File Organization

Kuan-Yu Chen (陳冠宇)

2018/12/19 @ TR-212, NTUST

Review

- Collisions occur when the hash function maps two different keys to the same location

- A method used to solve the problem of collision, also called **collision resolution technique**, is applied
  - Open addressing
    - linear probing, quadratic probing, double hashing, and rehashing
  - Chaining

field⟶record ⟶file ⟶directory

File.

- Every file contains data which can be organized in a hierarchy to present a systematic organization

- The data hierarchy includes data items such as **fields**, **records**, **files**, and **directory**
  - A data field is an elementary unit that stores a single fact
    - A data field is usually characterized by its type and size
    - For example, student's name is a data field that stores the name of students

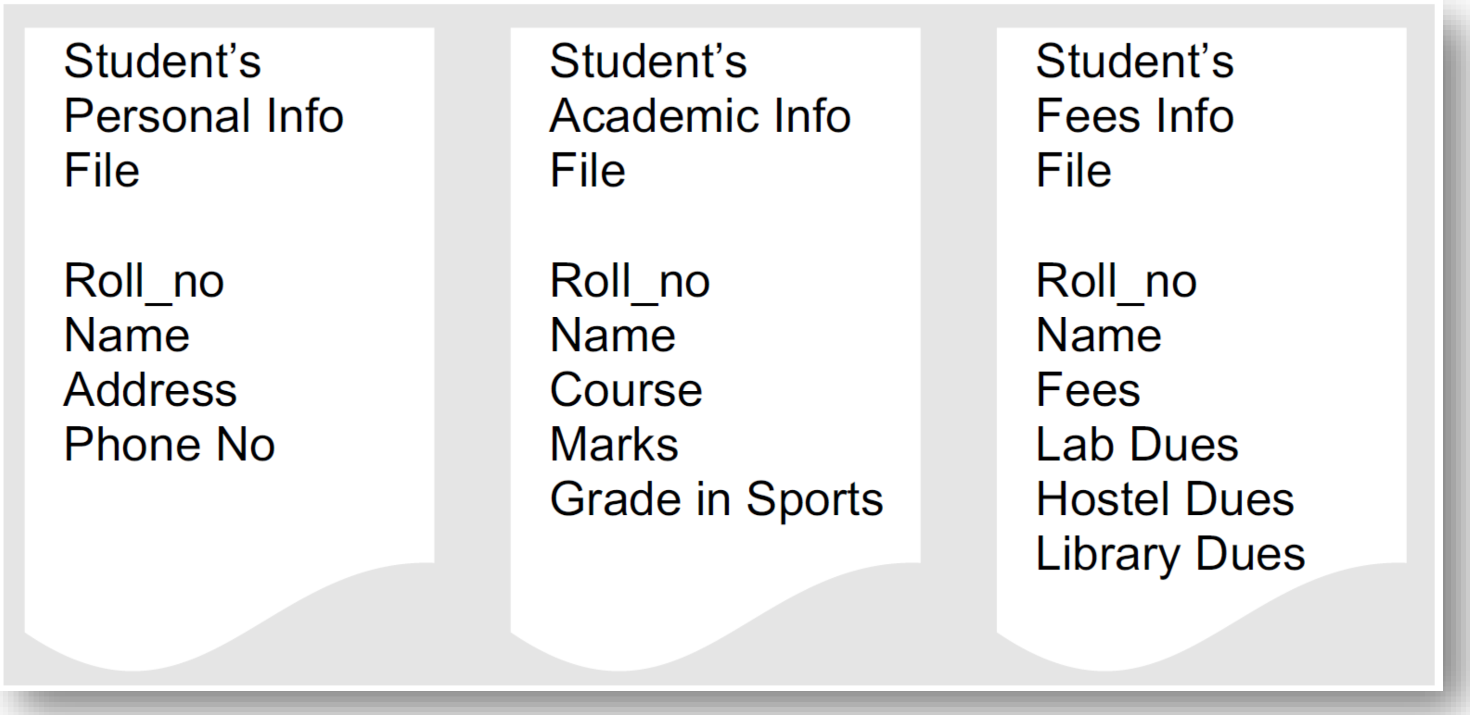
      This field is of type *character* and its size can be set to a maximum of 20 or 30 characters
  - A record is a collection of related data fields which is seen as a single unit from the application point of view
    - For example, the student's record may contain data fields such as name, address, phone number, roll number, marks obtained, and so on

field→record →file →directory

File..

