Project – Inverted Search

Data Structure –Project

Inverted Search

What ?

Data Structure –Project

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.

Data Structure –Project

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.

Data Structure –Project

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.

Data Structure –Project

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

Words	Documents
Hi	Doc1, Doc2
Hello	Doc1
How	Doc1, Doc2
Are	Doc1
You	Doc1
I	Doc2
Am	Doc2
Good	Doc2

Doc1

Hi Hello
How Are You

Doc2

I Am Good
Hi How

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.

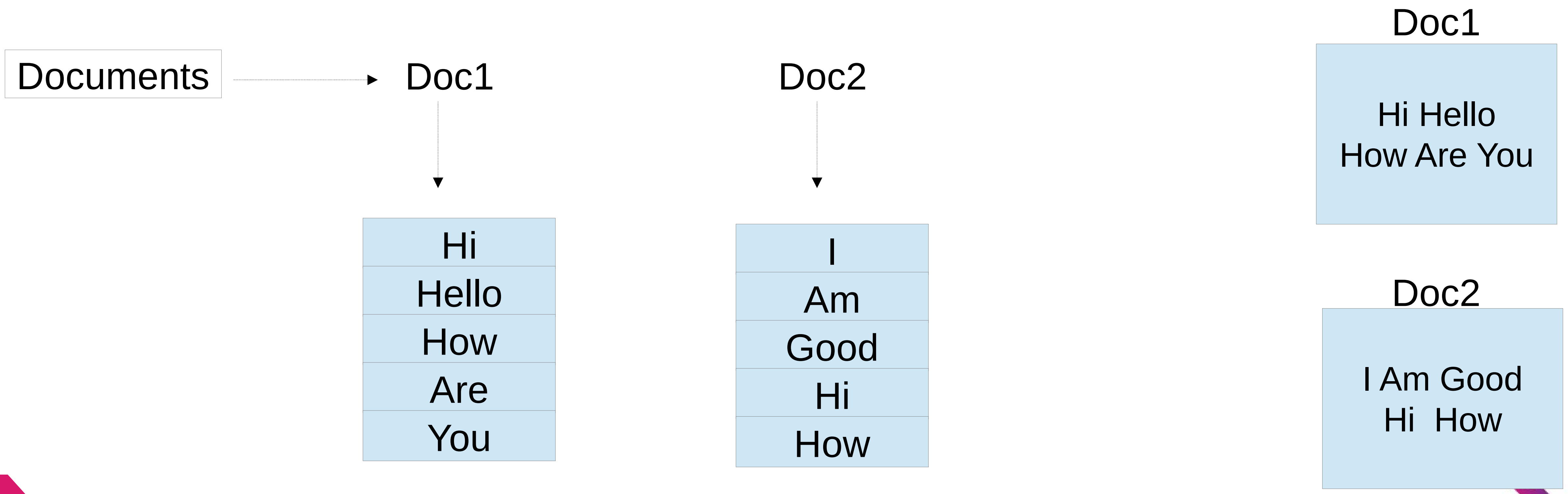
Forward Indexing

- 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



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

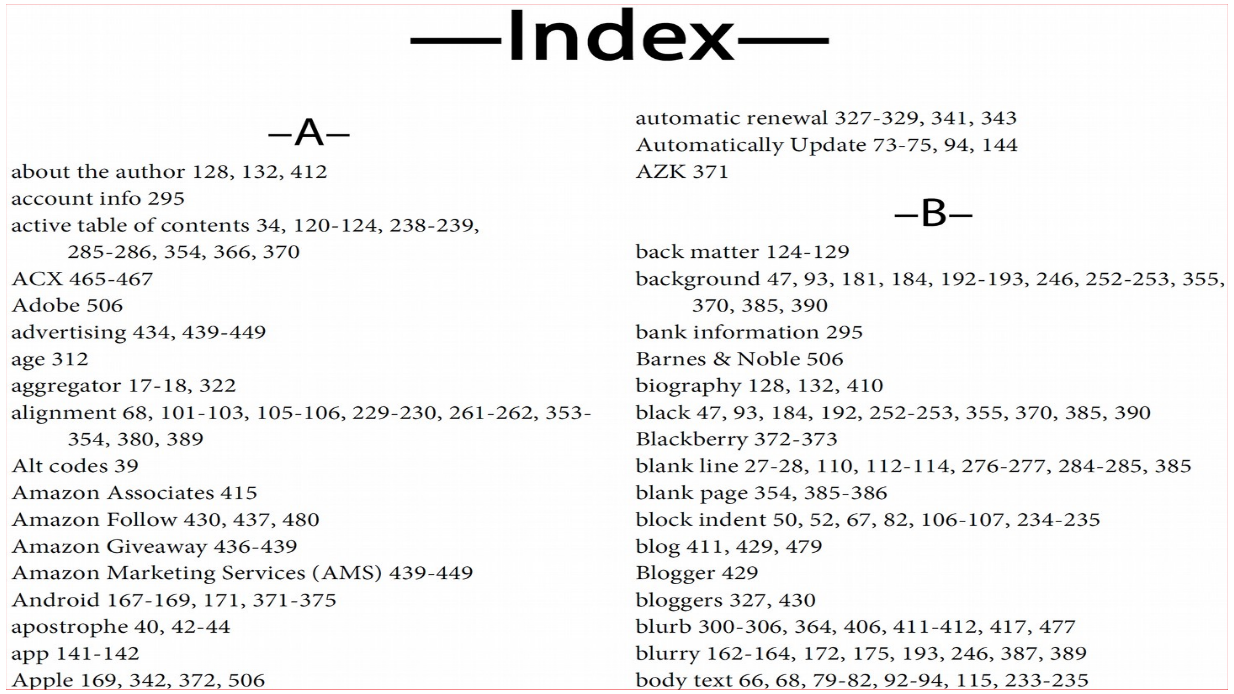
- It is a data structure that stores mapping from documents to words i.e. directs you from document to word.
- **Real life examples of Forward index:**
  - Table of contents in book

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
account info 295
active table of contents 34, 120-124, 238-239, 285-286, 354, 366, 370
ACX 465-467
Adobe 506
advertising 434, 439-449
age 312
aggregator 17-18, 322
alignment 68, 101-103, 105-106, 229-230, 261-262, 353-354, 380, 389
Alt codes 39
Amazon Associates 415
Amazon Follow 430, 437, 480
Amazon Giveaway 436-439
Amazon Marketing Services (AMS) 439-449
Android 167-169, 171, 371-375
apostrophe 40, 42-44
app 141-142
Apple 169, 342, 372, 506
automatic renewal 327-329, 341, 343
Automatically Update 73-75, 94, 144
AZK 371

–B–

back matter 124-129
background 47, 93, 181, 184, 192-193, 246, 252-253, 355, 370, 385, 390
bank information 295
Barnes & Noble 506
biography 128, 132, 410
black 47, 93, 184, 192, 252-253, 355, 370, 385, 390
Blackberry 372-373
blank line 27-28, 110, 112-114, 276-277, 284-285, 385
blank page 354, 385-386
block indent 50, 52, 67, 82, 106-107, 234-235
blog 411, 429, 479
Blogger 429
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

