Discussion 3

File organization & Index Files

& Quiz3

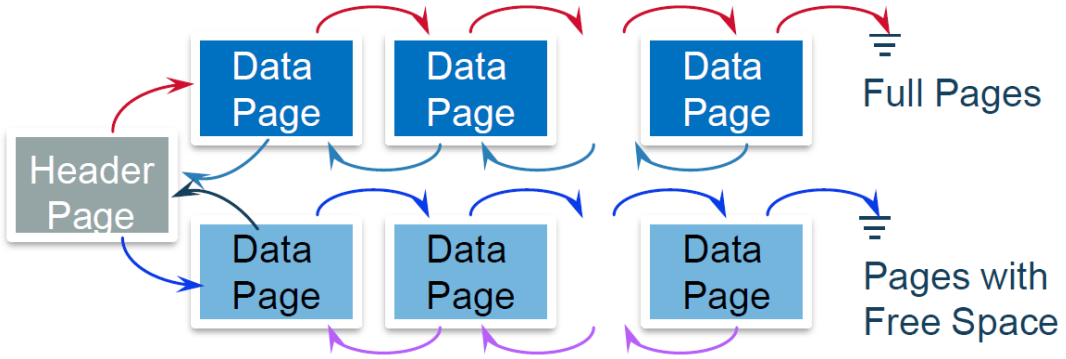
Jiahui Xu

Database File Structures

- Heap Files
  - Implemented as List
  - Better: Use a Page Directory
- Sorted Files
- Index Files
  - ISAM
  - B+ Tree
- Cost of Operations

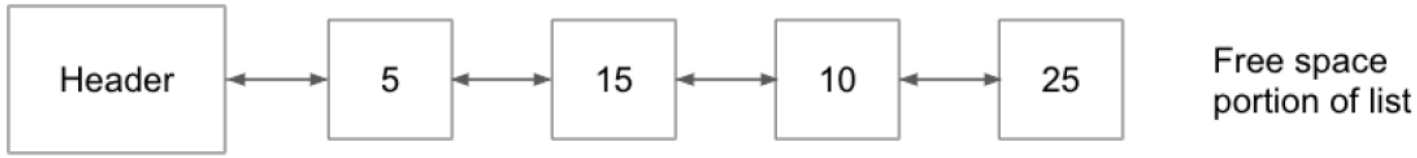
Heap Files - Linked List

- Each data page contains:
  - records, a free space tracker, and pointers (byte offsets) to the next and previous page
- One header page
  - separates the data pages into full pages and free pages