- A file is a collection of related records
  - For example, if there are 60 students in a class, then there are 60 records

- A directory stores information of related files
  - A directory organizes information so that users can find it easily



Student's
Personal Info
File

Roll_no
Name
Address
Phone No

Student's
Academic Info
File

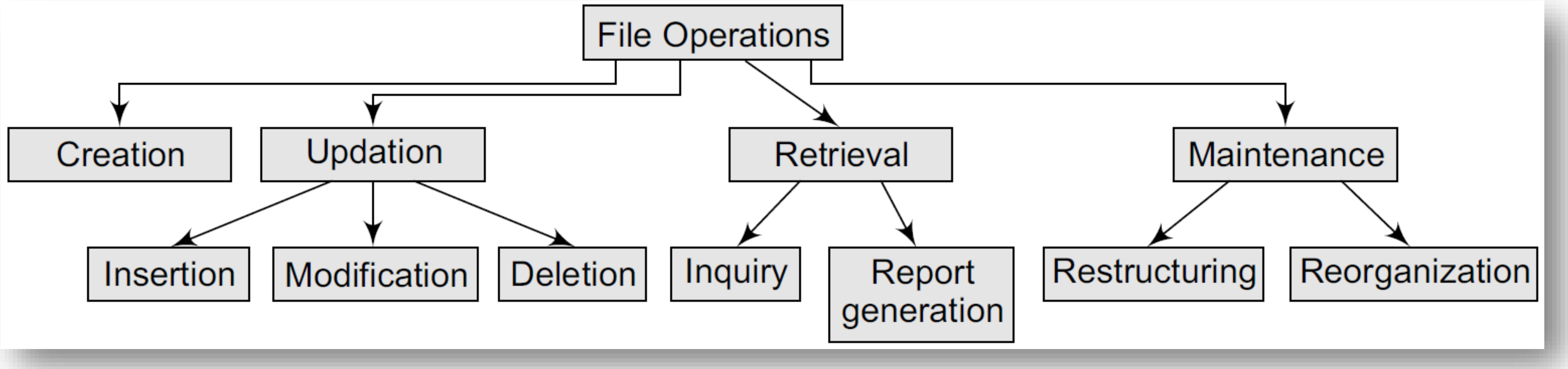
Roll_no
Name
Course
Marks
Grade in Sports

Student's
Fees Info
File

Roll_no
Name
Fees
Lab Dues
Hostel Dues
Library Dues

File Operations & Organizations

- The basic file operations

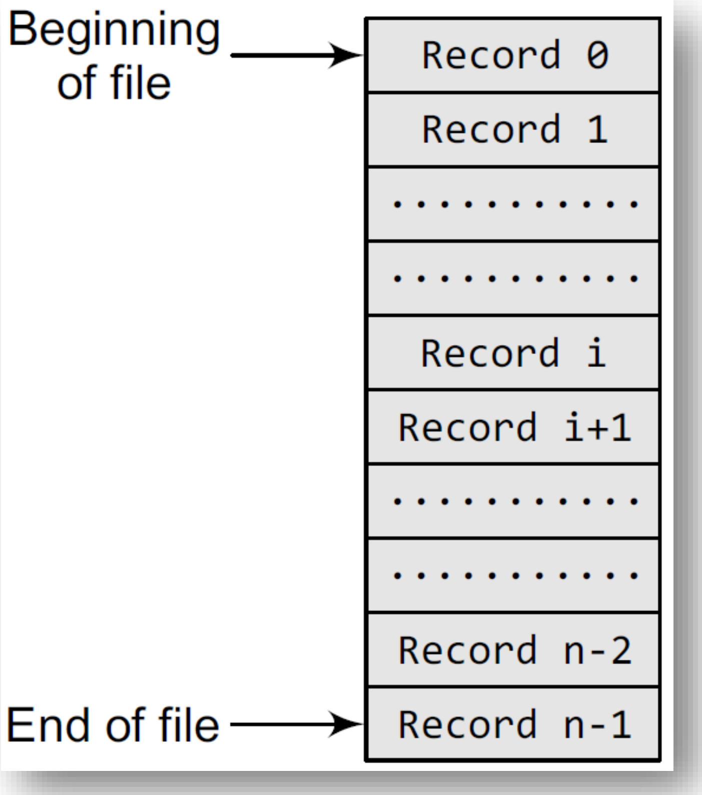


- Organization of records means the **logical** arrangement of records in the file and not the **physical** layout of the file as stored on a storage media
  - Rapid access to one or more records
  - Ease of inserting/updating/deleting one or more records without disrupting the speed of accessing record(s)
  - Efficient storage of records
  - Using redundancy to ensure data integrity

field→record →file →directory

File Organization.

