Data Management and File Organization

Sequential Files

Part two: Sorted Sequential Files

Basic Definitions

- Sorted Sequential Files: A sequential file which is sorted according to the value of an attribute. (This attribute is also called ***key***)
- Ex. The student file is sorted by "student ID" attribute.
- The main operation in a sorted sequential file is exhaustive read (same as pile files)

Why Sorting?

Remembering from pile files, exhaustive reading of a sequential file in order of an attribute needs

$n*T_F$ Seconds. (Several days in hospital file example)

If we sort the file, the exhaustive read will need only a few seconds (14 seconds in hospital file example)

In a sorted file (or list) search is much faster.

Advantage of Sorted Sequential Files

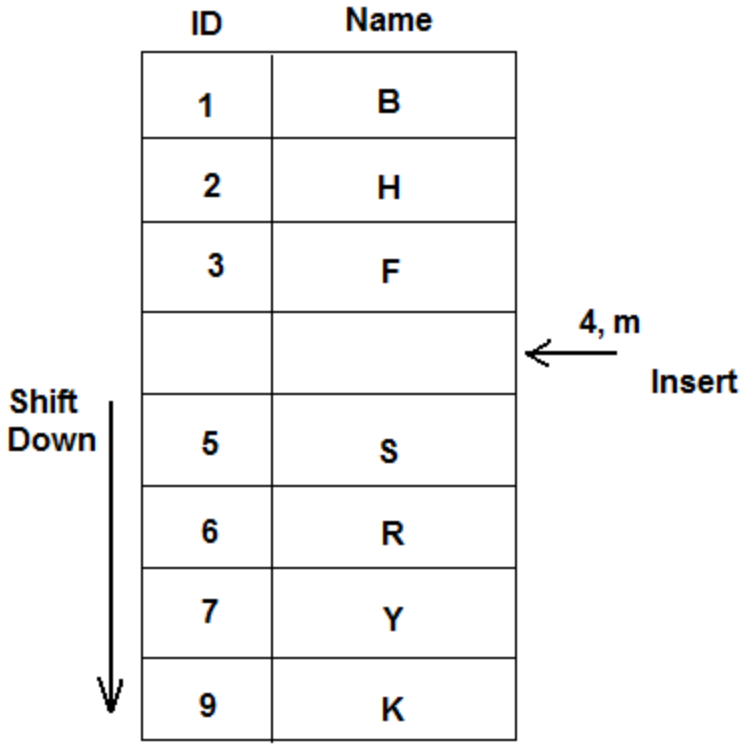
- Instead of exhaustive (sequential) search, much faster search algorithms such as:
  - Binary search
  - Interpolation search

  Can be used

Disadvantage of Sorting (1)