Data Structure –Project

Inverted Search

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

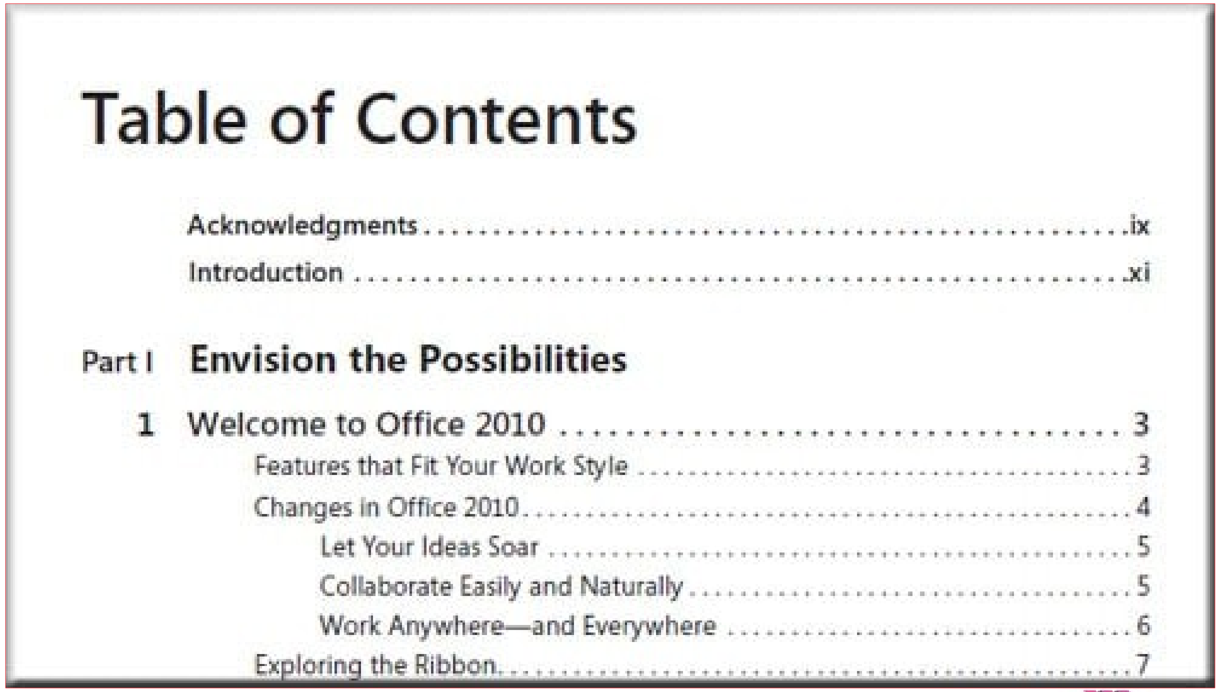


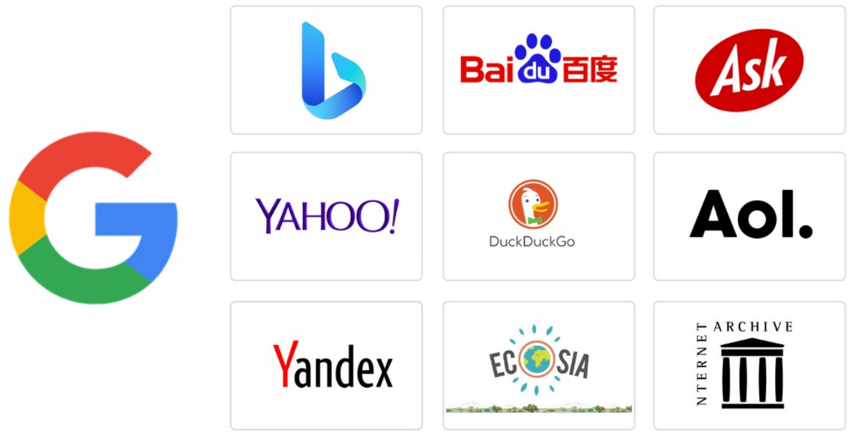
Table of Contents

Data Structure –Project

Inverted Search

Advantages of inverted Index:

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



Data Structure –Project

Inverted Search

Operations:

1.Create Database

2.Display Database

3.Search Database

4.Update Database

5.Save Database

Create Database

Inverted Search

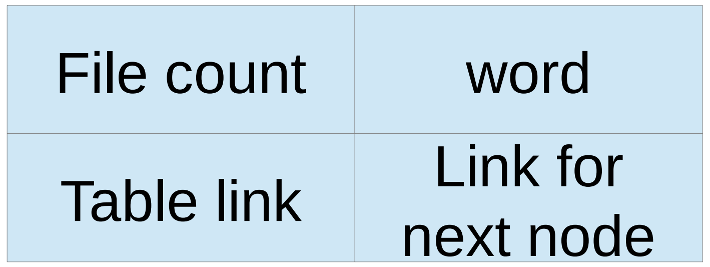
Create Database:

File1.txt

hi hello how are you

File2.txt

hello hi i am fine



Word

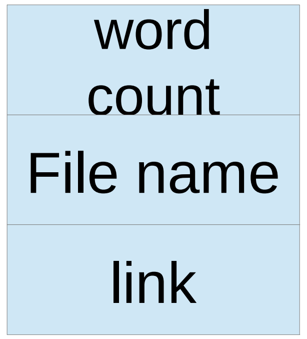


Table link

Data Structure –Project

Inverted Search

Create Database:

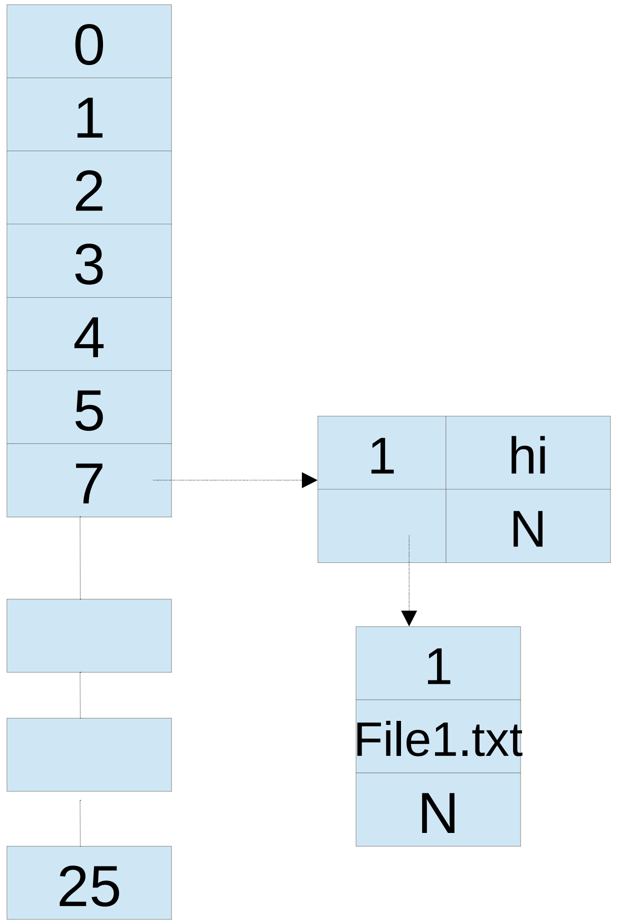
Index = data % 97
= ‘h’ % 97
= 7

File1.txt

hi hello how are you

File2.txt

hello hi i am fine



Data Structure –Project

Inverted Search

Create Database:

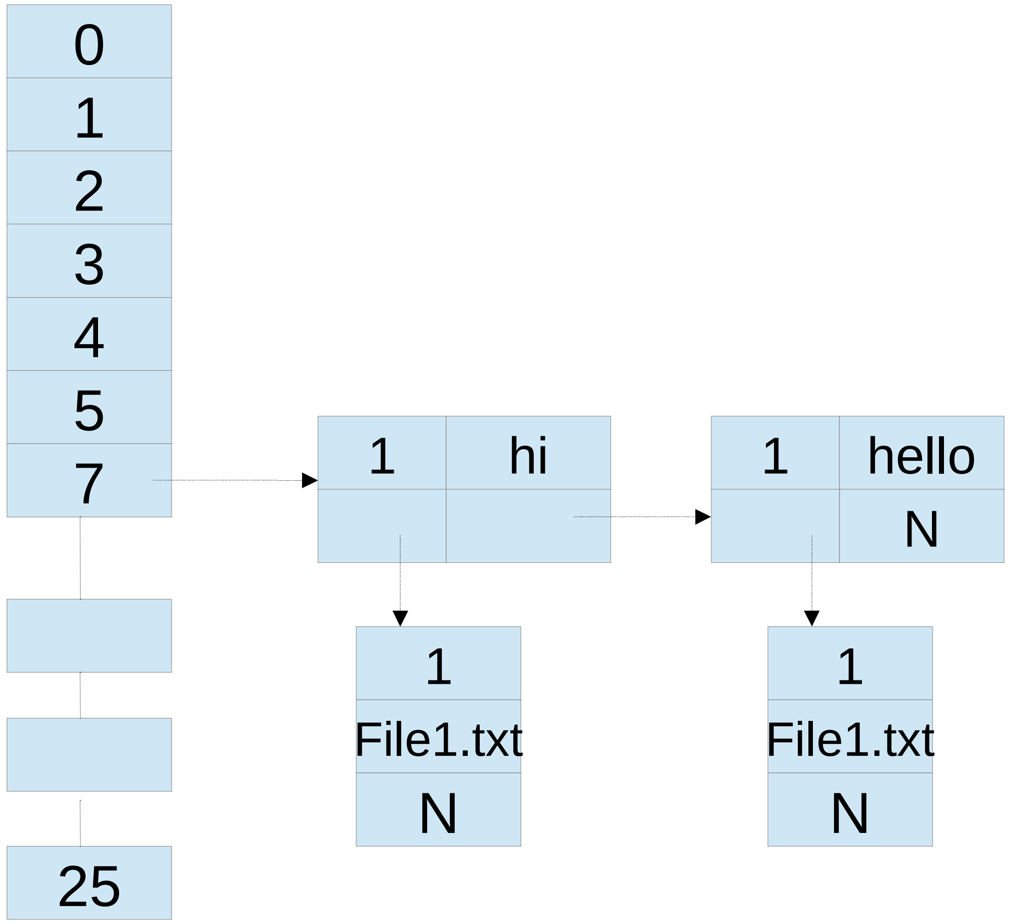
Index = data % 97
= ‘h’ % 97
= 7

File1.txt

hi hello how are you

File2.txt

hello hi i am fine



Data Structure –Project

Inverted Search

Create Database:

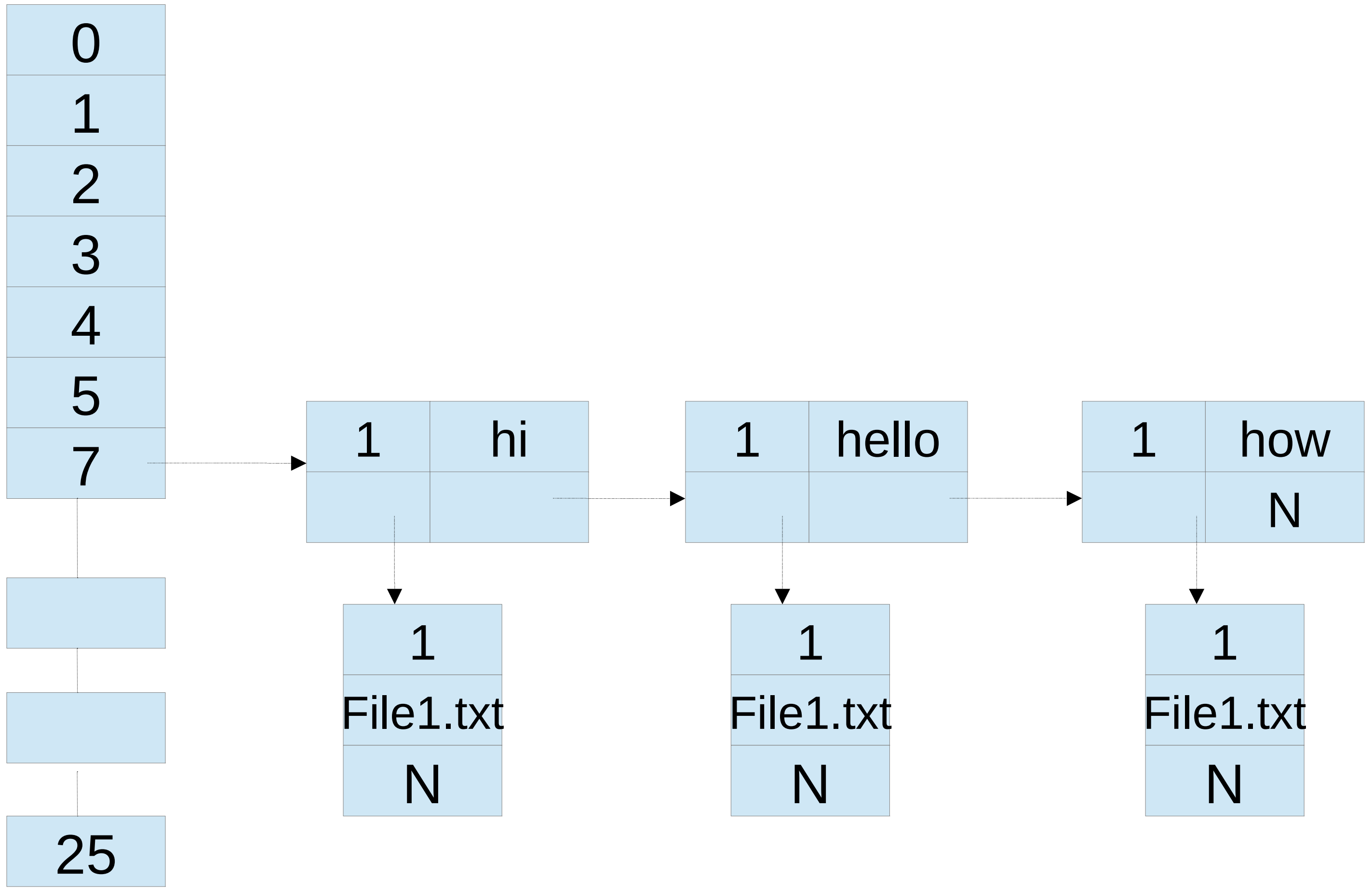
Index = data % 97
= ‘h’ % 97
= 7

File1.txt

hi hello how are you

File2.txt

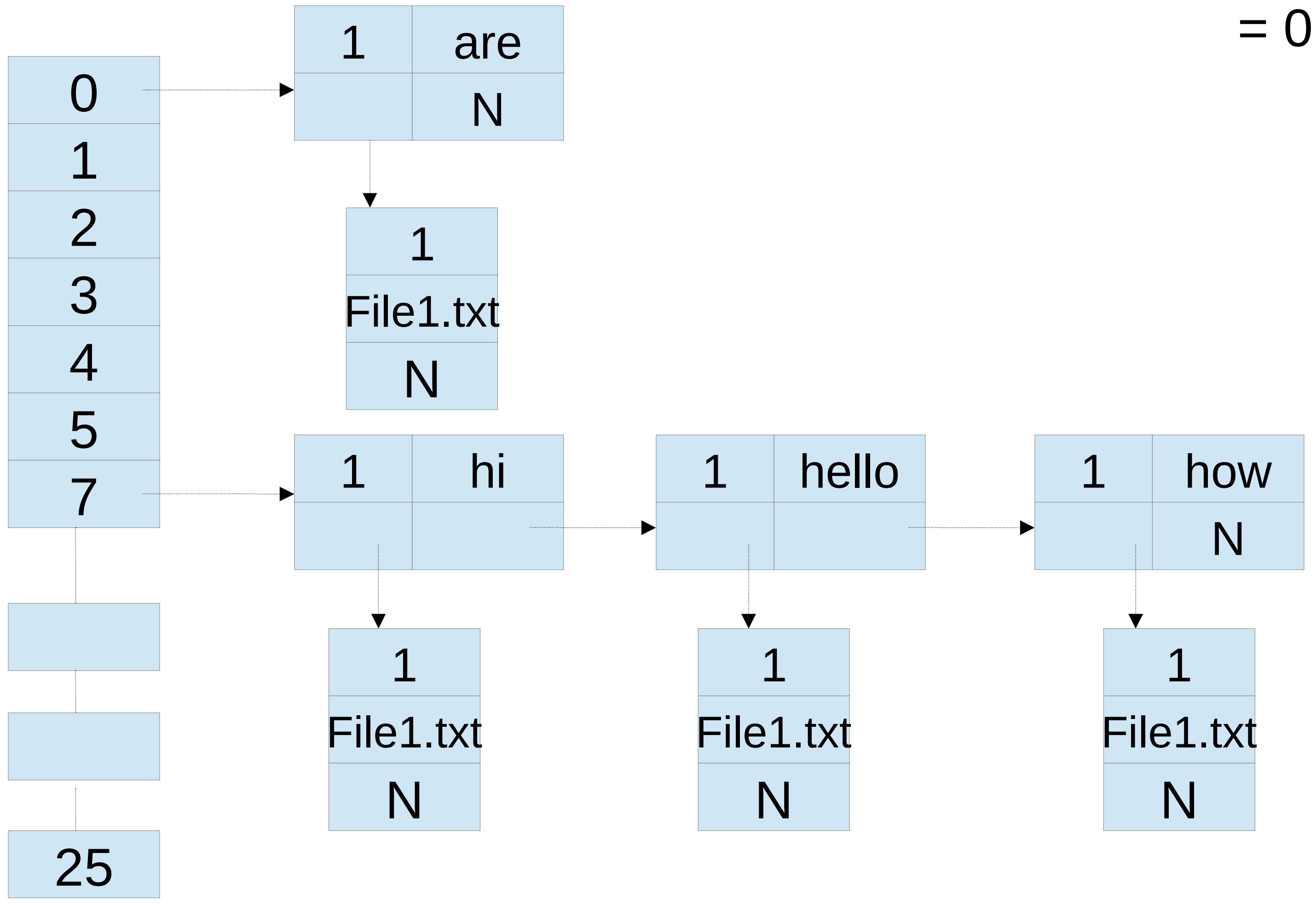
hello hi i am fine



Data Structure –Project

Inverted Search

Create Database:



Index = data % 97
= ‘a’ % 97
= 0

File1.txt

hi hello how are you

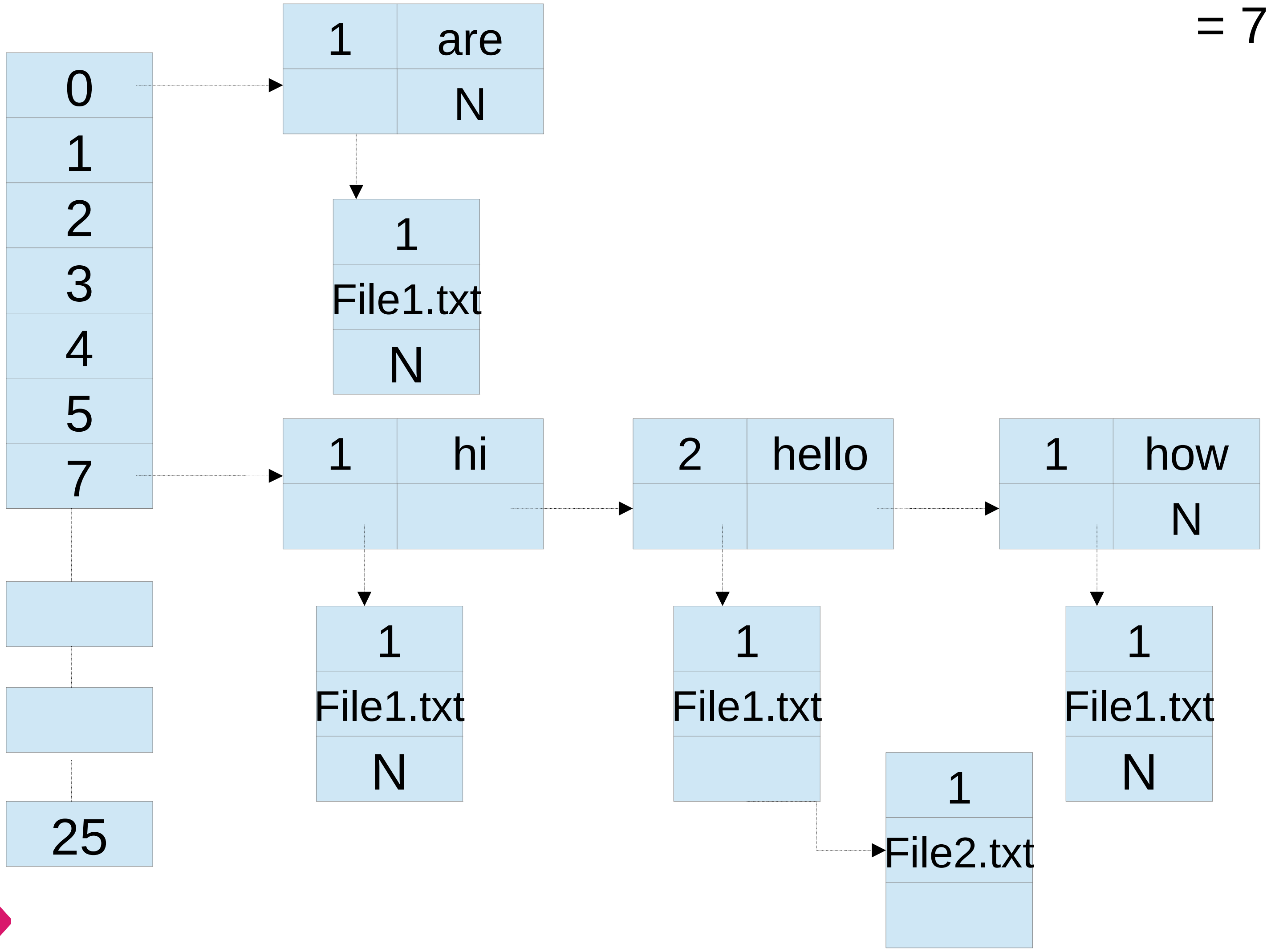
File2.txt

hello hi i am fine

Data Structure –Project

Inverted Search

Create Database:



Index = data % 97
= ‘h’ % 97
= 7

File1.txt

hi hello how are you

File2.txt

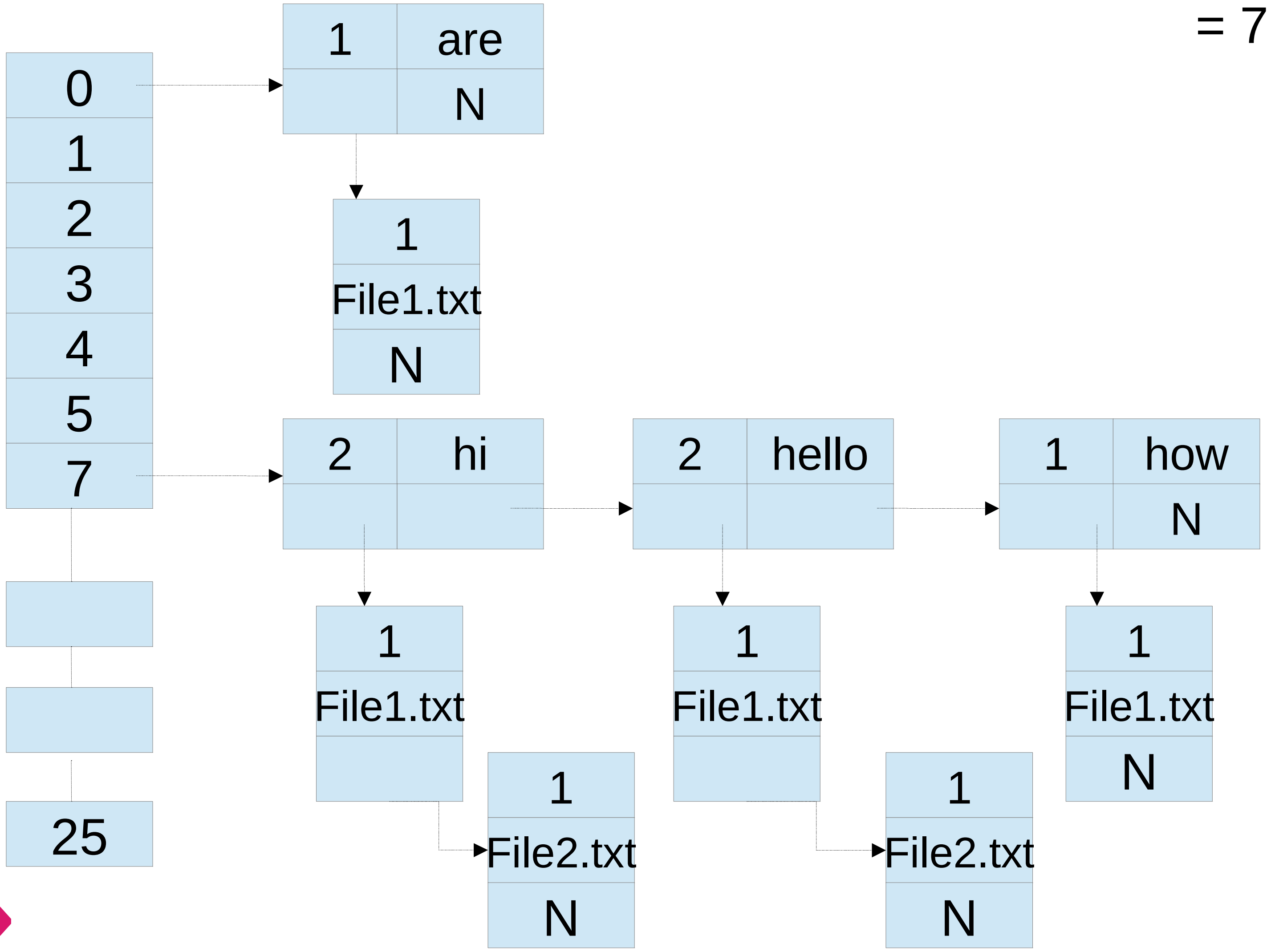
hello hi i am fine

Data Structure –Project

Inverted Search

Create Database:

Index = data % 97
= ‘h’ % 97
= 7



File1.txt

hi hello how are you

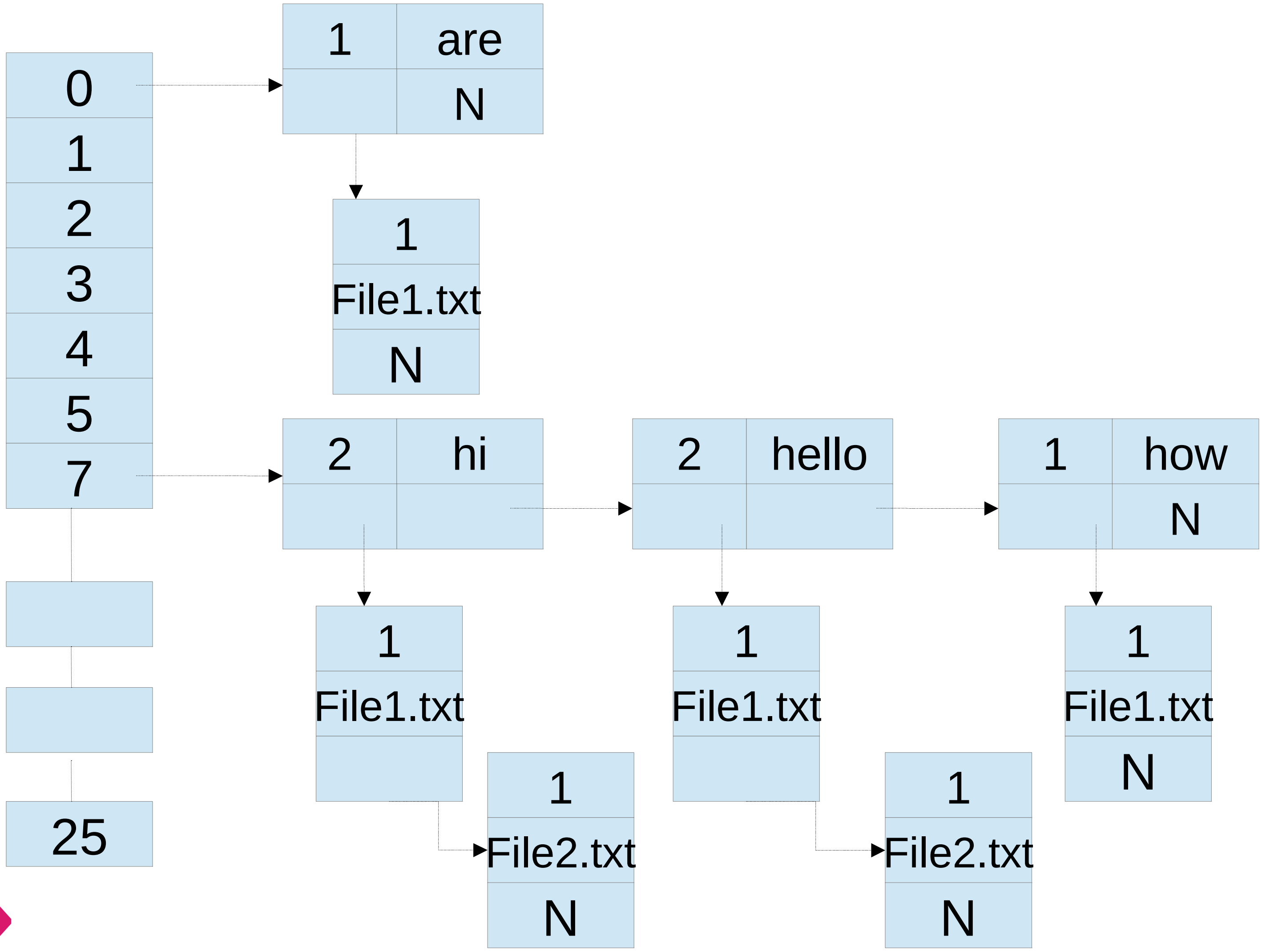
File2.txt

hello hi i am fine

Data Structure –Project

Inverted Search

Create Database:



File1.txt

hi hello how are you hi

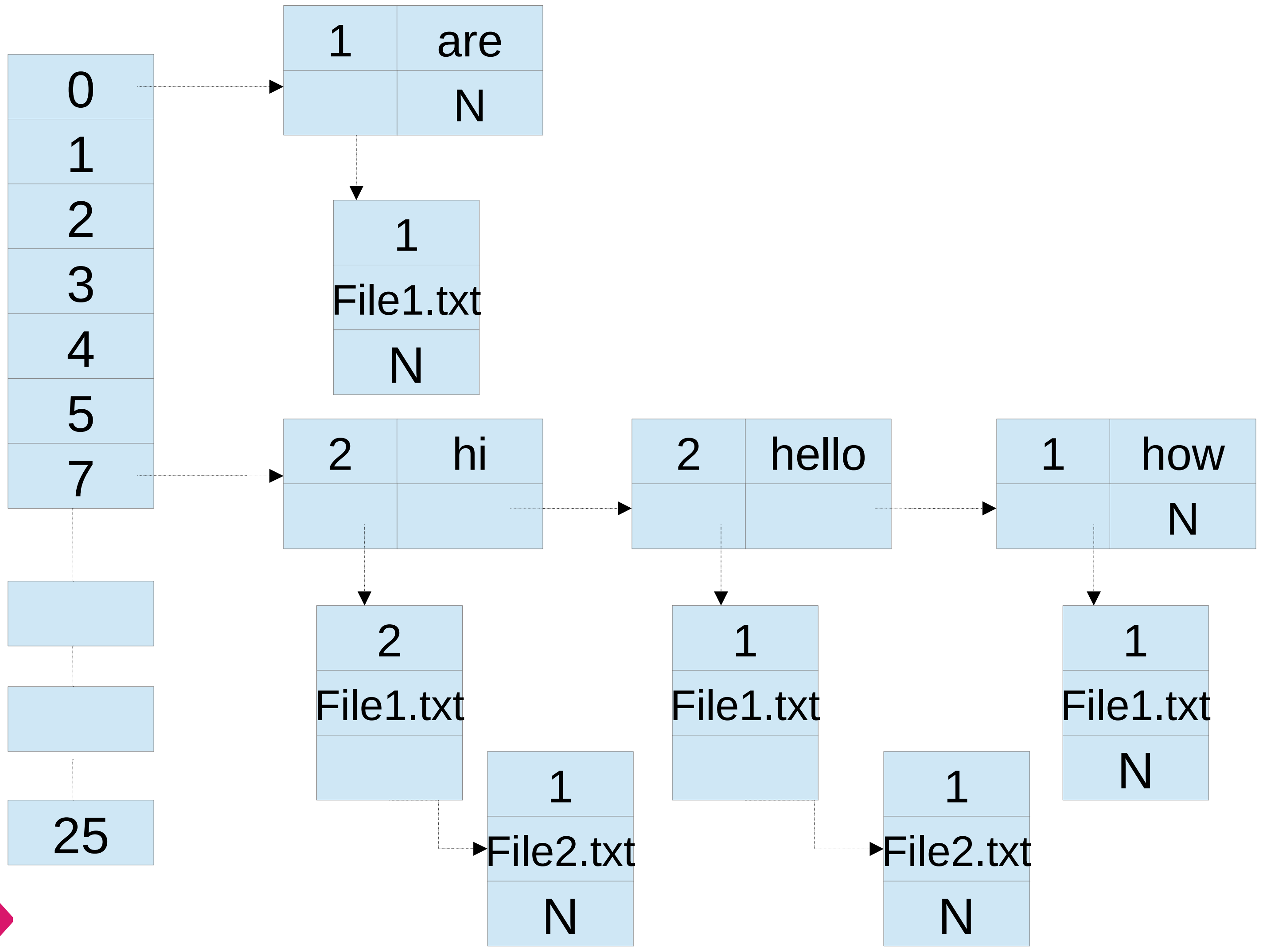
File2.txt

hello hi i am fine

Data Structure –Project

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:

Table node :

```
typedef struct table_node
{
    int word_count;
    char f_name[FNAME_SIZE];
    struct table *link;
}table_node_t;
```

Word node:

```
typedef struct word_node
{
    int file_count;
    char word[WORD_SIZE];
    struct word_node *link;
    struct table_node *t_link;
} word_node_t;
```

Data Structure –Project

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(word and table link)
4.If words are different and filenames are same
    -> Allocate the entire block(word and table link)

Data Structure –Project

Inverted Search

Create Database:

Create nodes:

Table node :

```
typedef struct table_node
{
      int word_count;
      char f_name[FNAME_SIZE];
      struct table_node *link;
}table_node_t;
```

f_name node :