- Techniques that are commonly used for file organization
  - **Sequential Organization**
    - A sequentially organized file stores the records in the order in which they were entered
    - Sequential files can be read only sequentially, starting with the first record in the file

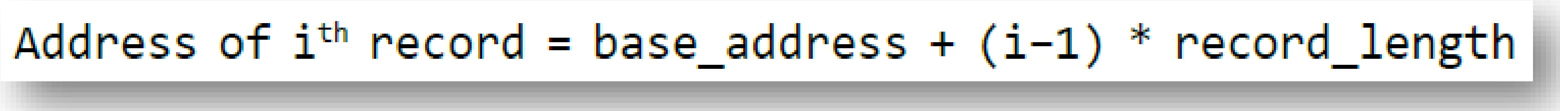


File Organization..

– Relative File Organization

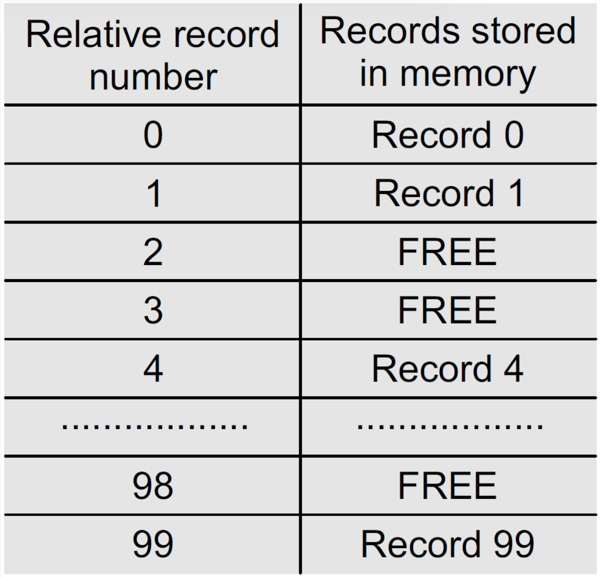
- Relative file organization provides an effective way to access individual records directly
- Records are ordered by their **relative key**

  Relative key represents the location of the record relative to the beginning of the file



```
Address of ith record = base_address + (i-1) * record_length
```

- Records in a relative file are of fixed length



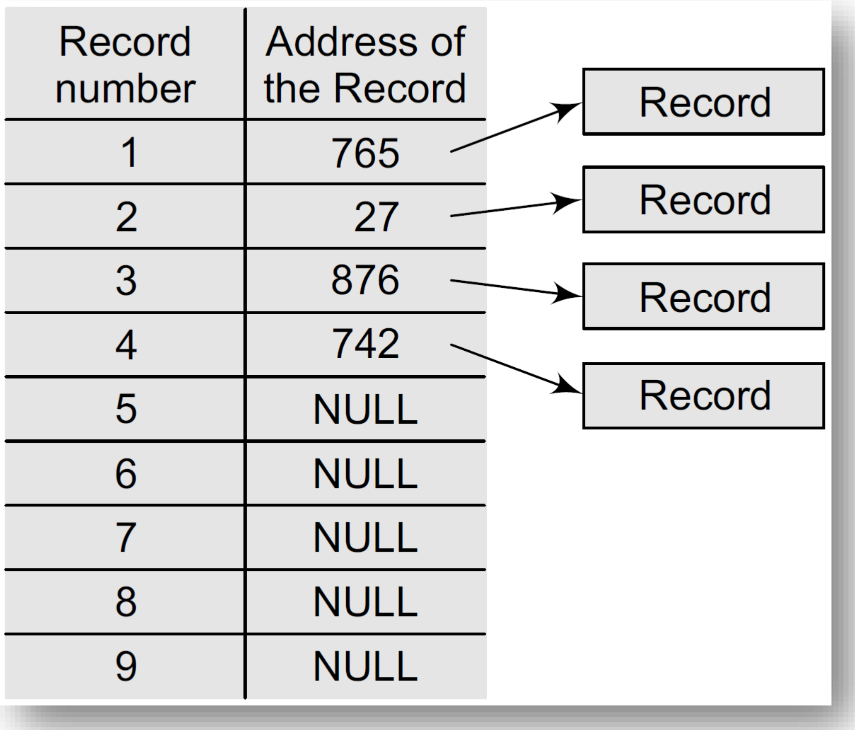
Relative record number	Records stored in memory
0	Record 0
1	Record 1
2	FREE
3	FREE
4	Record 4
....	....
98	FREE
99	Record 99

field→record →file →directory

File Organization…

- **Indexed Sequential File Organization**
  - Indexed sequential file organization stores data for fast retrieval
  - We maintain a table known as the **index table** which stores the record number and the address of all the records

Physically the records may be stored anywhere, but the index table stores the address of those records



Record number	Address of the Record
1	765
2	27
3	876
4	742
5	NULL
6	NULL
7	NULL
8	NULL
9	NULL

Indexing.

