

Data Structures

Inverted Search

Team Emertxe



Project1 – Inverted Search



Inverted Search



What ?

Inverted Search



What ?

An inverted index is an index data structure storing a mapping from content, such as words or numbers, to its locations in a table, or in a document or a set of documents.



Inverted Search



What ?

An inverted index is an index data structure storing a mapping from content, such as words or numbers, to its locations in a table, or in a document or a set of documents

Purpose

The purpose of an inverted index is to allow fast full text searches, at a cost of increased processing when a document is added to the database.



Inverted Search



Types

1. Forward Indexing
2. Inverted Indexing

Inverted Search



Inverted Indexing

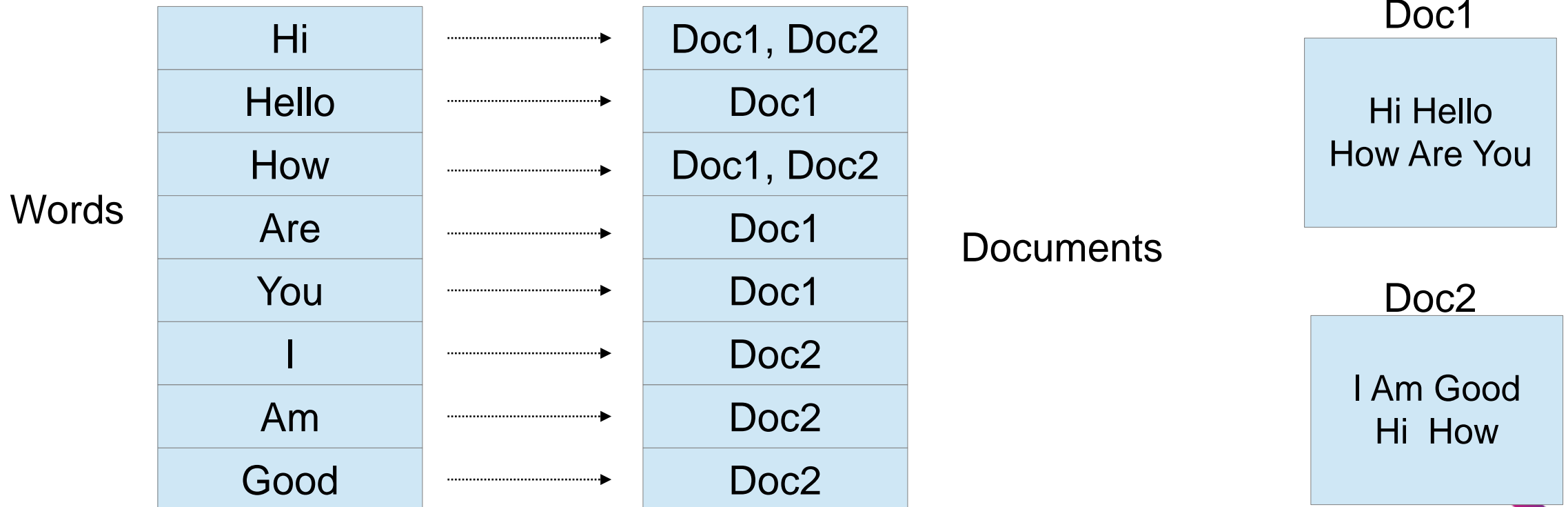
It is a data structure that stores mapping from words to documents or set of documents
i.e. directs you from word to document.



Inverted Search

Inverted Indexing

- Scan the documents, prepare a list of unique words.
- Prepare a list of indices of all the unique words and map them to a document search



Inverted Search



Inverted Indexing

- It is a data structure that stores mapping from words to documents or set of documents i.e. directs you from word to document.

Forward Indexing

- It is a data structure that stores mapping from documents to words i.e. directs you from document to word.

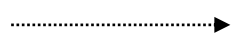


Inverted Search

Forward Indexing

- Scan the document,prepare a list of unique words
- Map all the words to a document as an index

Documents



Doc1



Hi
Hello
How
Are
You

Doc2



I
Am
Good
Hi
How

Doc1

Hi Hello
How Are You

Doc2

I Am Good
Hi How

Inverted Search



Inverted Indexing

- It is a data structure that stores mapping from words to documents or set of documents i.e. directs you from word to document.
- **Real life example of Inverted index:**
 - Index at the back of the book.

Forward Indexing

- It is a data structure that stores mapping from documents to words i.e. directs you from document to word.
- **Real life examples of Forward index:**
 - Table of contents in book

Data Structure –Project

Inverted Search



Inverted Indexing

- It is a data structure that stores mapping from words to documents or set of documents i.e. directs you from word to document.
- Real life example of Inverted index:
 - Index at the back of the book.

—Index—	
—A—	
about the author 128, 132, 412	automatic renewal 327-329, 341, 343
account info 295	Automatically Update 73-75, 94, 144
active table of contents 34, 120-124, 238-239, 285-286, 354, 366, 370	AZK 371
ACX 465-467	
Adobe 506	—B—
advertising 434, 439-449	back matter 124-129
age 312	background 47, 93, 181, 184, 192-193, 246, 252-253, 355, 370, 385, 390
aggregator 17-18, 322	bank information 295
alignment 68, 101-103, 105-106, 229-230, 261-262, 353-354, 380, 389	Barnes & Noble 506
Alt codes 39	biography 128, 132, 410
Amazon Associates 415	black 47, 93, 184, 192, 252-253, 355, 370, 385, 390
Amazon Follow 430, 437, 480	Blackberry 372-373
Amazon Giveaway 436-439	blank line 27-28, 110, 112-114, 276-277, 284-285, 385
Amazon Marketing Services (AMS) 439-449	blank page 354, 385-386
Android 167-169, 171, 371-375	block indent 50, 52, 67, 82, 106-107, 234-235
apostrophe 40, 42-44	blog 411, 429, 479
app 141-142	Blogger 429
Apple 169, 342, 372, 506	bloggers 327, 430
	blurb 300-306, 364, 406, 411-412, 417, 477
	blurry 162-164, 172, 175, 193, 246, 387, 389
	body text 66, 68, 79-82, 92-94, 115, 233-235



Forward Indexing

- It is a data structure that stores mapping from documents to words i.e. directs you from document to word.

