

CSE 444: Database Internals

Section 3:

Indexing and Operator Algorithms

Problem 1:

B+ tree insertion and deletion

- On board
- (We will do it after problem 2)

Notations

- $B(R)$
- $T(R)$
- $V(R, a)$
- M

Problem 2

Algorithms for Group By and Aggregate Operators

For homework 2:

Understand what is going on, do not blindly apply formula!
Try to choose outer relation carefully to reduce cost/fit data in memory

- Modified Tweet Example:

Tweet(tid, uid, tlen) tlen = tweet length

```
SELECT uid, MIN(tlen)
FROM Tweet
GROUP BY uid
```

Problem 2a: One pass, hash-based grouping

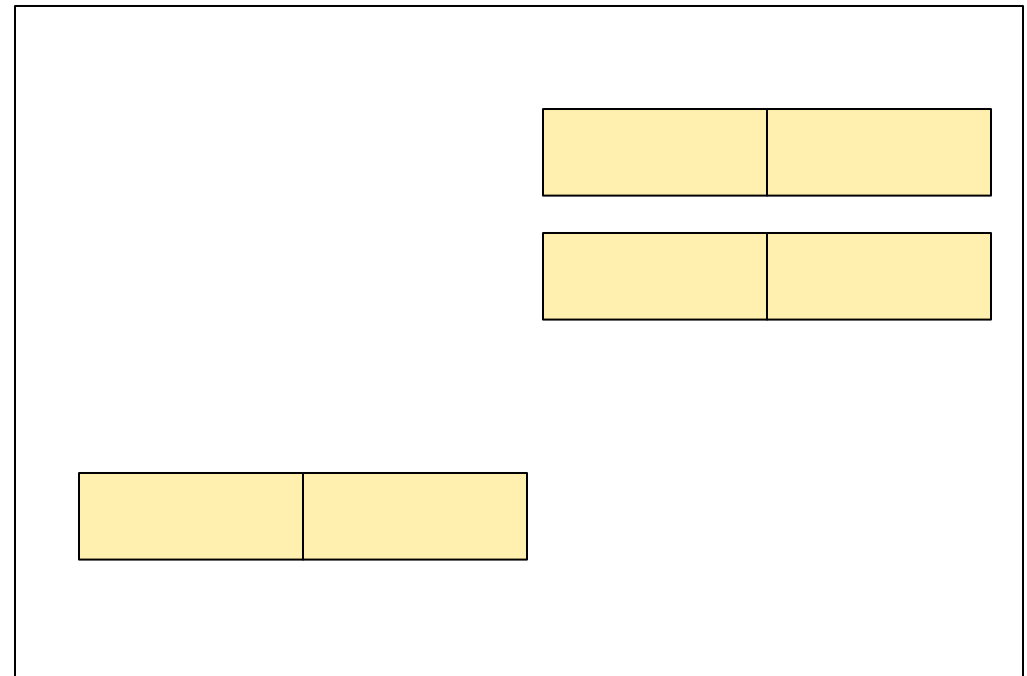
$M = 3$

Showing
tid, uid, tlen

Disk

Tweet

5, 1, 7	4, 2, 10
1, 3, 3	3, 1, 5
7, 3, 8	2, 2, 5
6, 3, 9	8, 1, 10



Problem 2a:

One pass, hash-based grouping

$M = 3$

Main memory data structure
(holds minimum for every
group)

Showing
tid, uid, tlen

Disk

Tweet

5, 1, 7	4, 2, 10
1, 3, 3	3, 1, 5
7, 3, 8	2, 2, 5
6, 3, 9	8, 1, 10

$H = \text{uid} \% 2$

1, 7

2, 10

5, 1, 7 4, 2, 10

Could use other main-memory data
structures as well

One pass, hash-based grouping

M = 3

Showing
tid, uid, tlen

Disk

Tweet

5, 1, 7	4, 2, 10
1, 3, 3	3, 1, 5
7, 3, 8	2, 2, 5
6, 3, 9	8, 1, 10

$H = \text{uid} \% 2$

1, 3, 3	3, 1, 5
---------	---------

1, 5	3, 3
2, 10	

Minimum
updated
from 7 to 5

Discussion: Problem 2a

Cost:

- Clustered?
- B(R): assuming $M - 1$ pages can hold all groups – tuples for groups can be shorter or larger than original tuples
- Unclustered?
- Also B(R)

Which method does the grouping:

open(), next(), or close()?