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

Word node:

```
typedef struct word_node
{
      int file_count;
      char word[WORD_SIZE];
      struct word_node *link;
      struct table *t_link;
} word_node_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>

Data Structure –Project

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:

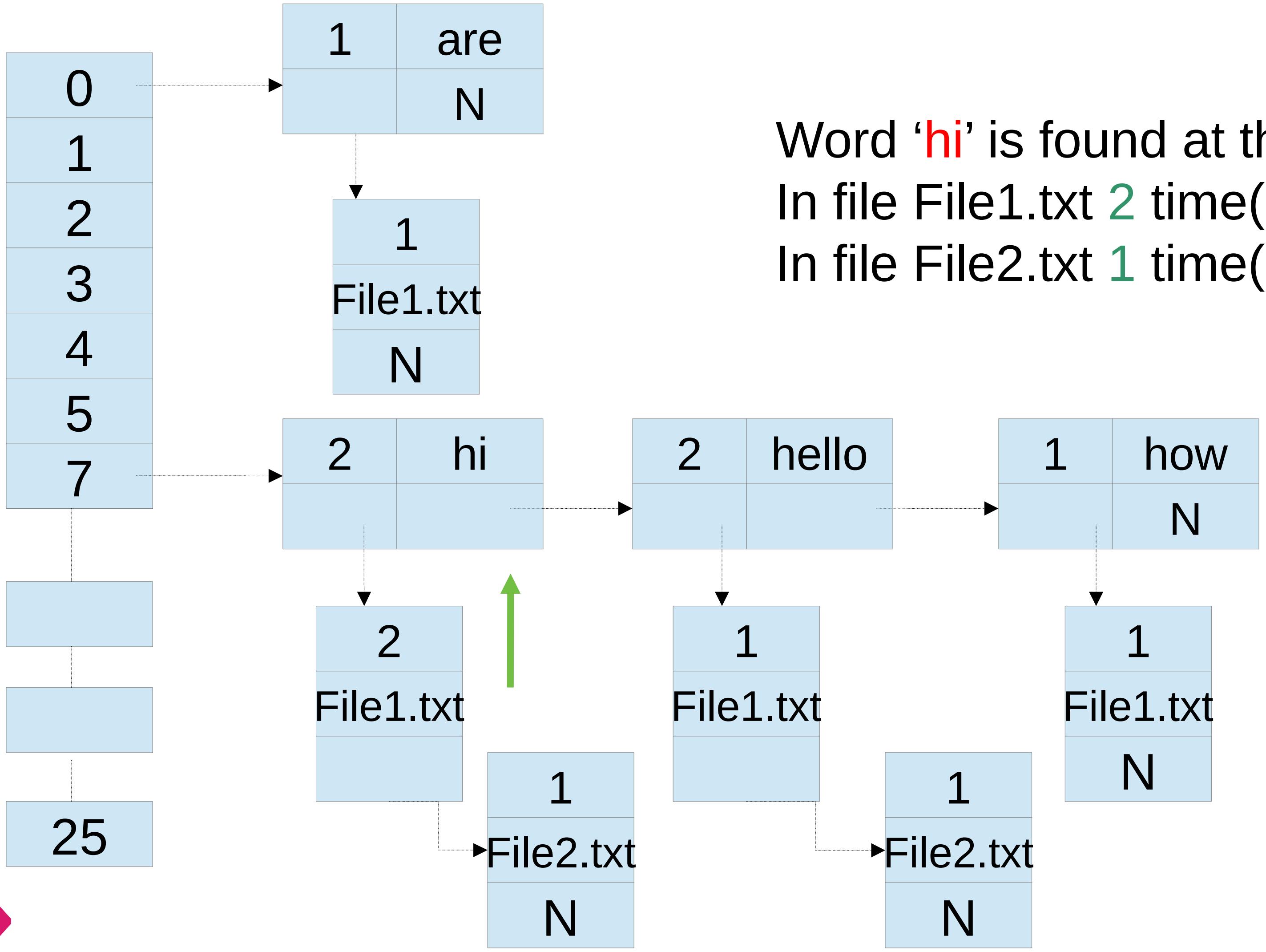
1. Read the word to be searched in Database
2. Example the word is hi

Data Structure –Project

Inverted Search

Search Database:

Index = data % 97
= ‘h’ % 97
= 7



Word ‘hi’ is found at the database and it present in 2 files
In file File1.txt 2 time(s)
In file File2.txt 1 time(s)

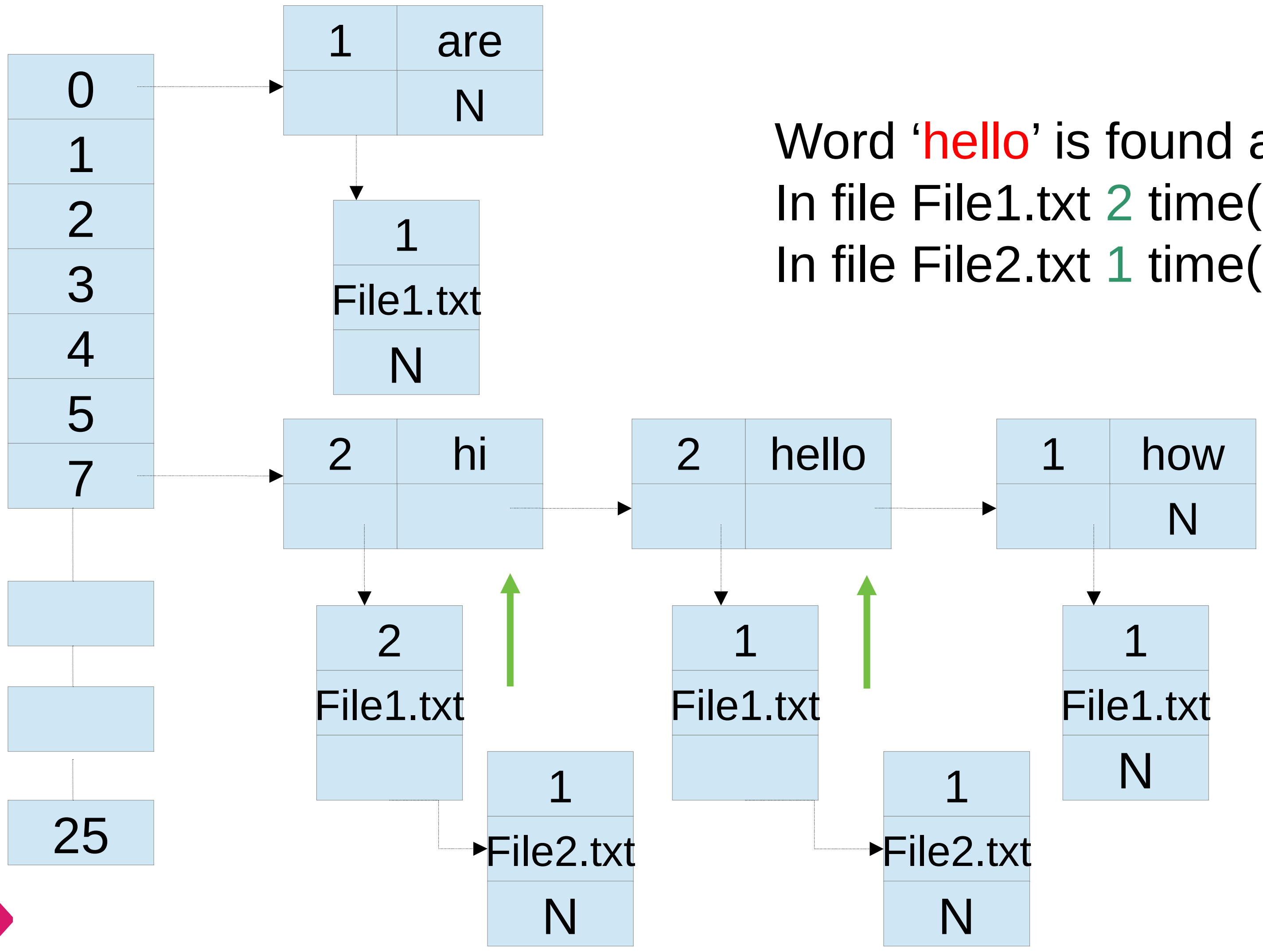
Data Structure –Project

Inverted Search

Word to be searched: hello

Index = data % 97
= ‘h’ % 97
= 7

Search Database:



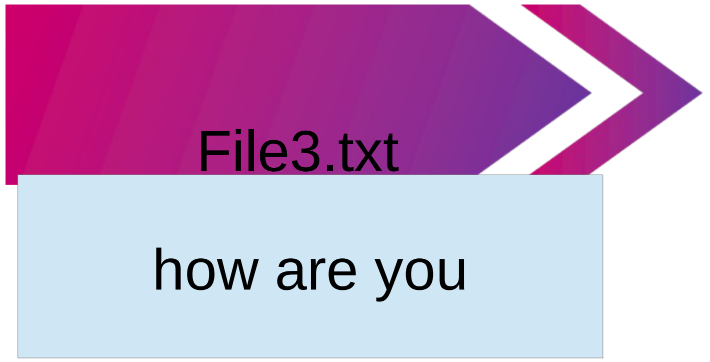
Word ‘hello’ is found at the database and it present in 2 files
In file File1.txt 2 time(s)
In file File2.txt 1 time(s)

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



Data Structure –Project

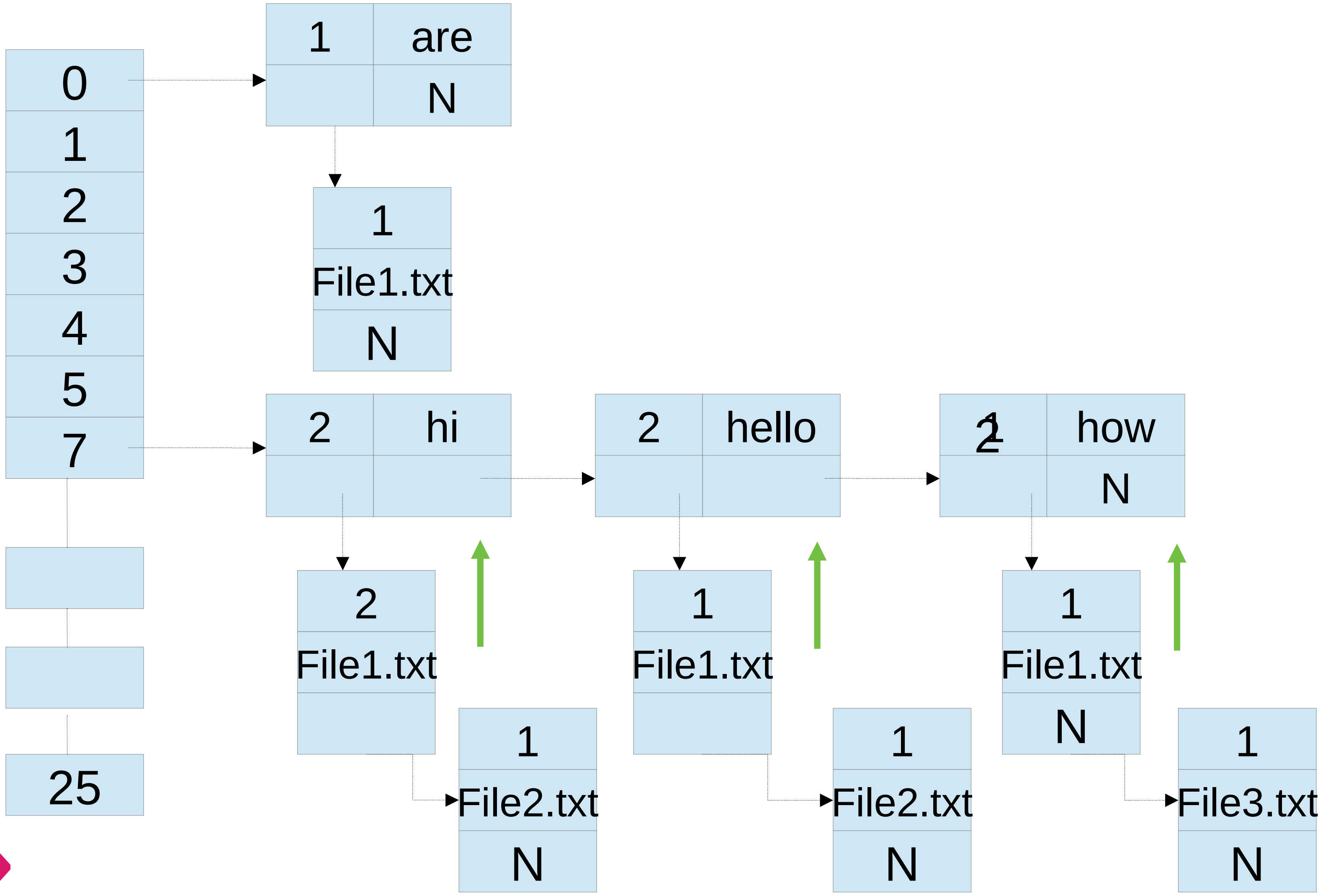
Inverted Search

Index = data % 97
= ‘h’ % 97
= 7

File3.txt

how are you

Update Database:

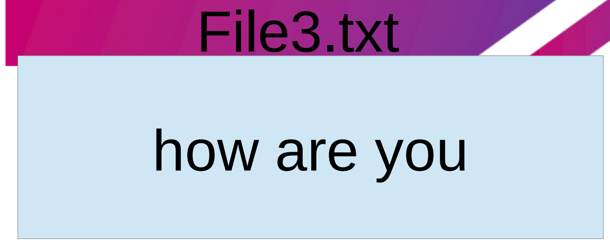


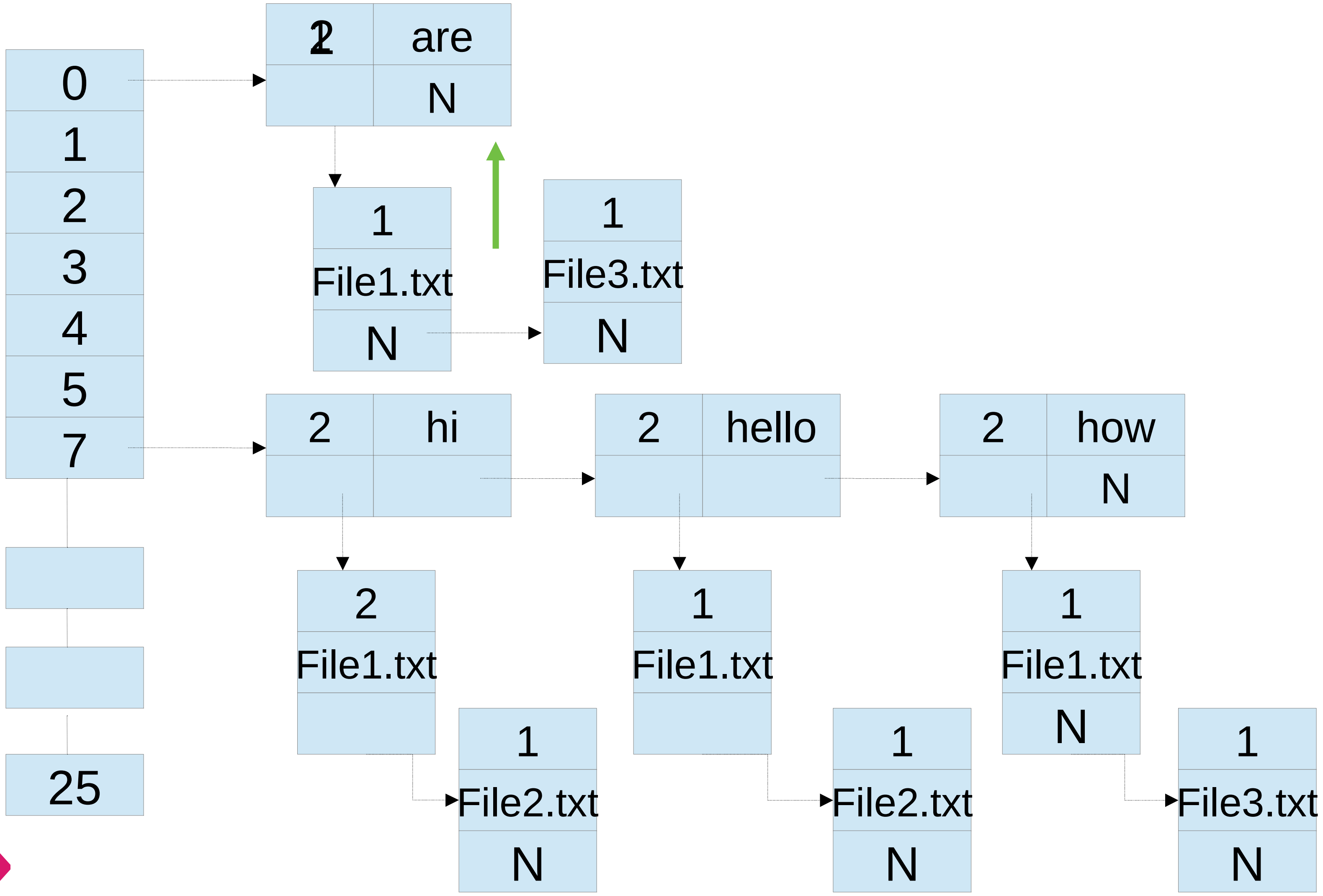
Data Structure –Project

Inverted Search

Update Database:

Index = data % 97
= ‘a’ % 97
= 0





Save Database

Data Structure –Project

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

Data Structure –Project

Inverted Search

Validations :

1. Check the filename passed through CL.
   1. If yes continue furthur.
   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.