- Real life examples of Forward index:

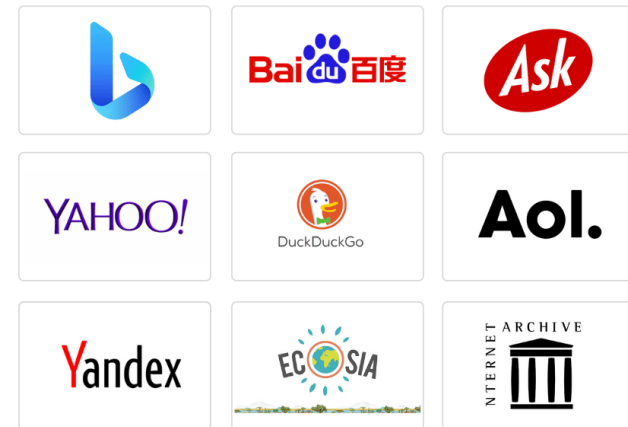
- Table of contents in book

Table of Contents	
Acknowledgments	ix
Introduction	xi
Part I Envision the Possibilities	
1 Welcome to Office 2010	3
Features that Fit Your Work Style	3
Changes in Office 2010	4
Let Your Ideas Soar	5
Collaborate Easily and Naturally	5
Work Anywhere—and Everywhere	6
Exploring the Ribbon	7

Inverted Search

Advantages of inverted Index:

- It is easy to develop
- It is used in document retrieval system
- Search engines



Inverted Search



Operations:

- 1.Create Database
- 2.Display Database
- 3.Search Database
- 4.Update Database
- 5.Save Database



Create Database



Data Structure –Project

Inverted Search

Create Database:

1. Main node
2. Sub node

File count	word
Sub node link	Link for next node

Main
node

word count
File name
link

Sub
node

File1.txt

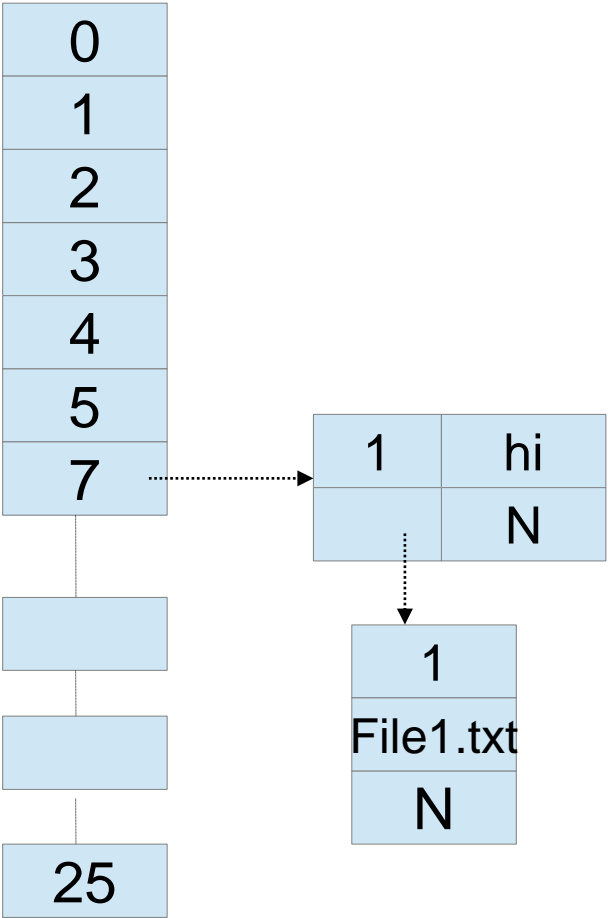
hi hello how are you

File2.txt

hello hi i am fine

Data Structure –Project Inverted Search

Create Database:



$$\begin{aligned}\text{Index} &= \text{data} \% 97 \\ &= \text{'h'} \% 97 \\ &= 7\end{aligned}$$

File1.txt

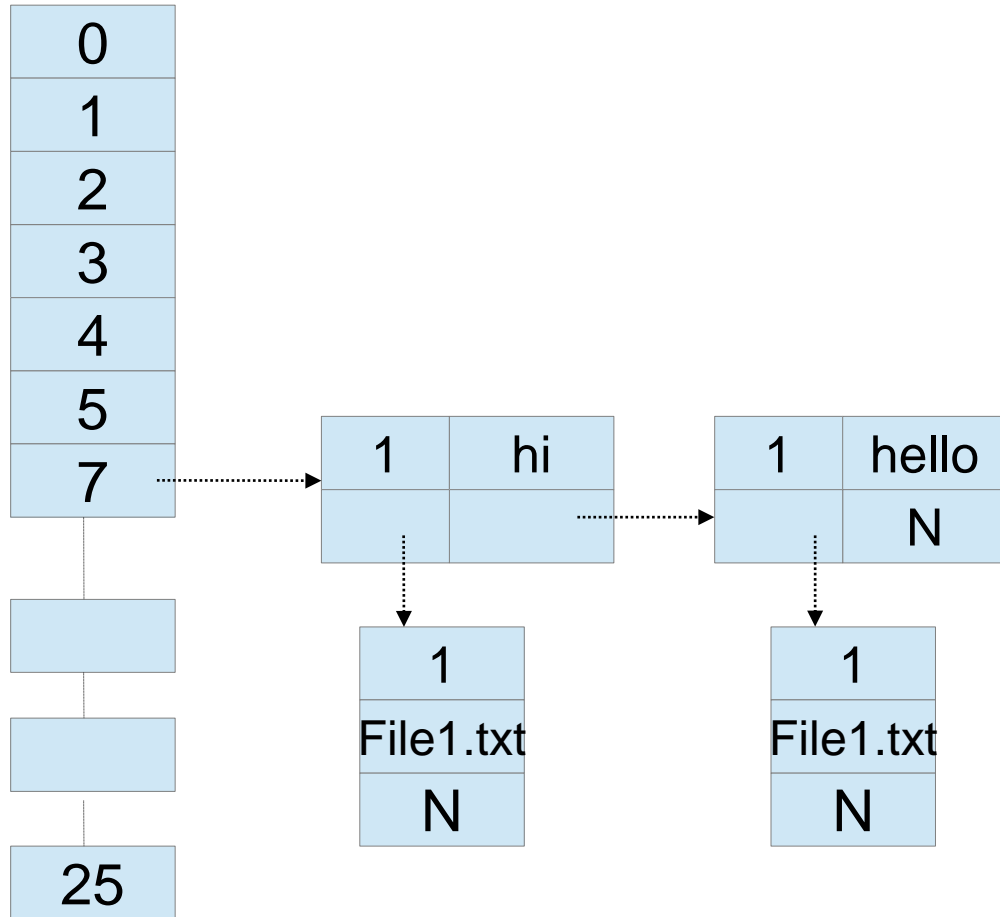
hi hello how are you

File2.txt

hello hi i am fine

Inverted Search

Create Database:



$$\begin{aligned}\text{Index} &= \text{data} \% 97 \\ &= \text{'h'} \% 97 \\ &= 7\end{aligned}$$