- Cannot return anything until the entire data is read. Open() needs to do grouping

What to do for AVG(tlen)?

- Keep both SUM(tlen) and COUNT(*) for each group in memory

One pass, hash-based grouping

Showing
tid, uid, tlen

Main memory data structure
(holds minimum for every
group)

$M = 3$

Tweet

5, 1, 7	4, 2, 10
1, 3, 3	3, 5, 5
7, 3, 8	2, 2, 5
6, 3, 9	8, 1, 10

$H = \text{uid} \% 2$

5, 1, 7	4, 2, 10
---------	----------

1, 7	
2, 10	

Data has been changed

One pass, hash-based grouping

Showing uid, tlen

Main memory data structure (holds minimum for every group)

$M = 3$

Tweet

5, 1, 7	4, 2, 10
1, 3, 3	3, 5, 5
7, 3, 8	2, 2, 5
6, 3, 9	8, 1, 10

$H = \text{uid} \% 2$

1, 3, 3	3, 5, 5
---------	---------

1, 7	3, 3
------	------

2, 10	
-------	--

Three keys map to the same bucket, need to flush

- Solution?

Two pass Hash-based Aggregate algorithms

- First Hash all tuples, then perform the aggregate in second pass

Problem 2b:

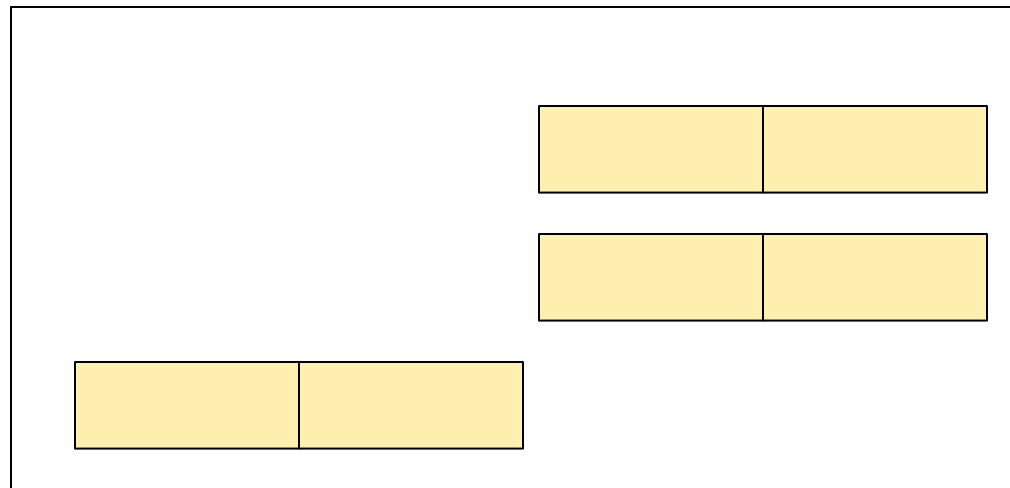
Two pass, hash-based grouping

$M = 3$

Showing
Tid, uid, tlen

Tweet

5, 1, 7	4, 2, 10
1, 3, 3	3, 5, 5
7, 3, 1	2, 2, 5
6, 4, 9	8, 4, 10



Hint: Two-pass hash-based join in yesterday's lecture!