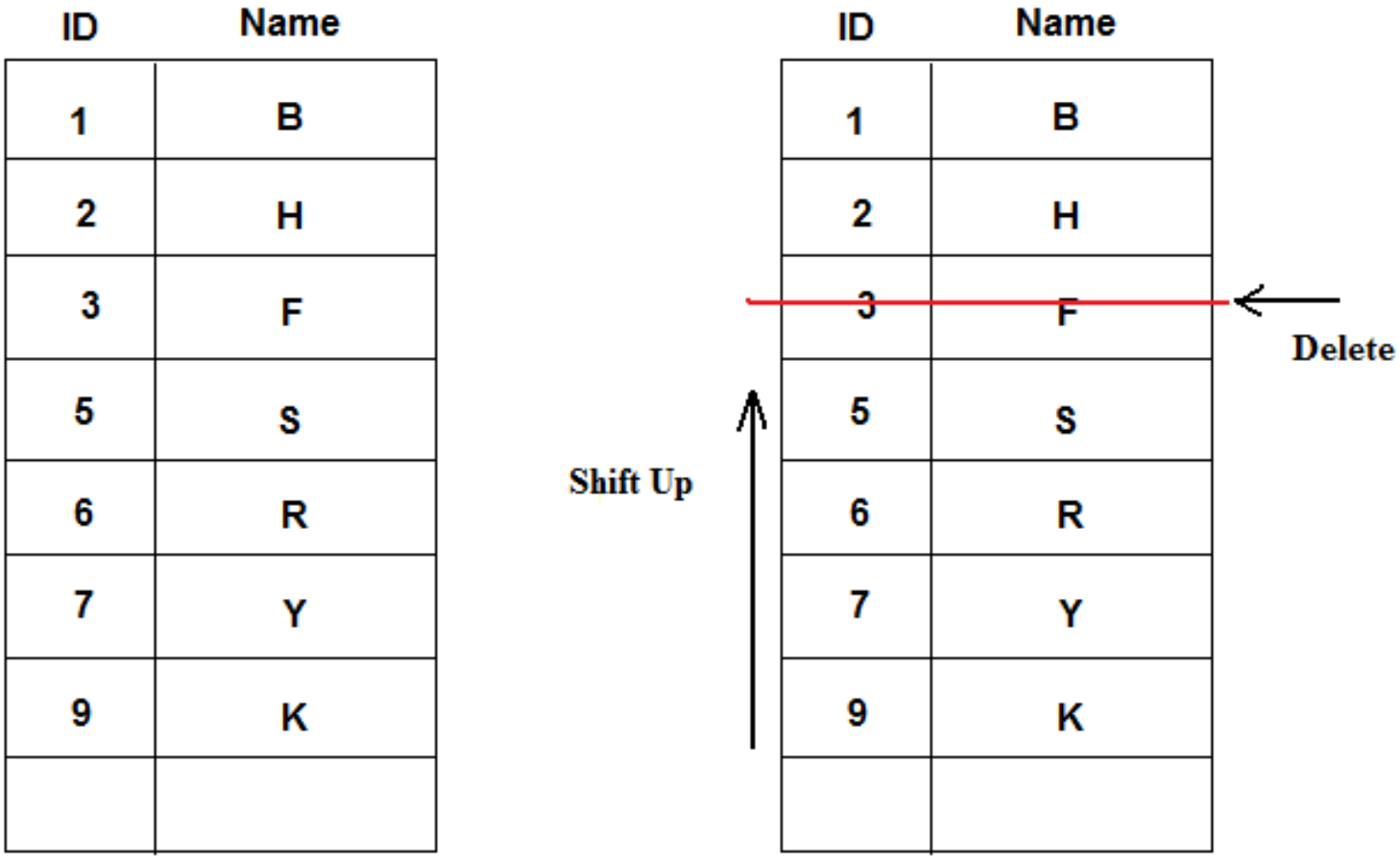
- New insertions are expensive

ID	Name
1	B
2	H
3	F
5	S
6	R
7	Y
9	K



Disadvantage of Sorting (2)

- Delete operation is expensive



Topics for today

- Search algorithms
  - Sequential search
  - Binary search
  - Interpolation Search
- Timings in sorted sequential files

Search Algorithms

- Sequential Search
  - Start from the first record
  - Read until either the record is found or end of the file is reached
  - On average half of the records are read

Sequential Search Example

ID	Name
1	B
2	H
3	F
5	S
6	R
7	Y
9	K
11	N
13	M
14	H
18	G
19	B
22	T

Query 1: Find record with ID=3
3 comparisons

Query 2: Find record with ID=18
11 comparisons

Query 3: Find record with ID=9
8 comparisons

On average for each query N/2 comparisons are needed (N is the number of records)

Search Algorithms

- Binary search
  - Define two boundaries (up and Down) for the search area
  - Find the middle of the search area: mid=(Up+Down)/2
  - If the record at position "mid" has an attribute greater than key then search in the first half

    (Down = mid)
  - Else search the second half (Up=mid)
  - Repeat until the record is found or Up==Down
  - With $Log_2 n$ read, the record is found (worst case)

Binary Search Example

	ID	Name
1	1	B
2	2	H
3	3	F
4	5	S
5	6	R
6	7	Y
7	9	K
8	11	N
9	13	M
10	14	H
11	18	G
12	19	B
13	22	T

Find record with ID = 5

Up = 1, Down = 13, Mid=(1+13)/2 = 7

Record[7].ID = 9 > 5 Then
Down = Mid = 7
Mid = (1+7)/2 =4

Record[4].ID = 5 (found)

Interpolation Search

- If the key value is closer to the record value at Up(or Down) then mid is selected closer to Up(or Down)

$$mid = \frac{Key - Record[Up]}{Record[Down] - Record[Up]} \times (Down - Up) + Up$$

Record[Up] means the attribute value at position Up, Up and Down are index or position at the data file

Example

	ID	Name
1	1	B
2	2	H
3	3	F
4	5	S
5	6	R
6	7	Y
7	9	K
8	11	N
9	13	M
10	14	H
11	18	G
12	19	B
13	22	T

Find record with ID = 2

Interpolation Search

Up = 1, Down = 13
Record[Up].ID = 1
Record[Down].ID=22

Mid = (2-1)/(22-1)*(13-1)+1
Mid = 1.57 (rounded to 2)