File1.txt

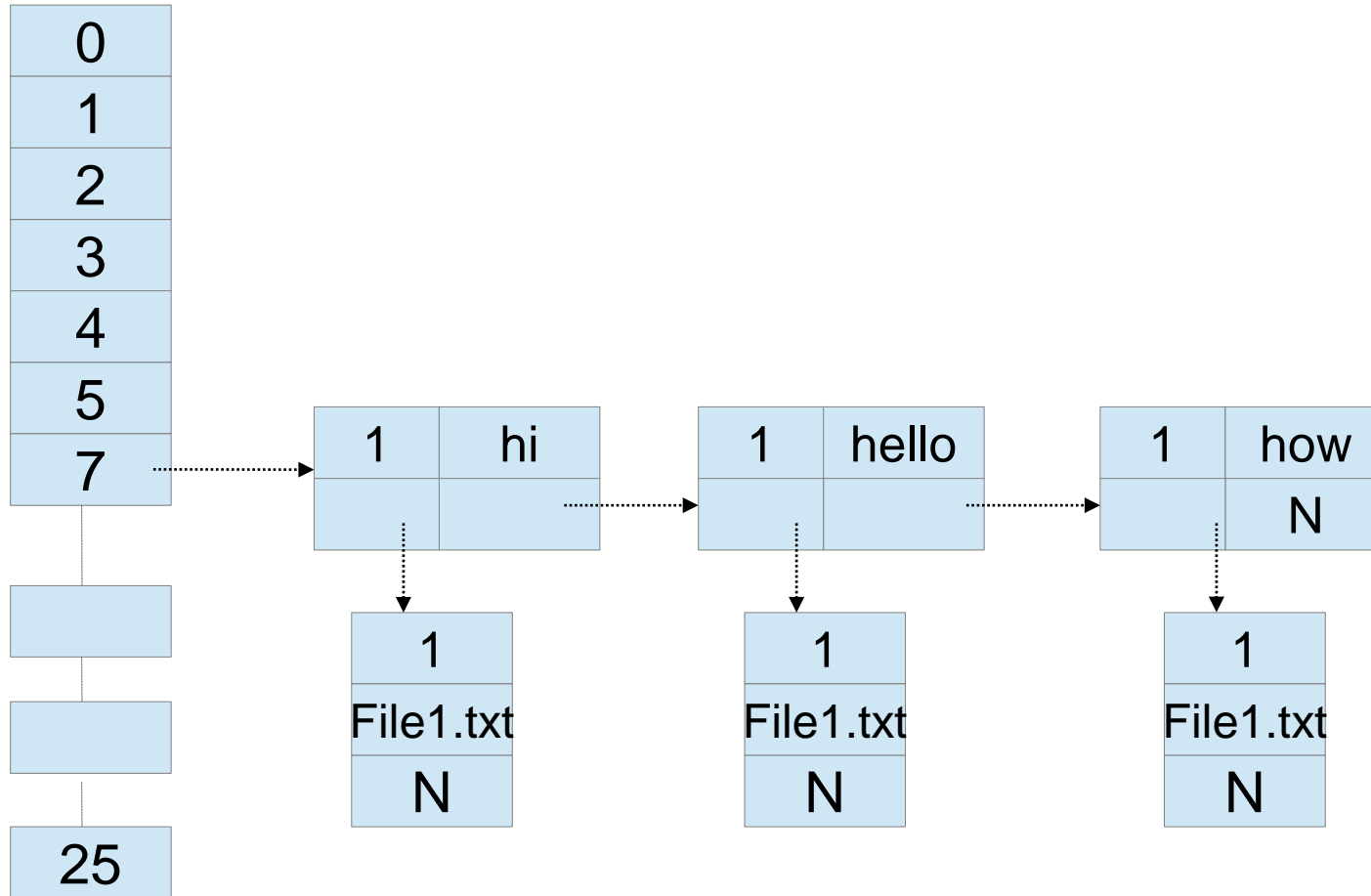
hi hello how are you

File2.txt

hello hi i am fine

Inverted Search

Create Database:



$$\begin{aligned}\text{Index} &= \text{data} \% 97 \\ &= \text{'h'} \% 97 \\ &= 7\end{aligned}$$

File1.txt

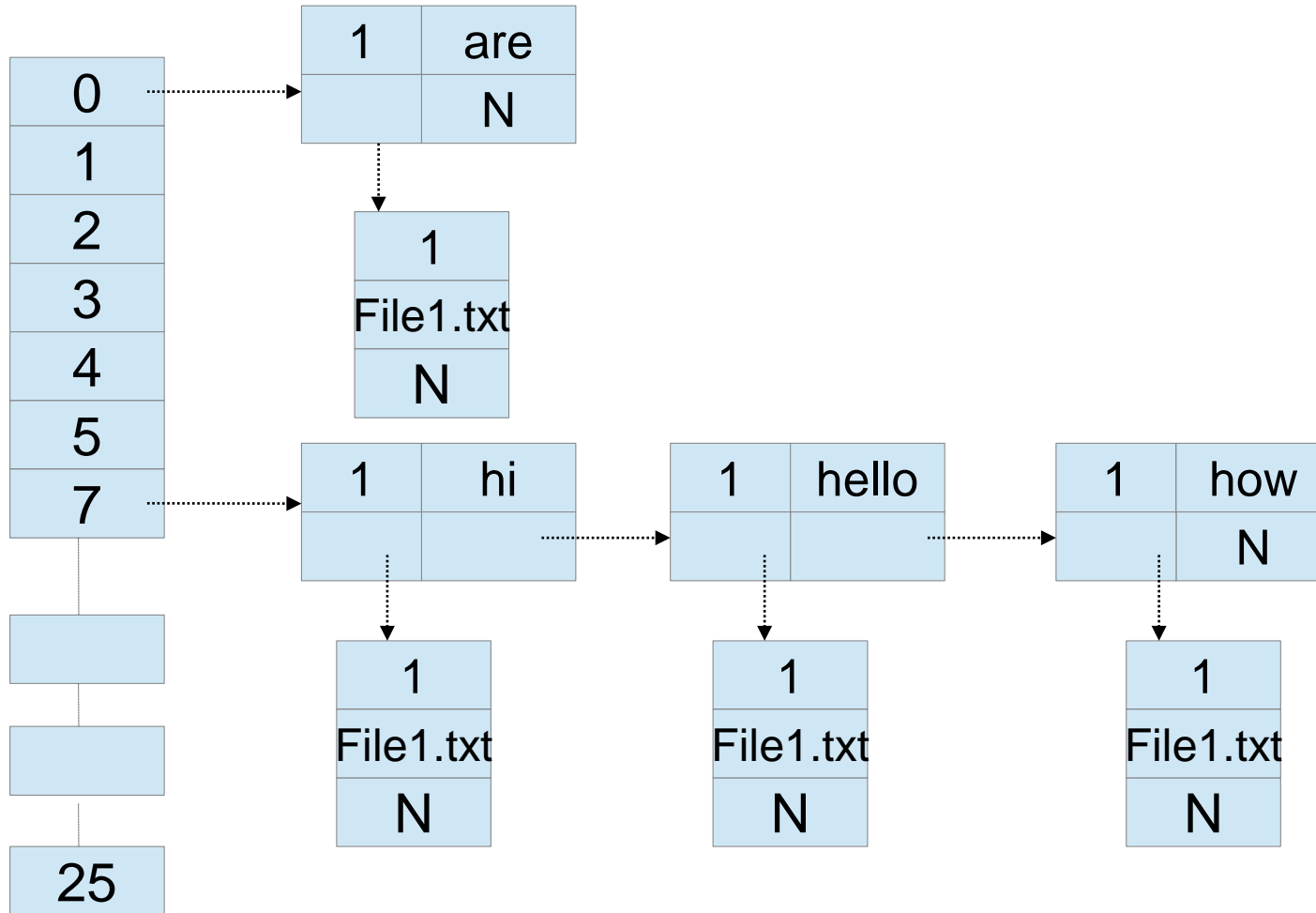
hi hello how are you

File2.txt

hello hi i am fine

Inverted Search

Create Database:



$$\begin{aligned}\text{Index} &= \text{data} \% 97 \\ &= 'a' \% 97 \\ &= 0\end{aligned}$$