Problem 2b:

Two pass, hash-based grouping

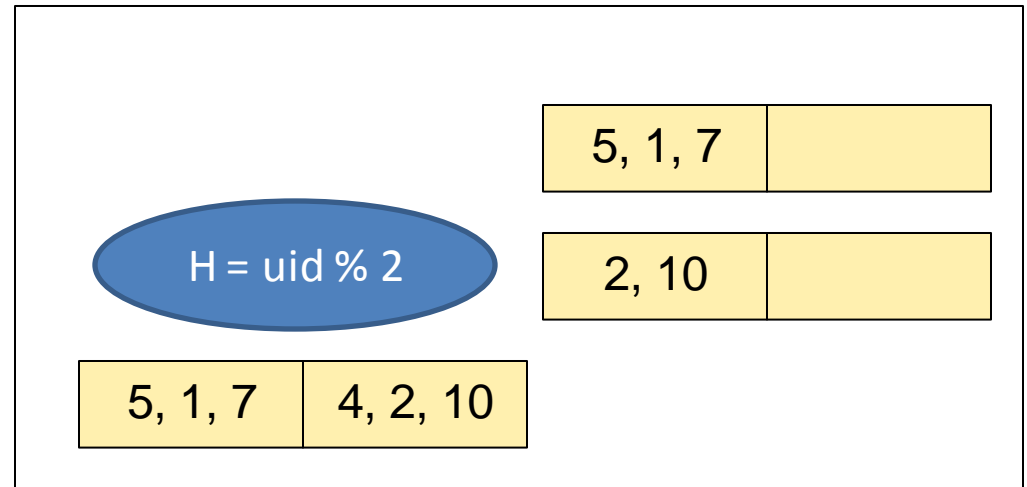
No Aggregation is performed in the first pass