- An index for a file can be compared with a catalogue in a library
  - Like a library has card catalogues based on authors, subjects, or titles, a file can also have one or more indices
  - A file may have multiple indices based on different fields

- There are several indexing techniques and each technique works well for a particular application

Indexing..

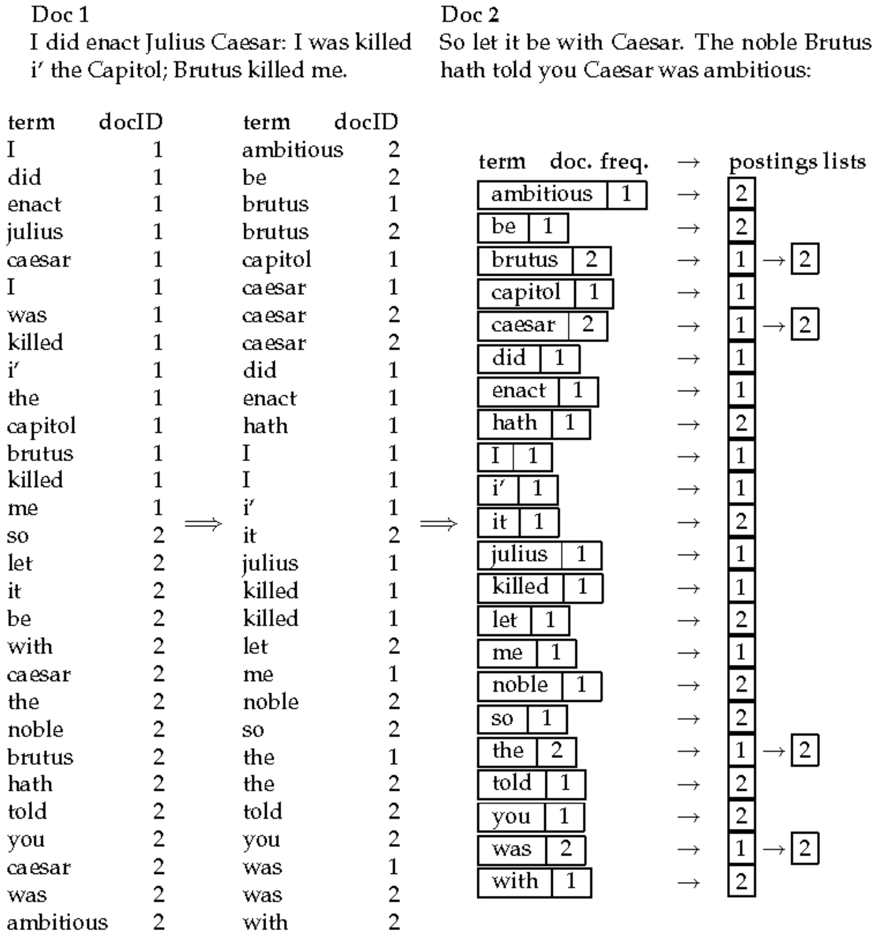
- Primary Indexing
  - The index whose search key specifies the sequential order of the file is defined as the primary index
    - For example, suppose records of students are stored in a STUDENT file in a sequential order starting from roll number 1 to roll number 60

- Secondary Indexing
  - An index whose search key specifies an order different from the sequential order of the file is called as the secondary index
    - For example, if the record of a student is searched by his name, then the name is a secondary index

field⟶record ⟶file ⟶directory

Indexing..

- Inverted Indexing
  - Inverted files are commonly used in document retrieval systems for large textual databases



Doc 1
I did enact Julius Caesar: I was killed
i′ the Capitol; Brutus killed me.