File1.txt

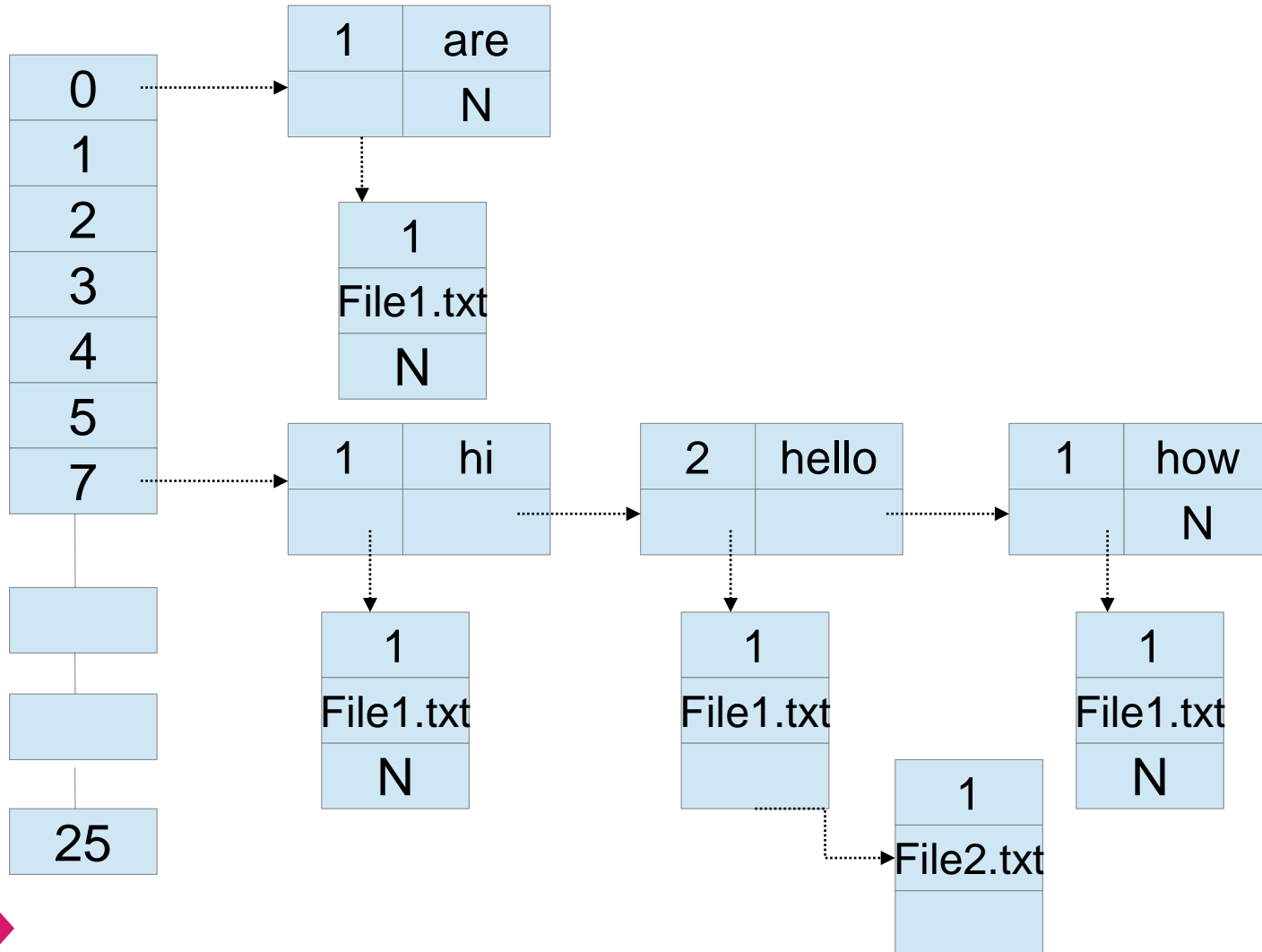
hi hello how are you

File2.txt

hello hi i am fine

Inverted Search

Create Database:



$$\begin{aligned}\text{Index} &= \text{data} \% 97 \\ &= \text{'h'} \% 97 \\ &= 7\end{aligned}$$