$M = 3$

Showing
Tid, uid, tlen

Tweet

5, 1, 7	4, 2, 10
1, 3, 3	3, 5, 5
7, 3, 1	2, 2, 5
6, 4, 9	8, 4, 10



Two pass, hash-based grouping

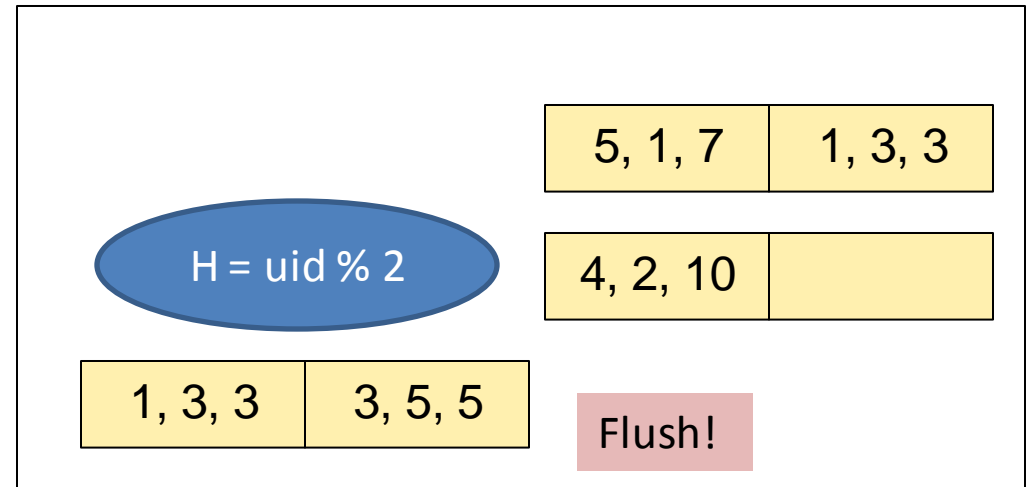
Showing
tid, uid, tlen

No Aggregation is performed in the first pass

$M = 3$

Tweet

5, 1, 7	4, 2, 10
1, 3, 3	3, 5, 5
7, 3, 1	2, 2, 5
6, 4, 9	8, 4, 10



Two pass, hash-based grouping

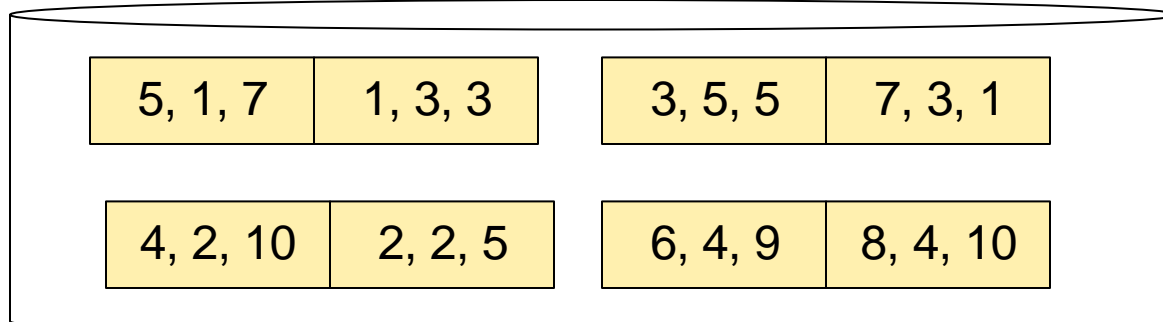
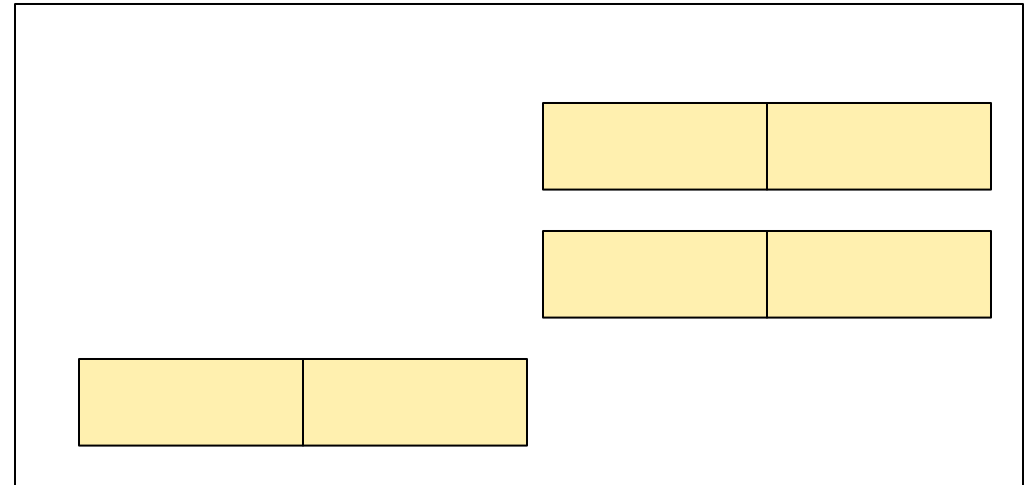
Showing
tid, uid, tlen

Final buffer and disk after pass1

$M = 3$

Tweet

5, 1, 7	4, 2, 10
1, 3, 3	3, 5, 5
7, 3, 1	2, 2, 5
6, 4, 9	8, 4, 10



Two pass, hash-based grouping

Showing
tid, uid, tlen

Second pass: compute aggregate in each bucket
Need to keep only one record per group

$M = 3$

Tweet

5, 1, 7	4, 2, 10
1, 3, 3	3, 5, 5
7, 3, 1	2, 2, 5
6, 4, 9	8, 4, 10

	1, 7	3, 3
5, 1, 7	1, 3, 3	

5, 1, 7	1, 3, 3	3, 5, 5	7, 3, 1
4, 2, 10	2, 2, 5	6, 4, 9	8, 4, 10

Two pass, hash-based grouping

Showing
tid, uid, tlen

Second pass: compute aggregate in each bucket
Need to keep only one record per group

$M = 3$

Tweet

5, 1, 7	4, 2, 10
1, 3, 3	3, 5, 5
7, 3, 1	2, 2, 5
6, 4, 9	8, 4, 10

Update min

1, 7	3, 3
5, 5	

3, 5, 5	7, 3, 1
---------	---------

5, 1, 7	1, 3, 3	3, 5, 5	7, 3, 1
4, 2, 10	2, 2, 5	6, 4, 9	8, 4, 10

Discussion: Problem 2b

Cost?

- $3B(R)$

Assumptions?