Doc 2
So let it be with Caesar. The noble Brutus
hath told you Caesar was ambitious:

term	docID
I	1
did	1
enact	1
julius	1
caesar	1
I	1
was	1
killed	1
i′	1
the	1
capitol	1
brutus	1
killed	1
me	1
so	2
let	2
it	2
be	2
with	2
caesar	2
the	2
noble	2
brutus	2
hath	2
told	2
you	2
caesar	2
was	2
ambitious	2

⟹

term	docID
ambitious	2
be	2
brutus	1
brutus	2
capitol	1
caesar	1
caesar	2
caesar	2
did	1
enact	1
hath	1
I	1
I	1
i′	1
it	2
julius	1
killed	1
killed	1
let	2
me	1
noble	2
so	2
the	1
the	2
told	2
you	2
was	1
was	2
with	2

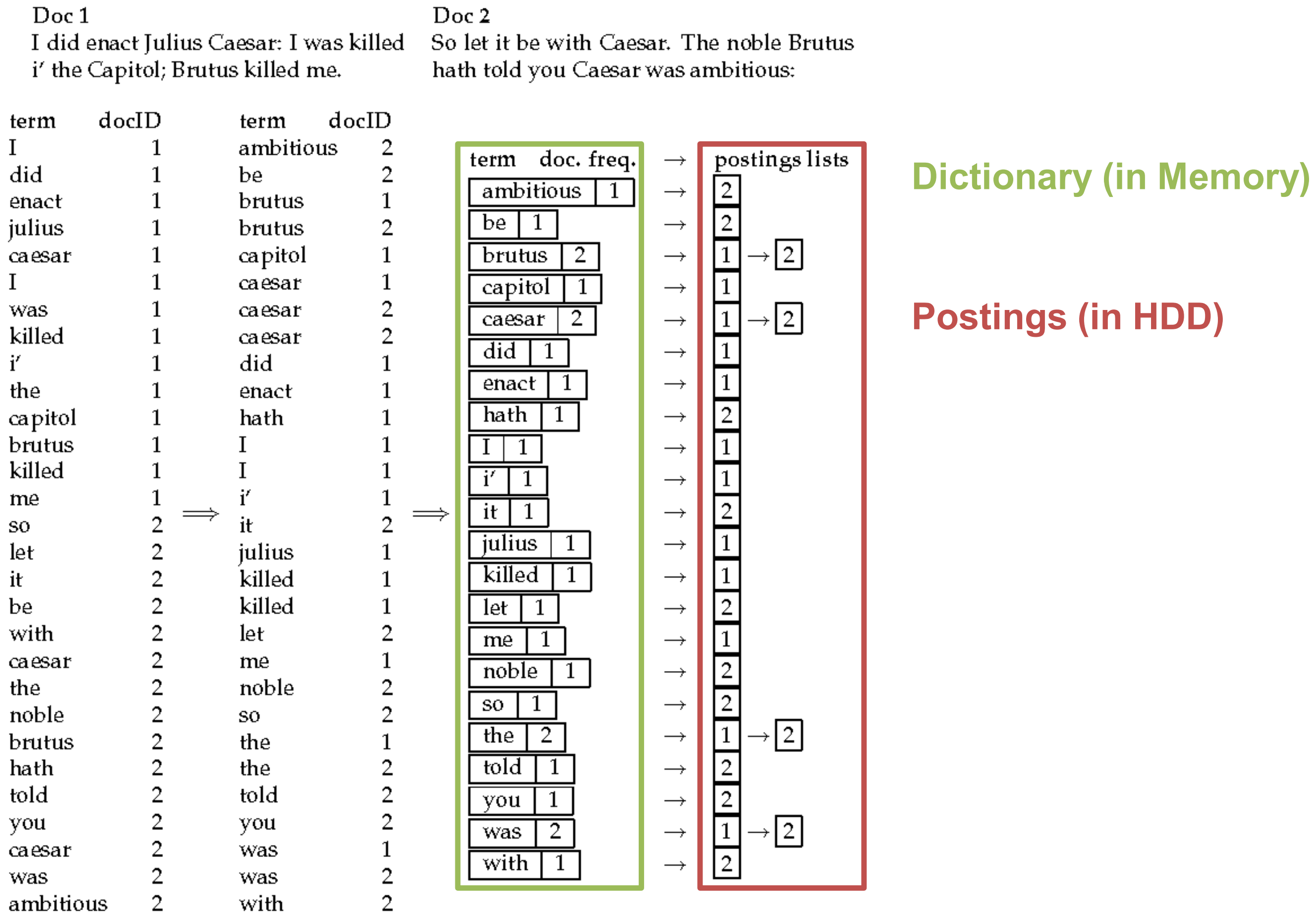
⟹

term	doc. freq.	→	postings lists
ambitious	1	→	2
be	1	→	2
brutus	2	→	1 → 2
capitol	1	→	1
caesar	2	→	1 → 2
did	1	→	1
enact	1	→	1
hath	1	→	2
I	1	→	1
i′	1	→	1
it	1	→	2
julius	1	→	1
killed	1	→	1
let	1	→	2
me	1	→	1
noble	1	→	2
so	1	→	2
the	2	→	1 → 2
told	1	→	2
you	1	→	2
was	2	→	1 → 2
with	1	→	2