File1.txt

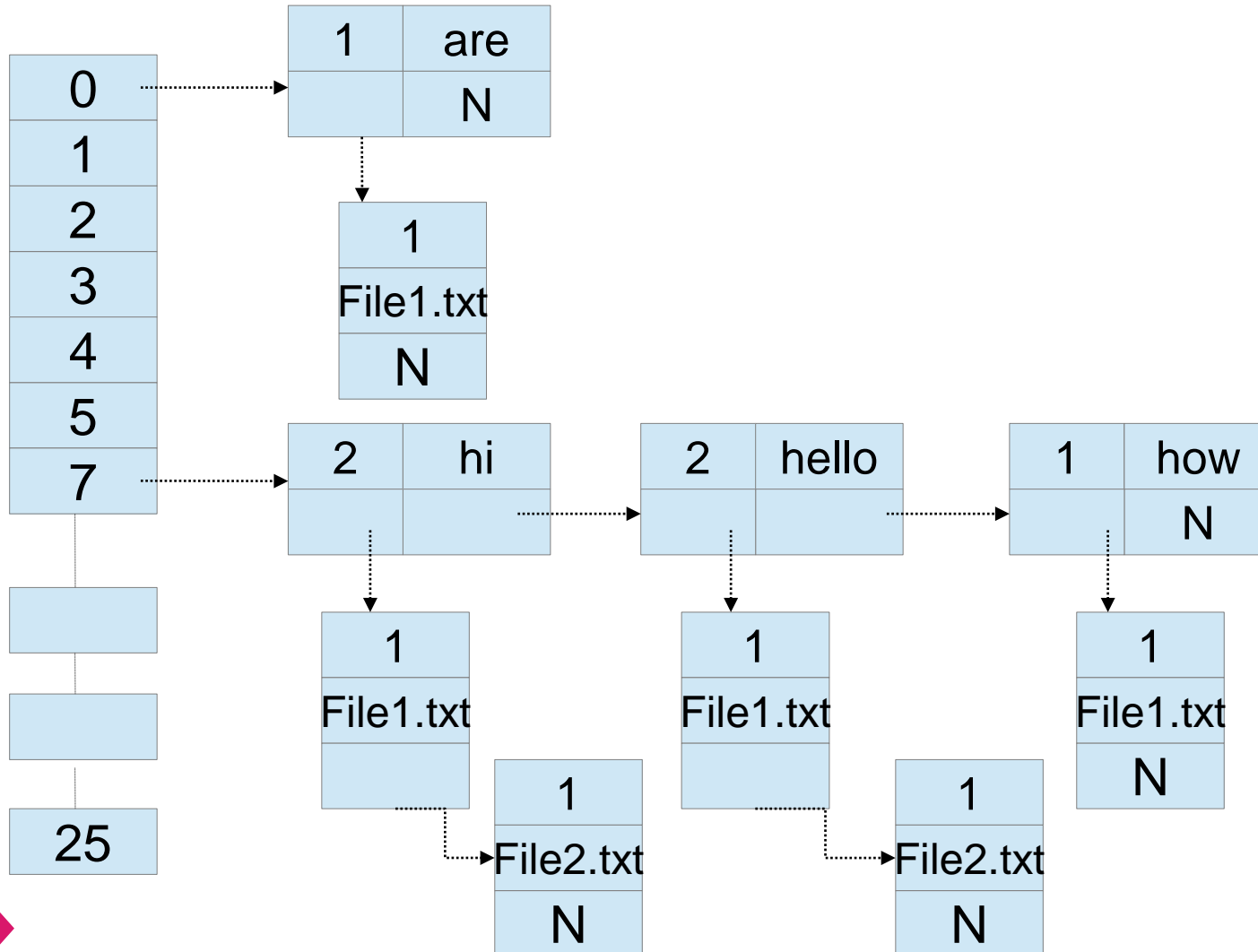
hi hello how are you

File2.txt

hello hi i am fine

Inverted Search

Create Database:



$$\begin{aligned}\text{Index} &= \text{data} \% 97 \\ &= 'h' \% 97 \\ &= 7\end{aligned}$$

File1.txt

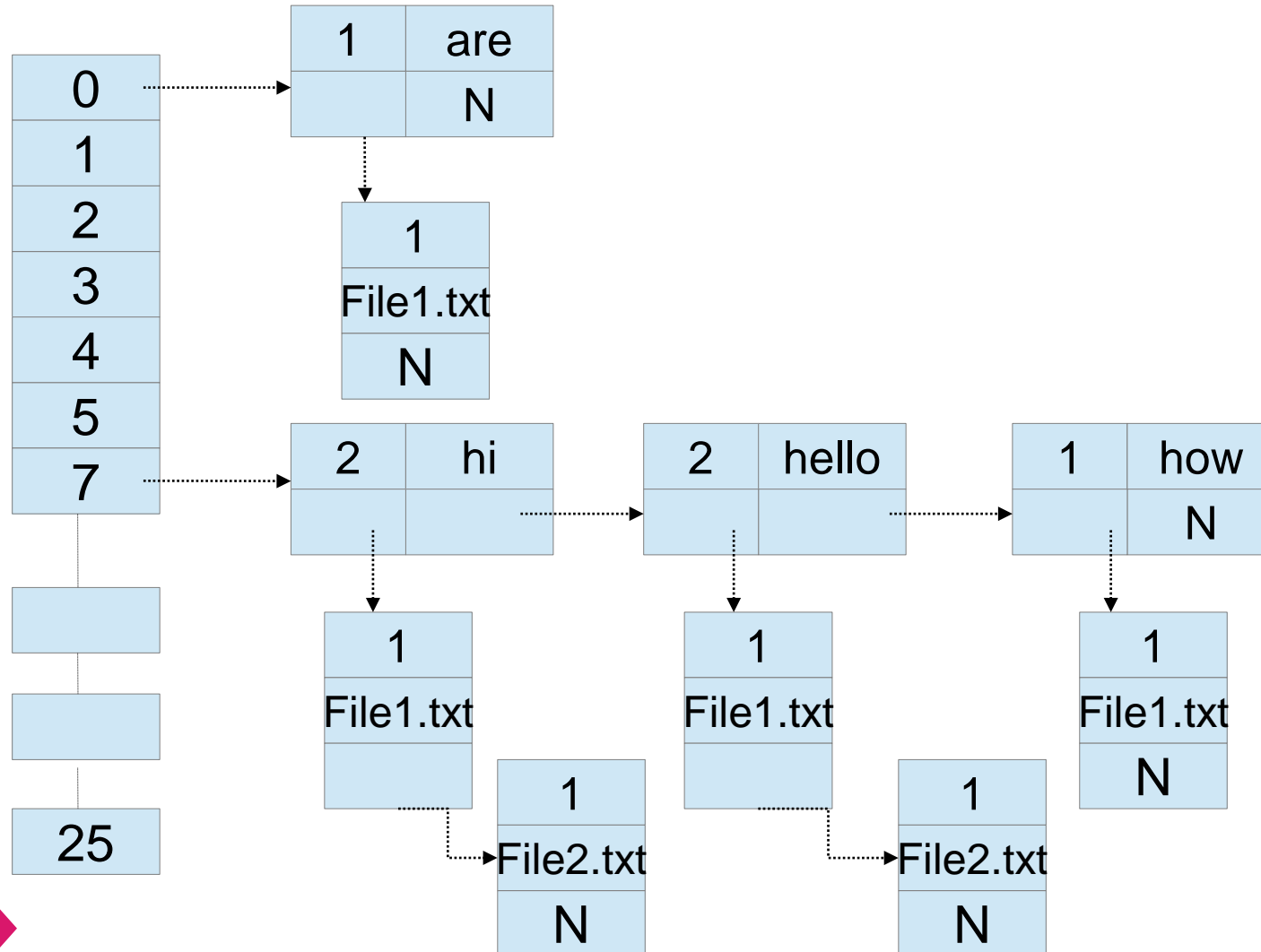
hi hello how are you

File2.txt

hello hi i am fine

Inverted Search

Create Database:



File1.txt

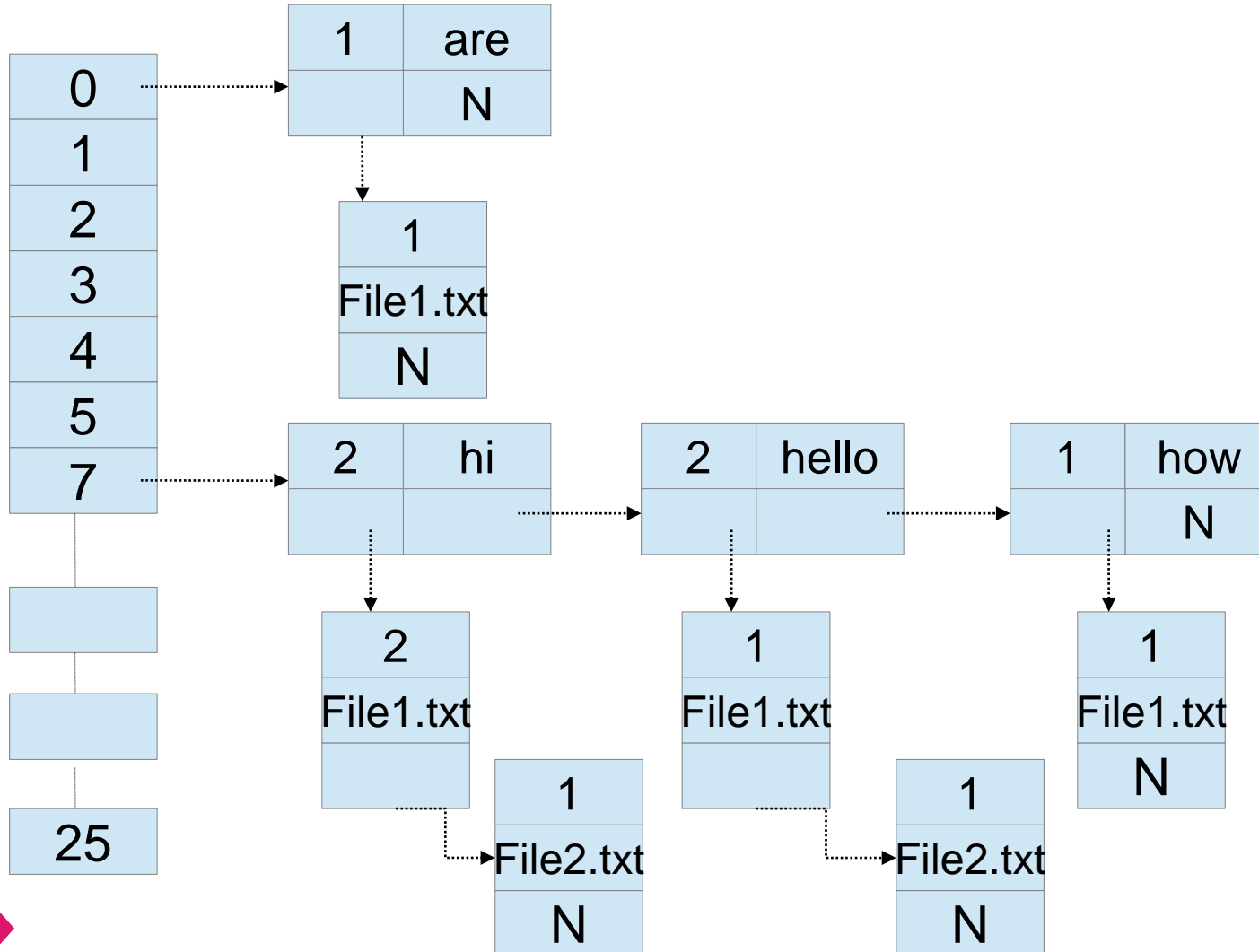
hi hello how are you
hi

File2.txt

hello hi i am fine

Inverted Search

Create Database:



File1.txt

hi hello how are you
hi

File2.txt

hello hi i am fine

Inverted Search



Create Database:

Create nodes:

Sub node :

```
typedef struct sub
{
    int word_count;
    char f_name[FNAME_SIZE];
    struct sub *link;
}subnode_t;
```

Main node :

```
typedef struct mainnode
{
    int file_count;
    char word[WORD_SIZE];
    struct mainnode *link;
    subnode_t *link;
} mainnode_t;
```

Inverted Search



Create Database:

Rules:

- 1 If Words are same and Filenames are also same
-> **Increment word count**
2. If words are same and Filenames are different
-> **Increment file count and allocate memory for table link**
3. If word are different and Filenames are different
-> **Allocate the entire block(main and sub)**
- 4.If words are different and filenames are same
-> **Allocate the entire block(main and sub)**

Inverted Search

Create Database:

Create nodes:

Sub node :

```
typedef struct sub
{
    int word_count;
    char f_name[FNAME_SIZE];
    struct sub *link;
}subnode_t;
```

f_name node :

```
typedef struct file
{
    char filename[WORD_SIZE];
    struct file *link;
}filenames_t;
```

Main node :

```
typedef struct mainnode
{
    int file_count;
    char word[WORD_SIZE];
    struct mainnode *link;
    struct sub *link;
} mainnode_t;
```

Display Database



Inverted Search



Display Database:

Printing Pattern:

1. Search for the index which is not empty.
2. Display the index number and details as follows.
 - > [ind_no] <word> <file_count> <filename> <word_count>

Inverted Search



Display Database:

```
[0]      [are]    1 file(s) : file : File1.txt : 1 time(s)
[7]      [hi]     2 file(s) : file : File1.txt : 2 time(s) : File2.txt : 1 time(s)
          [hello] 2 file(s) : file : File1.txt : 1 time(s) : File2.txt : 1 time(s)
          [how]   1 file(s) : file : File1.txt : 1 time(s)
```



Search Database



Inverted Search



Search Database:

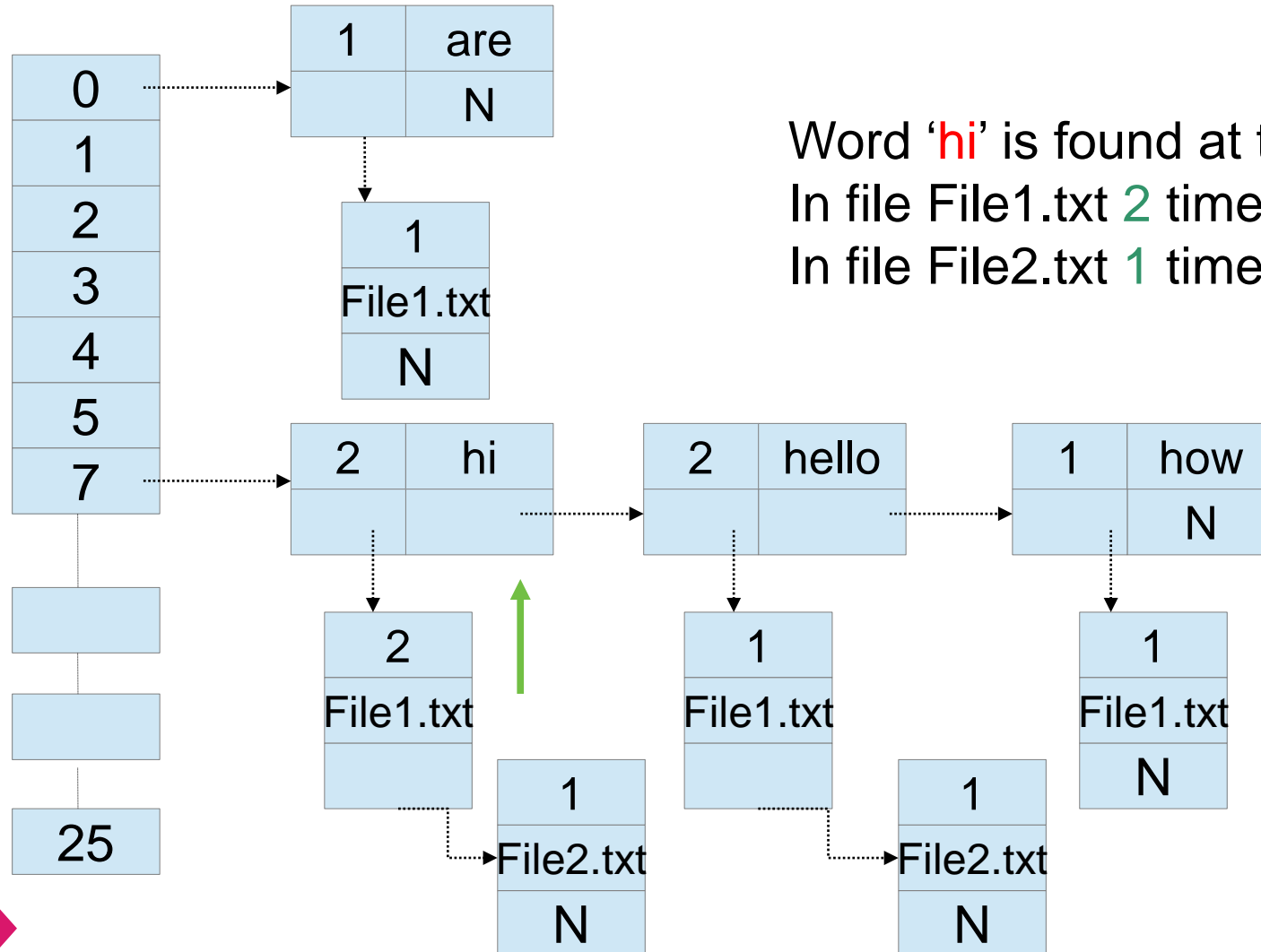
1. Read the word to be searched in Database
2. Example the word is **hi**