- Need to hold all distinct values in the same bucket in $M-1$
- Assuming uniformity, $B(R) \leq M^2$ is safe to assume
- But note that can handle much bigger relations R if the groups are large and #groups is small.

Problem 2c:

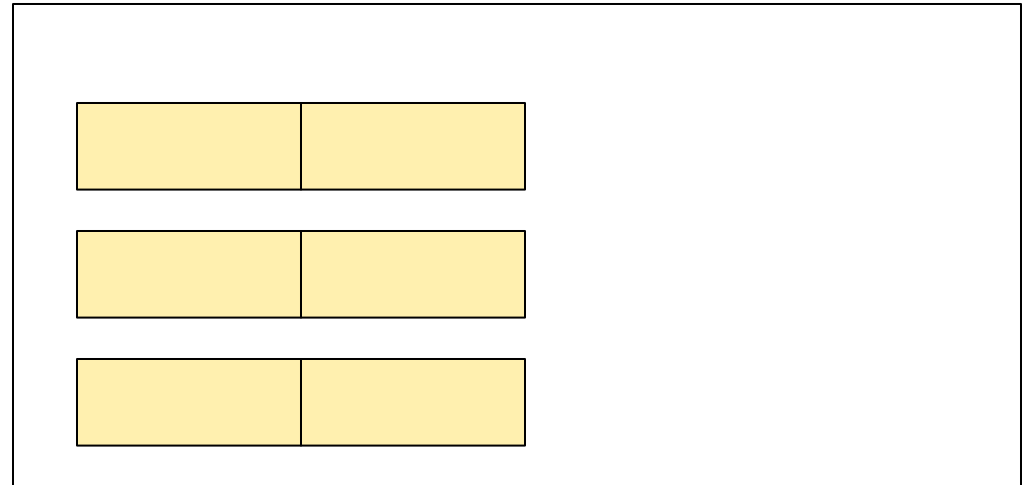
Two pass, sort-merge-based grouping

$M = 3$

Showing
tid, uid, tlen

Tweet

5, 1, 7	4, 2, 10
1, 3, 3	3, 5, 5
7, 3, 1	2, 2, 5
6, 4, 9	8, 4, 10



Hint: Two-pass sort-merged join in yesterday's lecture!

Two pass, sort-based grouping

Showing
Tid, uid, tlen

Step 1: Divide R into M partitions
sort each partition in memory
(on group by attr = uid)
Write to disk

M = 3

Tweet

5, 1, 7	4, 2, 10
1, 3, 3	3, 5, 5
7, 3, 1	2, 2, 5
6, 4, 9	8, 4, 10

5, 1, 7	4, 2, 10
2, 2, 5	1, 3, 3
7, 3, 1	3, 5, 5

5, 1, 7	4, 2, 10	2, 2, 5	1, 3, 3	7, 3, 1	3, 5, 5
---------	----------	---------	---------	---------	---------

Two pass, sort-merged-based grouping

Showing uid, tlen

Step 1: Divide R into M partitions
sort each partition in memory
(on group by attr = uid)
Write to disk

M = 3

Tweet

5, 1, 7	4, 2, 10
1, 3, 3	3, 5, 5
7, 3, 1	2, 2, 5
6, 4, 9	8, 4, 10

6, 4, 9	8, 4, 10

5, 1, 7	4, 2, 10	2, 2, 5	1, 3, 3	7, 3, 1	3, 5, 5
6, 4, 9	8, 4, 10				

Two pass, sort-merged-based grouping

Showing uid, tlen

Step 2:

- Load first blocks from all runs
- Find minimum of each key by “Combine” approach in merge-sort
- Repeatedly find the least value of the sort key: next group

Tweet

5, 1, 7	4, 2, 10
1, 3, 3	3, 5, 5
7, 3, 1	2, 2, 5
6, 4, 9	8, 4, 10

5, 1, 7	4, 2, 10
6, 4, 9	8, 4, 10

(uid, min(tlen))
(1, 7)

