrow \quad \text{read,parse} \\
& +R(1.2k \log k)t_c + I'(t_r + t_d) && \leftarrow \quad \text{sort,compress,writ} \\
& f\lceil \log R \rceil t_c + (I' + I)(t_s/b + t_r + t_d) && \leftarrow \quad \text{merge,recode} \\
& 2I'(t_s/b + t_r) && \leftarrow \quad \text{permute}
\end{aligned}
$$

$\approx 11h$

B = text size
F = total n. terms
R = n. runs
t_c = compare+swap
t_d = coding time
t_s = seek time
I = file size

t_r = transfer rate
t_p = parse time
k = size of run
I' = temp. file size
f = n. index pointers
$b \leq M/R$ = input buffer

Algorithm Requirements

Time Requirements

$$
\begin{array}{rlcl}
T = & Bt_r + Ft_p & \leftarrow & \text{read,parse} \\
 & +R(1.2k\log k)t_c + I'(t_r + t_d) & \leftarrow & \text{sort,compress,writ} \\
 & f\lceil \log R \rceil t_c + (I' + I)(t_s/b + t_r + t_d) & \leftarrow & \text{merge,recode} \\
 & 2I'(t_s/b + t_r) & \leftarrow & \text{permute}
\end{array}
$$

$\approx 11h$

Space Requirements

Inverted file and temporary file occupy the same space: 150 Mb

Outline

1. Basic Concepts
2. Index Construction
3. Memory-Based Inversion
4. Sort-based Inversion
5. Multiway Merging
6. In-place Merging
7. Comparison of Inversion Methods

Comparison of Inversion Methods

Method	Memory (Mb)	Disk (Mb)	Time (h)
Dictionary-based (mem.)	4 000	0	6
Dictionary-based (disk)	30	4 000	1 100
Sort-based	40	8 000	20
Sort-based compressed	40	1 080	26
Multiway merge	40	540	11
In-place multiway merge	40	150	11

Processamento e Recuperação de Informação

Basic Concepts
Index Construction
Memory-Based Inversion
Sort-based Inversion
Multiway Merging
In-place Merging
Comparison of Inversion Methods

Memory-based inversion methods

- Using minimal perfect hashing to eliminate the lexicon from memory
- Using in-memory compression
- Partitioning the inversion process
  - Lexicon-based partition
  - Text-based partition
- All these imply several passes through the text

Questions?