Record found

Binary Search

Up=1, Down = 13
Mid = 7

Record[Mid] =9> 2 Then
  Down = Mid = 7
  Mid = (1+7)/2 = 4

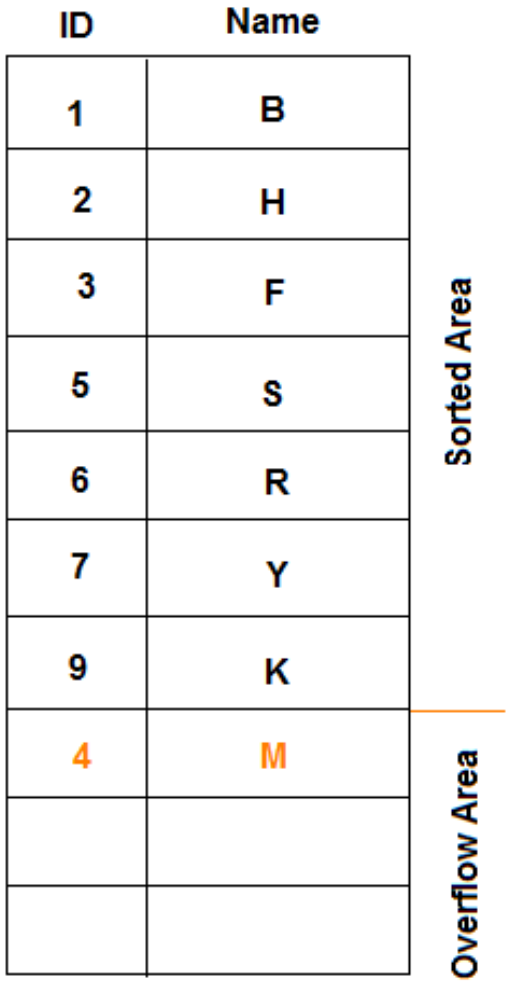
Record[Mid] = 5> 2 Then
  Down = Mid = 5
  Mid = (1+5)/2 = 3
Record[Mid] = 3>2 Then
  Down = Mid = 3
  Mid = (3+1)/2 = 2

Record[Mid] = 2
Found

Sorted sequential file operations

Insertion in Sorted Sequential Files

- An overflow area is defined at the end of the file with unsorted records.



Sequential File Operations and Timings

- Fetch one record T_F
- Fetch next record T_N
- Insert a record T_I
- Update a record T_U
- Delete a record T_D
- Exhaustive reading of the file T_X
- Re-Organize a file T_Y

Fetch One Record

- Find and read a record given an attribute value. Ex. File student record with Student ID=200612345
- In a sorted sequential file, using binary search $Log_2 n$ blocks are read
- If overflow area is empty then

  $T_F = (s + r + btt) * Log_2 n$

  n: number of records in the file

Example

- Find T_F given:
  - Total number of records (n) = 100,000
  - btt = 0.8 msec
  - s=16msec
  - r=8.3msec