Not showing the outputs in output buffer

5, 1, 7	4, 2, 10	2, 2, 5	1, 3, 3	7, 3, 1	3, 5, 5
6, 4, 9	8, 4, 10				

Two pass, sort-merged-based grouping

Showing uid, tlen

Step 2: Find minimum of each key by "Combine" approach in merge-sort

M = 3

Repeatedly find the lest value of the sort key:
next group

Tweet

5, 1, 7	4, 2, 10
1, 3, 3	3, 5, 5
7, 3, 1	2, 2, 5
6, 4, 9	8, 4, 10

5, 1, 7	4, 2, 10
6, 4, 9	8, 4, 10

(uid, min(tlen))
(1, 7)
(2, 10)

Not showing the outputs in output buffer

5, 1, 7	4, 2, 10	2, 2, 5	1, 3, 3	7, 3, 1	3, 5, 5
6, 4, 9	8, 4, 10				

Two pass, sort-merged-based grouping

Step 2: Find minimum of each key by "Combine" approach in merge-sort

Repeatedly find the lest value of the sort key:
next group

M = 3

Showing uid, tlen

Tweet

5, 1, 7	4, 2, 10
1, 3, 3	3, 5, 5
7, 3, 1	2, 2, 5
6, 4, 9	8, 4, 10

5, 1, 7	4, 2, 10
6, 4, 9	8, 4, 10

(uid, min(tlen))
(1, 7)
(2, 10)

Not showing the outputs in output buffer

5, 1, 7	4, 2, 10	2, 2, 5	1, 3, 3	7, 3, 1	3, 5, 5
6, 4, 9	8, 4, 10				

Two pass, sort-merged-based grouping

Step 2: Find minimum of each key by "Combine" approach in merge-sort

Repeatedly find the lest value of the sort key:
next group

M = 3

Showing uid, tlen

Tweet

5, 1, 7	4, 2, 10
1, 3, 3	3, 5, 5
7, 3, 1	2, 2, 5
6, 4, 9	8, 4, 10

2, 2, 5	1, 3, 3	(uid, min(tlen))
6, 4, 9	8, 4, 10	(1, 7)
		(2, 10)

Not showing the outputs in output buffer

5, 1, 7	4, 2, 10	2, 2, 5	1, 3, 3	7, 3, 1	3, 5, 5
6, 4, 9	8, 4, 10				

Two pass, sort-merged-based grouping

Step 2: Find minimum of each key by "Combine" approach in merge-sort

Repeatedly find the lest value of the sort key:
next group

$M = 3$

Showing uid, tlen

Tweet

5, 1, 7	4, 2, 10
1, 3, 3	3, 5, 5
7, 3, 1	2, 2, 5
6, 4, 9	8, 4, 10

2, 2, 5	1, 3, 3
6, 4, 9	8, 4, 10

(uid, min(tlen))
(1, 7)
(2, 10)
(3, 3)

Not showing the outputs in output buffer

5, 1, 7	4, 2, 10	2, 2, 5	1, 3, 3	7, 3, 1	3, 5, 5
6, 4, 9	8, 4, 10				

Two pass, sort-merged-based grouping

Showing uid, tlen

Step 2: Find minimum of each key by "Combine" approach in merge-sort

M = 3

Repeatedly find the lest value of the sort key:
next group

Tweet

5, 1, 7	4, 2, 10
1, 3, 3	3, 5, 5
7, 3, 1	2, 2, 5
6, 4, 9	8, 4, 10

7, 3, 1	3, 5, 5
6, 4, 9	8, 4, 10

(uid, min(tlen))
(1, 7)
(2, 10)
(3, 3)

Not showing the outputs in output buffer

5, 1, 7	4, 2, 10	2, 2, 5	1, 3, 3	7, 3, 1	3, 5, 5
6, 4, 9	8, 4, 10				

Two pass, sort-merged-based grouping

Showing uid, tlen

Step 2: Find minimum of each key by "Combine" approach in merge-sort

M = 3

Repeatedly find the lest value of the sort key:
next group

Tweet

5, 1, 7	4, 2, 10
1, 3, 3	3,5, 5
7, 3, 1	2, 2,5
6, 4, 9	8, 4, 10

7, 3, 1	3, 5, 5
6, 4, 9	8, 4, 10

(uid, min(tlen))
 (1, 7)
 (2, 10)
 (3, 1)
 (4, 9)
 (5, 5)

Not showing the outputs in output buffer

5, 1, 7	4, 2, 10	2, 2,5	1, 3, 3	7, 3, 1	3,5, 5
6, 4, 9	8, 4, 10				

Discussion: Problem 2c

Cost?