field→record →file →directory

Indexing..

- Inverted Indexing
  - Inverted files are commonly used in document retrieval systems for large textual databases



Doc 1
I did enact Julius Caesar: I was killed
i′ the Capitol; Brutus killed me.

Doc 2
So let it be with Caesar. The noble Brutus
hath told you Caesar was ambitious:

term	docID
I	1
did	1
enact	1
julius	1
caesar	1
I	1
was	1
killed	1
i′	1
the	1
capitol	1
brutus	1
killed	1
me	1
so	2
let	2
it	2
be	2
with	2
caesar	2
the	2
noble	2
brutus	2
hath	2
told	2
you	2
caesar	2
was	2
ambitious	2

⟹

term	docID
ambitious	2
be	2
brutus	1
brutus	2
capitol	1
caesar	1
caesar	2
caesar	2
did	1
enact	1
hath	1
I	1
I	1
i′	1
it	2
julius	1
killed	1
killed	1
let	2
me	1
noble	2
so	2
the	1
the	2
told	2
you	2
was	1
was	2
with	2

⟹

term	doc. freq.	→	postings lists
ambitious	1	→	2
be	1	→	2
brutus	2	→	1 → 2
capitol	1	→	1
caesar	2	→	1 → 2
did	1	→	1
enact	1	→	1
hath	1	→	2
I	1	→	1
i′	1	→	1
it	1	→	2
julius	1	→	1
killed	1	→	1
let	1	→	2
me	1	→	1
noble	1	→	2
so	1	→	2
the	2	→	1 → 2
told	1	→	2
you	1	→	2
was	2	→	1 → 2
with	1	→	2

Dictionary (in Memory)

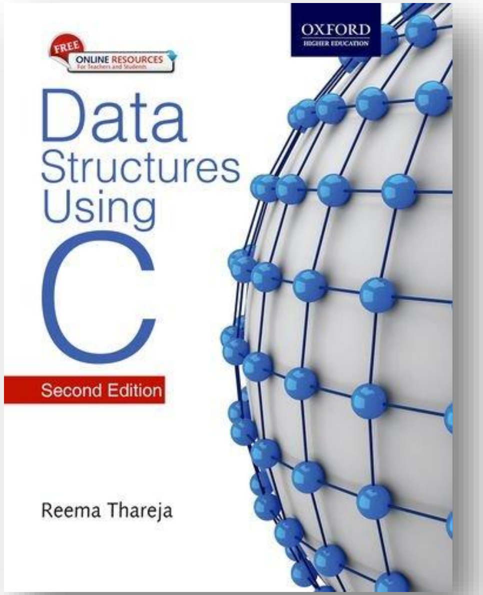
Postings (in HDD)

Summary

9/10	Course Overview
9/12	Algorithms & Recursions
9/17	Performance Analysis
9/19	Arrays
9/24	Moon Festival
9/26	Stacks
10/1	Stacks & Queues
10/3	Queues
10/8	Trees
10/10	National Day
10/15	Binary Search Trees
10/17	AVL Trees
10/22	Red-Black, Splay and Hoffman Trees
10/24	Multi-way Search Trees

10/29	B+ & 2-3 Trees
10/31	Searching
11/5	Midterm
11/7	Bubble, Insertion, Selection & Tree Sorts
11/12	Merge, Quick & Radix Sorts
11/14	Discussion on Midterm
11/19	Heap and Shell Sorts & Comparisons
11/21	Undirected & Directed Graphs
12/3	Advanced Graphs
12/5	Shortest Path Algorithms
12/10	Heaps
12/12	Hash Functions

12/17	Collision
12/19	
12/22	Make-up Class for 12/31? Break!
12/24	Final Exam
12/26	Break
12/31	Bridge Holiday
1/2	Discussion on Final Exam



Questions?

kychen@mail.ntust.edu.tw