Fetch One Record

- If y blocks are in sorted area and x blocks in overflow area then

$(y/b) * (Log_2 y * (s+r+btt)) + (x/b) * (x/2*ebt+s+r)$

y/b : probability of having the record in sorted area

x/b: probability of having the record in overflow area

Example

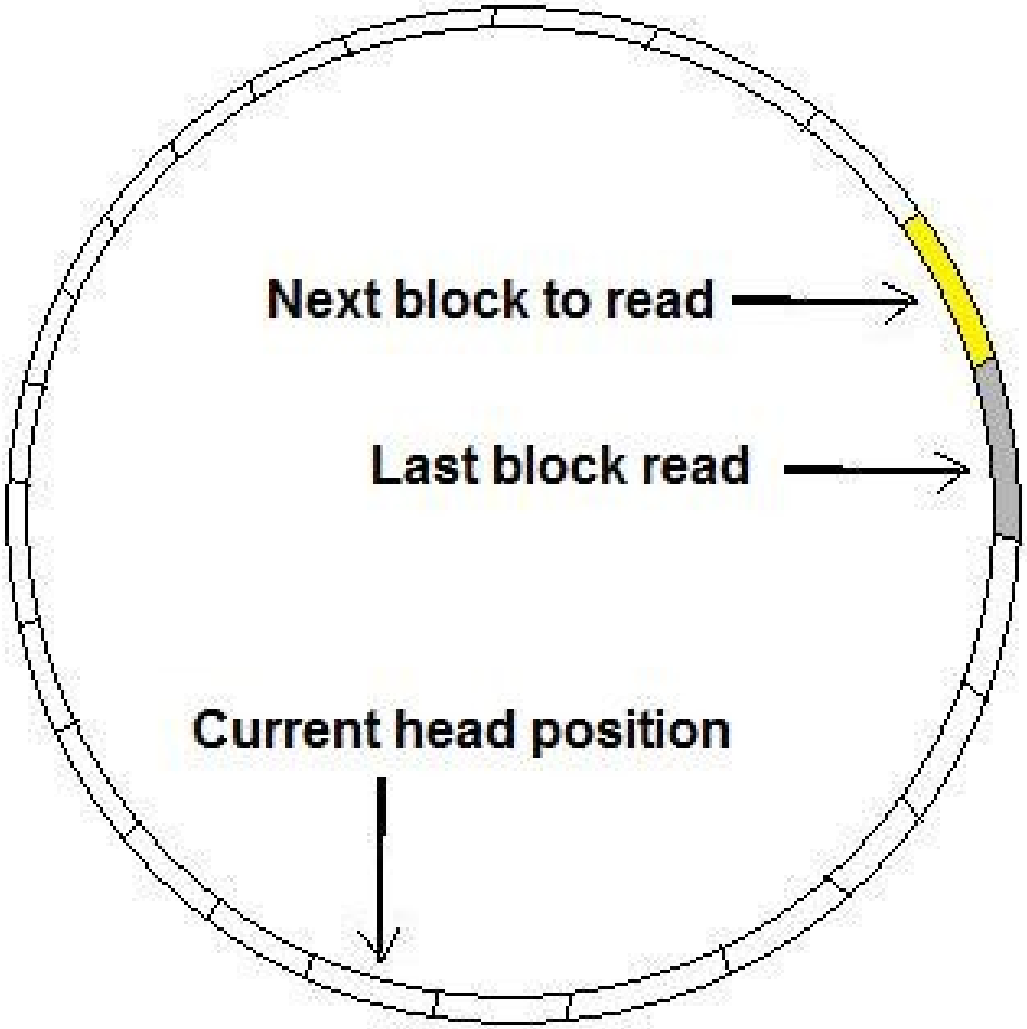
- Find T_F given:
  - Total number of blocks (b) = 16,667
  - 15,000 blocks in sorted area
  - 1,667 blocks in overflow area
  - btt = 0.8 msec
  - s=16msec
  - r=8.3msec
  - ebt=0.84

Fetch Next Record

- Find and read the next record in order of an attribute value.

- If the file is sorted, only in 1/Bfr cases we need to read a new block.
- As we are on the same track, s is not considered
- $T_N = (1/Bfr) * (r+btt)$

Fetch Next Record



Insert a Record

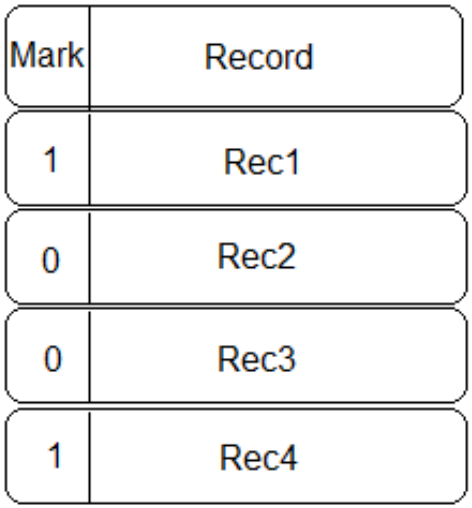
- Insert is always done at the overflow area
  - Read the last block of the overflow area (s+r+btt)
  - Add the new record and write back the block (2r)
  - $T_I = s+r+btt+2r$

Update a Record

- To update, first the block is read, then the record is updated and the block is written back
- Time to read the block $= T_F$
- Time to write back the block $= 2r$
- $T_U = T_F + 2r$

Delete a Record

- To delete a record, we mark it as deleted
- First read the block T_F
- Update the mark and write the block (2r)
- $T_D = T_F + 2r$



Exhaustive Reading of a File

- Case 1: If the overflow area is empty

T_X (No Overflow) = b*ebt + s + r

Ex:

b = 16667

btt = 0.84

s=16msec

r=8.3 msec

Exhaustive Reading of a File

- Case 2: There are some records in overflow area
  - Read overflow area into memory: (x * ebt + s+r)
  - Sort the records in the memory (time is ignored)
  - Read the sorted area and merge with the records in memory (y*ebt + s+r)

Merging two sorted lists

- Algorithm
  - Compare the top records of the lists and get the smaller one
  - Repeat until the end of the lists are reached

Example

- Merging two lists

ID	Name
1	B
3	H
6	F
9	S

ID	Name
2	R
5	Y
7	K
11	W

Re-Organizing a File

- In re-organization, the file is sorted again and the deleted records are removed.
- Sort algorithms are discussed next week

Question?