Linked Lists (cont.)

Instructor: Dr. Sunho Lim (Ph.D., Assistant Professor)

Lecture 04

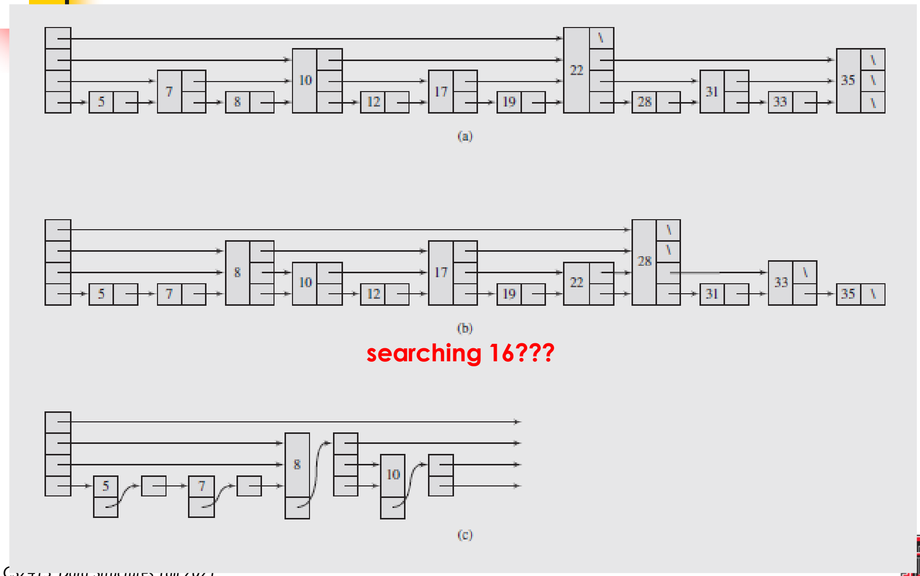
sunho.lim@ttu.edu

Adapted partially from Data Structures and Algorithms in C++, Adam Drozdek, 4th Edition, Cengage Learning; and Algorithms and Data Structures, Douglas Wilhelm Harder, Mmath

Skip Lists

- Drawback to the linked lists
  - sequential in nature, e.g., searching for a particular element
  - ordering the elements on **frequency of access** can help
  - still sequential access
- a variation called a ***skip list***
  - **non-sequential searching** of a linked list

Skip Lists (cont.)



(a)

(b)

searching 16???

(c)

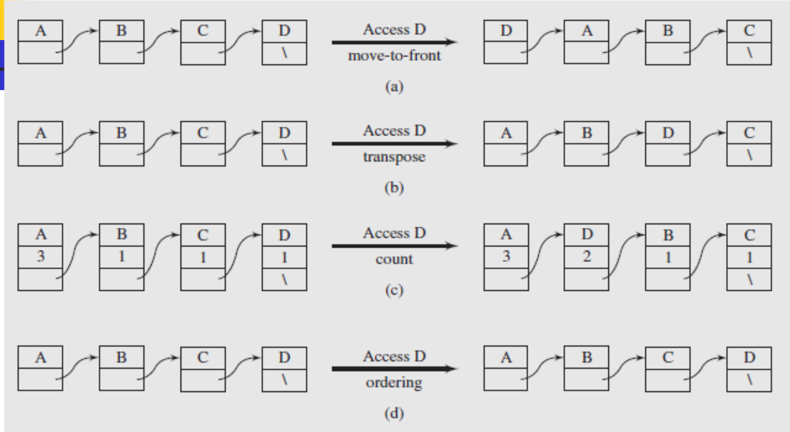
Skip Lists (cont.)

- Efficient searching but,
  - at the expense of insertion and deletion operations
  - inserting a new node,
    - require restructuring of the nodes that follow the new node
  - must change the number of pointers and their values

Self-Organizing Lists

- Skip lists,
  - speed up the searching process in lists
- Dynamically **re-organizing** the lists as they are used??
- Several ways to accomplish this
  - ***Move-to-front***: when found, the target is moved to the front of the list
  - ***Transpose***: when the element is found, it is swapped with its predecessor in the list
  - ***Count***: the list is ordered by frequency of access
  - ***Ordering***: the list is ordered based on the natural nature of the data

Self-Organizing Lists (cont.)