Inverted Search

$$\begin{aligned}\text{Index} &= \text{data} \% 97 \\ &= \text{'h'} \% 97 \\ &= 7\end{aligned}$$

Search Database:

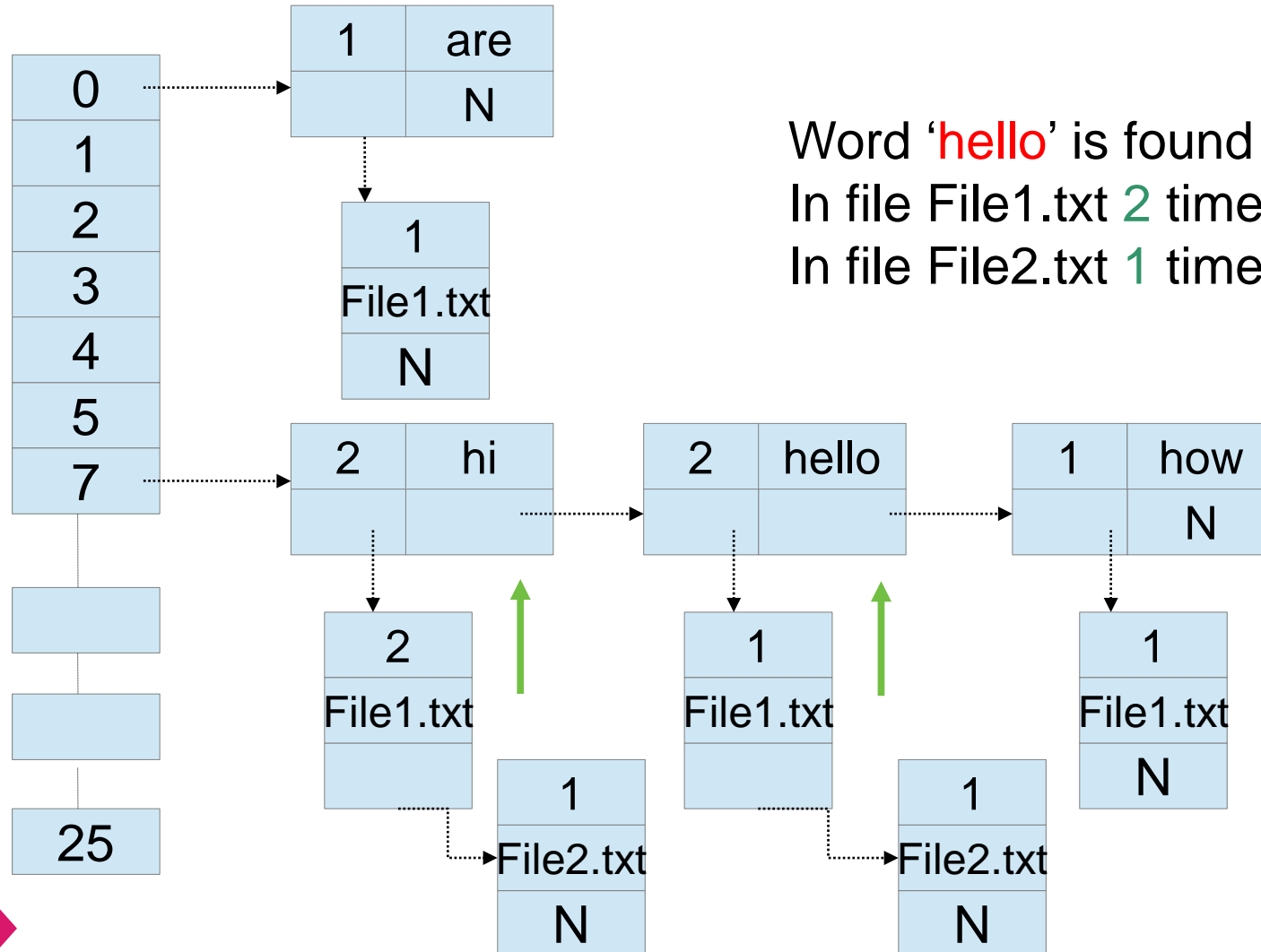


Inverted Search

$$\begin{aligned}\text{Index} &= \text{data} \% 97 \\ &= 'h' \% 97 \\ &= 7\end{aligned}$$

Word to be
searched:
hello

Search Database:



Update Database



Inverted Search

Update Database:

1. Read the file to be added in Database
2. Let's say the file name is file3.txt

File3.txt

how are you

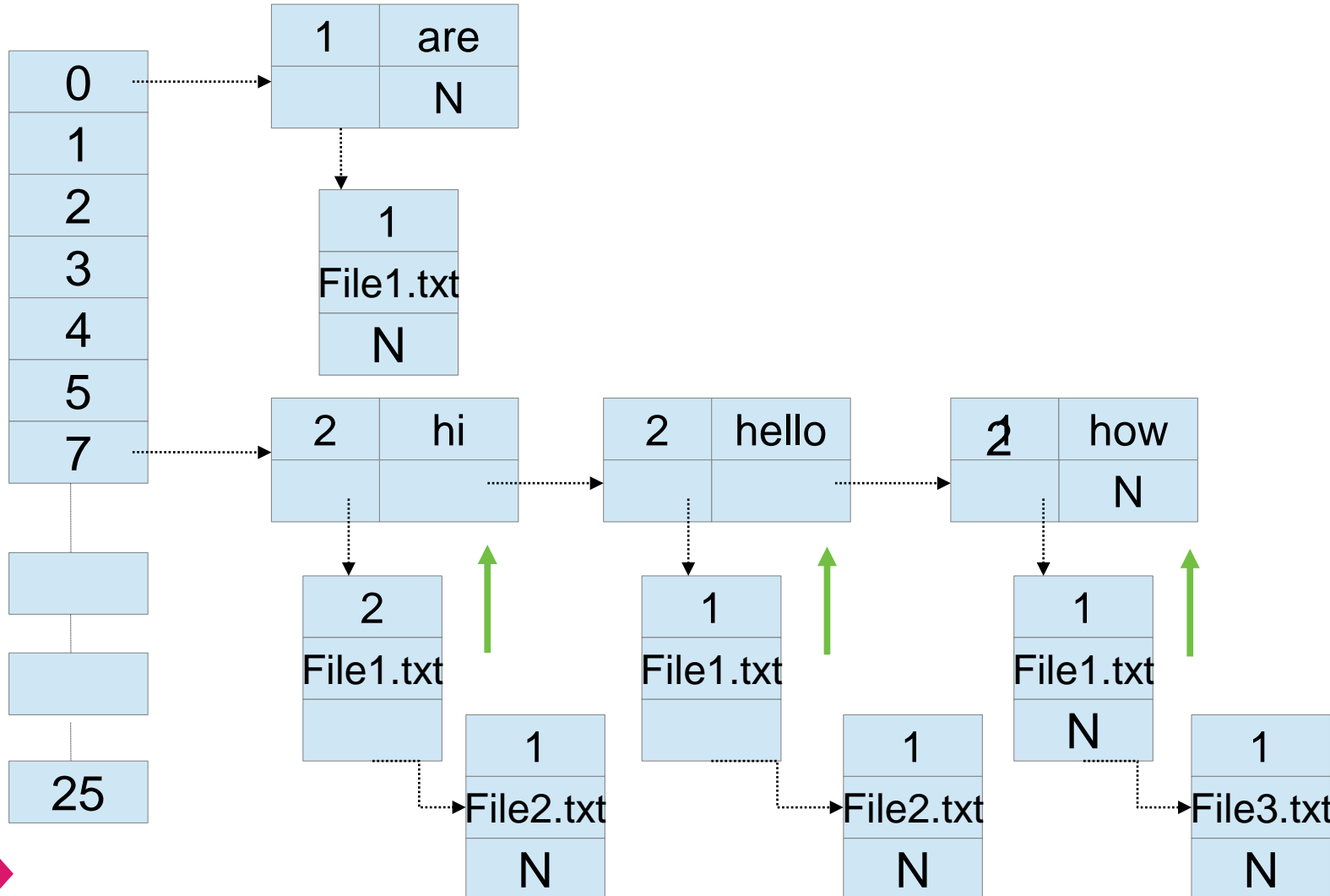
Inverted Search

$$\begin{aligned}\text{Index} &= \text{data} \% 97 \\ &= \text{'h'} \% 97 \\ &= 7\end{aligned}$$

File3.txt

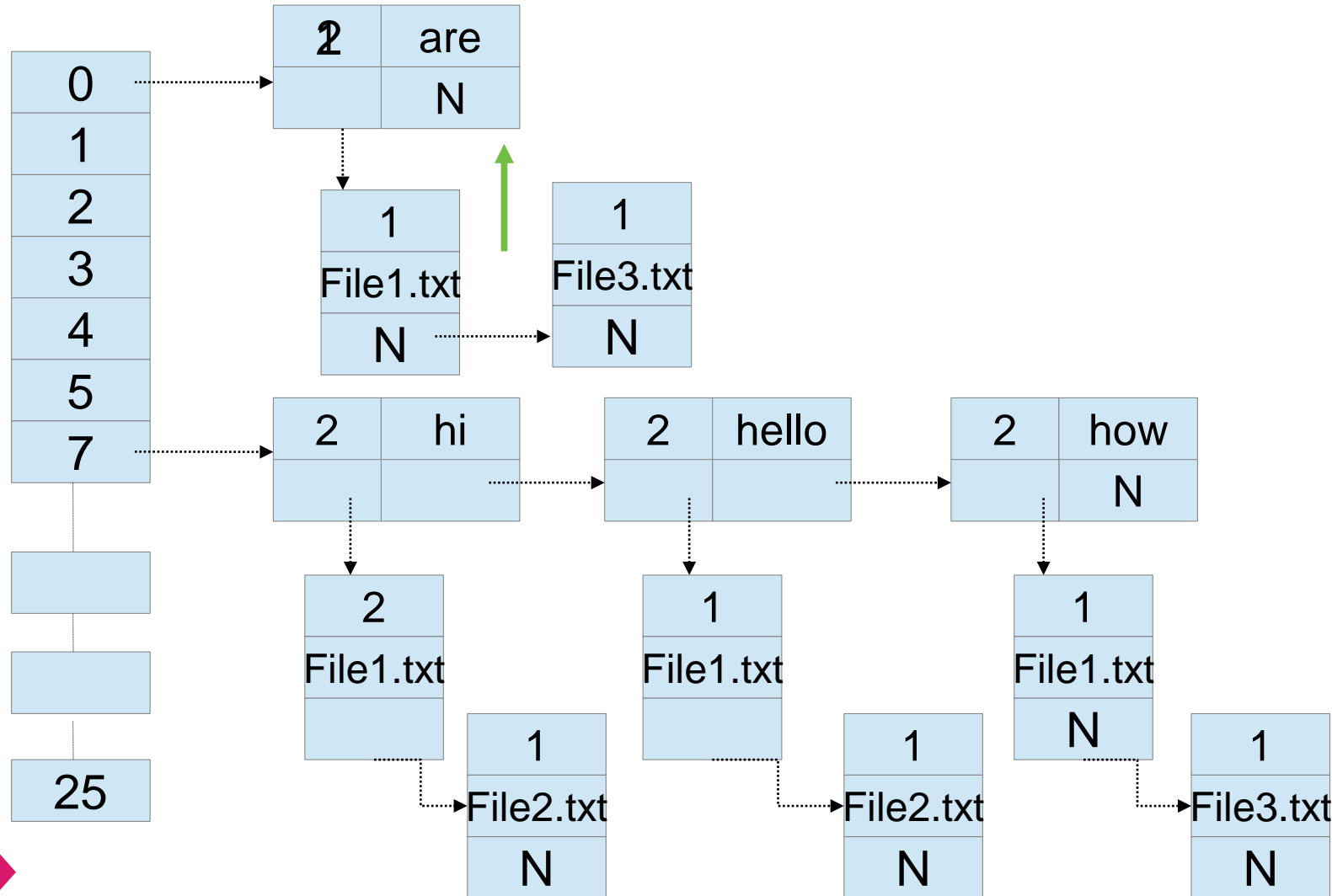
how are you

Update Database:



Inverted Search

Update Database:



$$\begin{aligned}\text{Index} &= \text{data} \% 97 \\ &= 'a' \% 97 \\ &= 0\end{aligned}$$

File3.txt

how are you

Save Database

Inverted Search



Save Database:

1. Read the backup file name.
2. Let's say the file name is backup.txt
3. Store the contents in given pattern.
 1. #<index_no>;
 2. <word>;<file_count>;<file_name>;<word_count>#

Validations



Inverted Search



Validations :

1. Check the filename passed through CL.
 1. If yes continue further.
 2. Else print error and stop
2. If passed then store the filenames in Linked list and check the filenames are different.
 1. If no print error for duplicate filenames.
 2. Check the file is present and it is not empty file.
3. If filenames are different then continue.
4. The above validations applicable for Update DataBase too.
5. For display database, you need to display the index numbers which are not empty.
6. Search Database :
 1. Read the word
 2. Check the word is present
 1. If present then print the details in given pattern.
 2. Else print error.