- $3B(R)$

Assumptions?

- Need to hold one block from each run in M pages
- $B(R) \leq M^2$

Merge-sort based single pass algorithm?

- Not good here: same IO cost, more CPU cost

One pass vs. Two pass

- One pass:
 - smaller disk I/O cost
 - e.g. $B(R)$ for one-pass hash-based aggregation
 - Handles smaller relations
 - e.g. $B(R) \leq M$
- Two/Multi pass:
 - Larger disk I/O cost
 - e.g. $3B(R)$ for two-pass hash-based aggregation
 - Can handle larger relations
 - e.g. $B(R) \leq M^2$

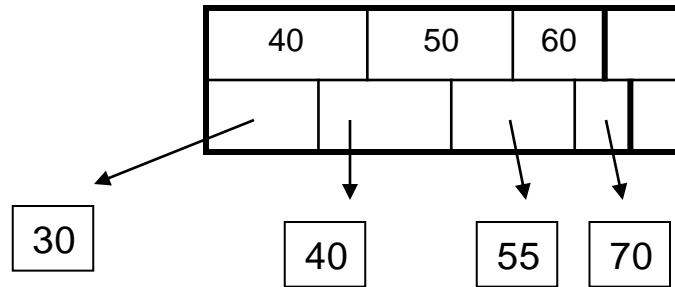
Review

- Two-pass Hash-based Join
 - Cost: $3B(R) + 3B(S)$
 - Assumption: $\text{Min}(B(R), B(S)) \leq M^2$
- Two-pass Sort-merge-based Join
 - Implementation 1:
 - Cost: $5B(R) + 5B(S)$
 - For R, S: sort runs/sublists (2 I/O, read + write)
 - Merge sublists to have entire R, S sorted individually (2 I/O, read + write)
 - Join by combining R and S (only read, write not counted - 1 I/O)
 - Assumption: $B(R) \leq M^2$, $B(S) \leq M^2$
 - Implementation 2:
 - Cost: $3B(R) + 3B(S)$
 - Assumption: $B(R) + B(S) \leq M^2$

Problem 1

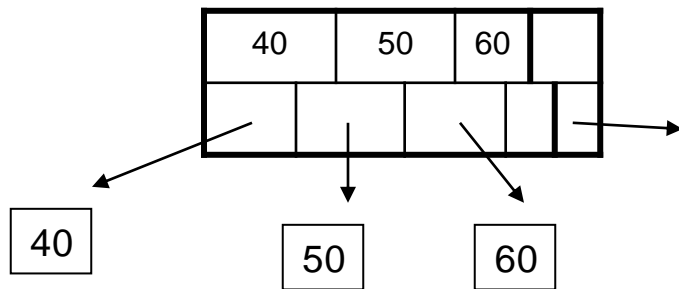
Insertions and Deletion in a B+ tree

- On whiteboard, see the scanned example
- Note: the $<$, \leq assumptions in this class:



Internal node:

- Left pointer from key = k: to keys $< k$
- Right pointer: to keys $\geq k$



Leaf node:

- Left pointer from key = k: to the block containing data with value k in that attribute
- Last remaining pointer on right: To the next leaf on right

Problem 1

Insertions and Deletion in a B+ tree

- Note: when a leaf is split, the middle ($d+1$ -th) key is copied to the new leaf on **right** (and also inserted in parent)
 - Since we assumed the **right pointer** from $\text{key} = k$ points to keys $\geq k$
- Note: when an internal node is split, we do not need to copy the middle ($d+1$ -th) key to the right, only insert it in parent
 - Use the left pointer of the new right internal node.
 - See the scanned example