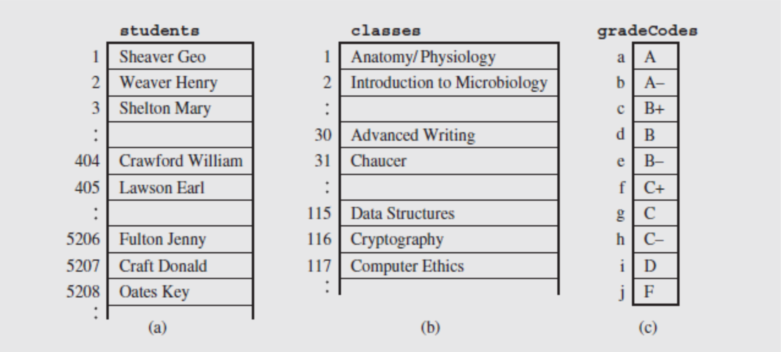
- For smaller lists, simple linking suffices
- As the amount of data and frequency of access increases,
  - other structures should be employed

Sparse Tables

- Tables..
  - a data structure of choice in many applications due to their ease of implementation, use, and access
- What if the table is mostly unoccupied, a ***sparse table***
  - using **linked lists**
  - for example, want to store the grades for 8000 students in 300 classes

Sparse Tables (cont.)

- **Student numbers** (rows) and **course numbers** (columns)
  - To save space, grades can be encoded using single character
- Three tables,



students

1	Sheaver Geo
2	Weaver Henry
3	Shelton Mary
:	
404	Crawford William
405	Lawson Earl
:	
5206	Fulton Jenny
5207	Craft Donald
5208	Oates Key
:	

(a)

classes

1	Anatomy/ Physiology
2	Introduction to Microbiology
:	
30	Advanced Writing
31	Chaucer
:	
115	Data Structures
116	Cryptography
117	Computer Ethics
:	

(b)

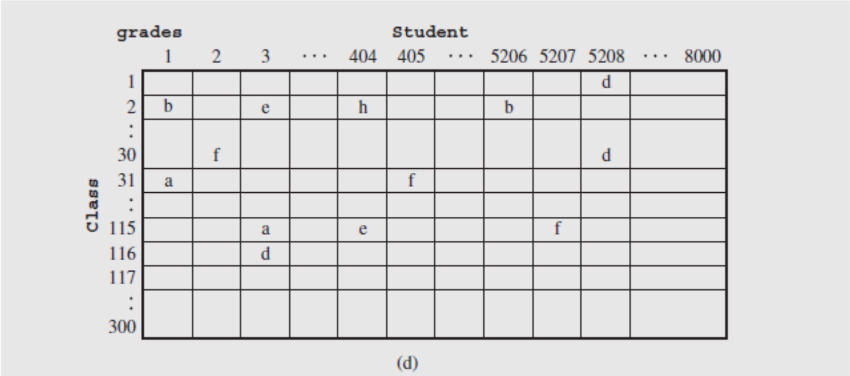
gradeCodes

a	A
b	A–
c	B+
d	B
e	B–
f	C+
g	C
h	C–
i	D
j	F

(c)

Sparse Tables (cont.)

- **Two-dimensional array**
  - many open spaces



grades — Student — Class

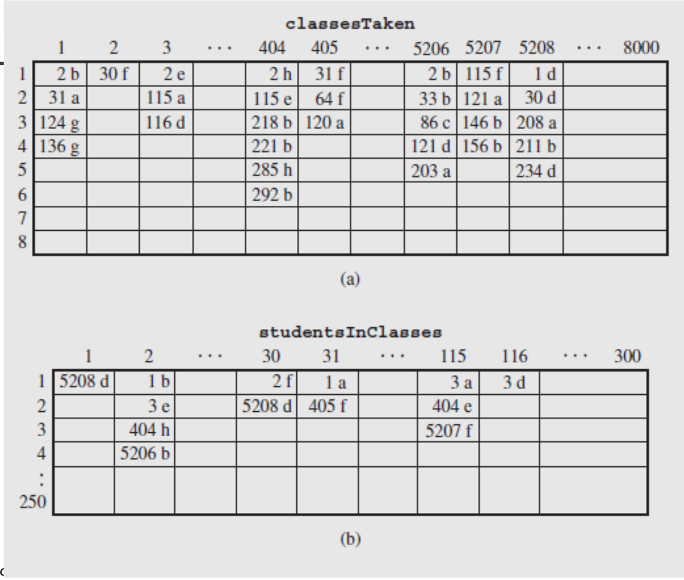
Class	1	2	3	···	404	405	···	5206	5207	5208	··· 8000
1										d	
2	b		e		h			b			
⋮ 30		f								d	
31	a					f					
⋮											
115			a		e				f		
116			d								
117											
⋮ 300											

(d)

Sparse Tables (cont.)

- The table itself,
  - 8000 (students) by 300 (classes), with one byte per grade, totaling 2.4 million bytes
  - e.g., if every student takes only four classes each semester -- only four entries in each column, **wasting almost 99% of the total space** of the table
- An alternative approach, **two 2-dimensional arrays**
  - classesTaken,
    - record each class a student takes (to a maximum of eight), along with the student's grade
  - studentsInClasses,
    - record the students in each class (to a maximum of 250), along with their grade

Sparse Tables (cont.)



classesTaken

	1	2	3	···	404	405	···	5206	5207	5208	··· 8000
1	2 b	30 f	2 e		2 h	31 f		2 b	115 f	1 d	
2	31 a		115 a		115 e	64 f		33 b	121 a	30 d	
3	124 g		116 d		218 b	120 a		86 c	146 b	208 a	
4	136 g				221 b			121 d	156 b	211 b	
5					285 h			203 a		234 d	
6					292 b						
7											
8											

(a)

studentsInClasses

	1	2	···	30	31	···	115	116	··· 300
1	5208 d	1 b		2 f	1 a		3 a	3 d	
2		3 e		5208 d	405 f		404 e		
3		404 h					5207 f		
4		5206 b							
: 250									

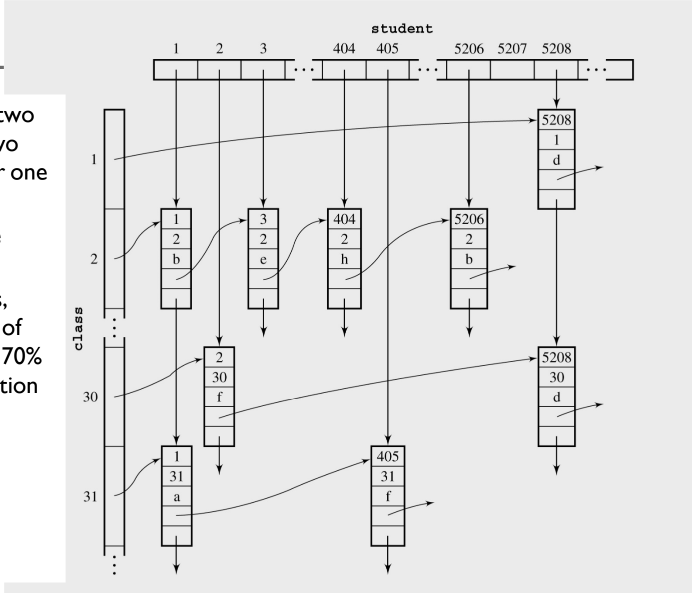
(b)

Sparse Tables (cont.)

- An alternative approach, **two 2-dimensional arrays** (cont.)
  - under assumption, an integer requires 2 bytes of storage, 417,000 bytes would be needed
  - considerably less than the original single table, but still wasteful and **inflexible** if conditions change
- Utilize **two arrays of linked lists**,
  - each node:
    - the student number
    - class number
    - grade,
    - a pointer to the next class for the student
    - a pointer to the next student for the class

Sparse Tables (cont.)

- If a pointer occupies two bytes, each integer two bytes, and a character one byte, then
  - each node is nine bytes in size
- occupy 288,000 bytes, which is roughly 10% of the original table and 70% of the two table variation
- no wasted space & extensible lists


student
1
2
3
404
405
5206
5207
5208
class
1
2
30
31
5208
1
d
1
2
b
3
2
e
404
2
h
5206
2
b
2
30
f
5208
30
d
1
31
a
405
31
f