Heap Files - Linked List

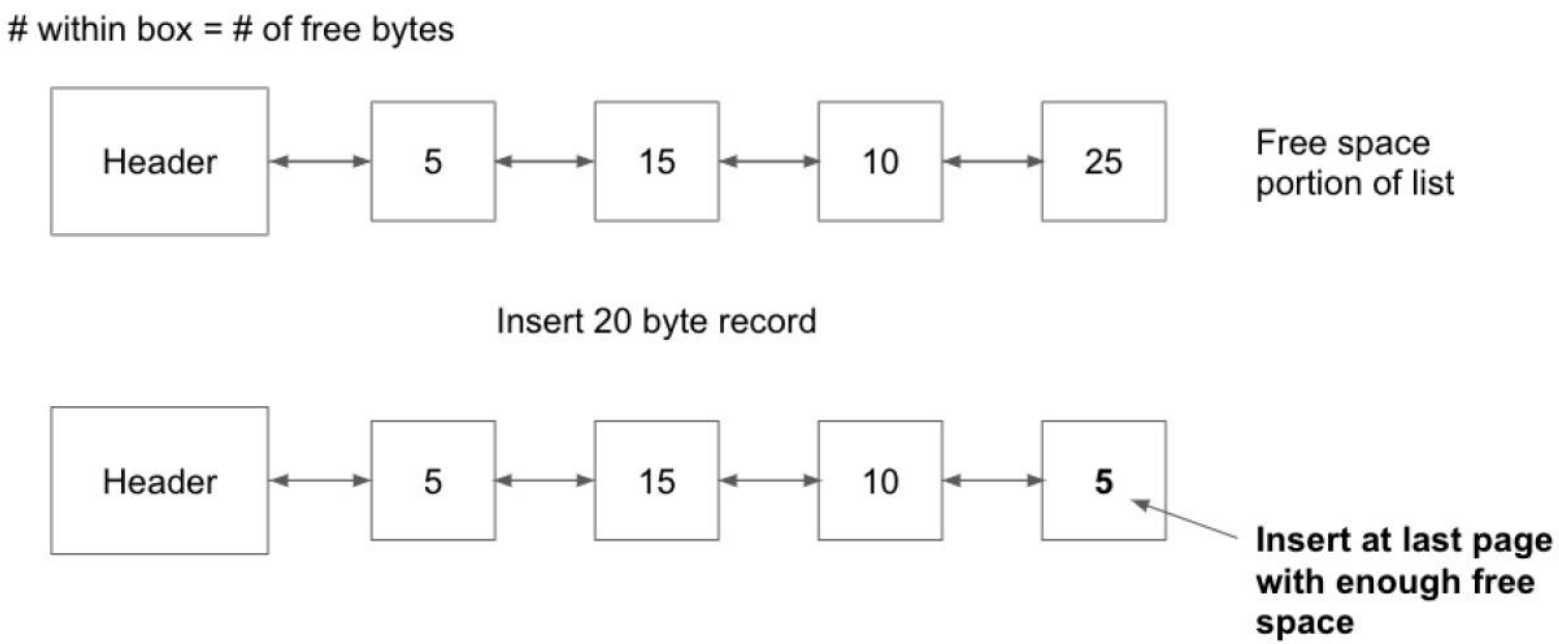
within box = # of free bytes



Free space portion of list

Insert 20 byte record

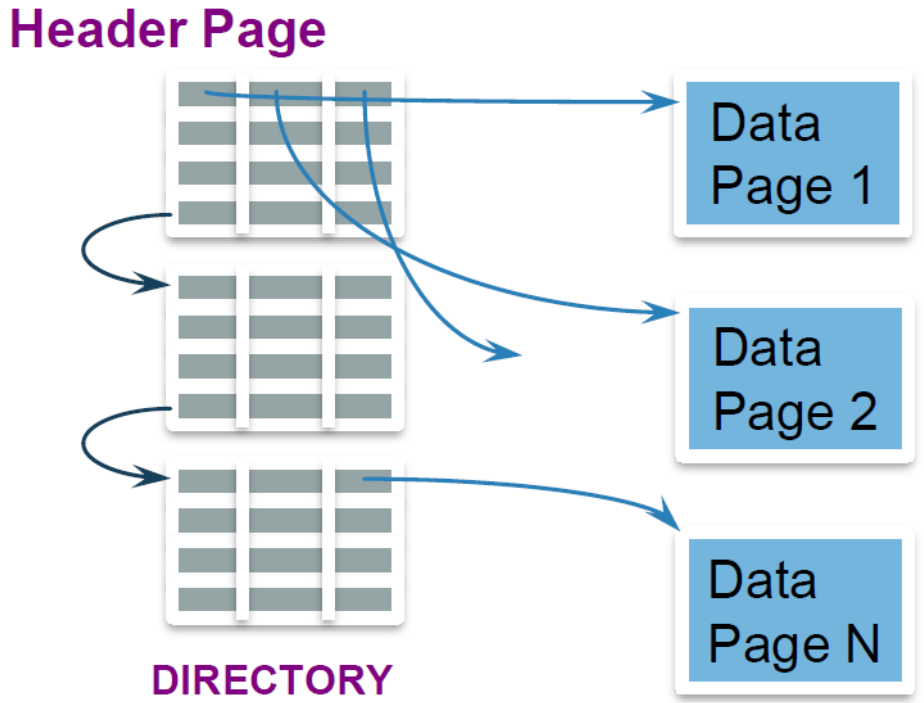
Heap Files - Linked List



I/O Cost: 5 (read all pages) + 1 (write last data page) = **6 I/Os**

Heap Files - Page Directory

- using a linked list for header pages
- Each header page contains
  - a pointer (byte offset) to the next header page
  - its entries contain
    - a pointer to a data page
    - the amount of free space left within that data page
- Inserting records is often faster
  - Read the header pages until finding a page with enough free space -> Use pointer to that data page to read the page -> Write data -> Write header



Heap Files - Page Directory

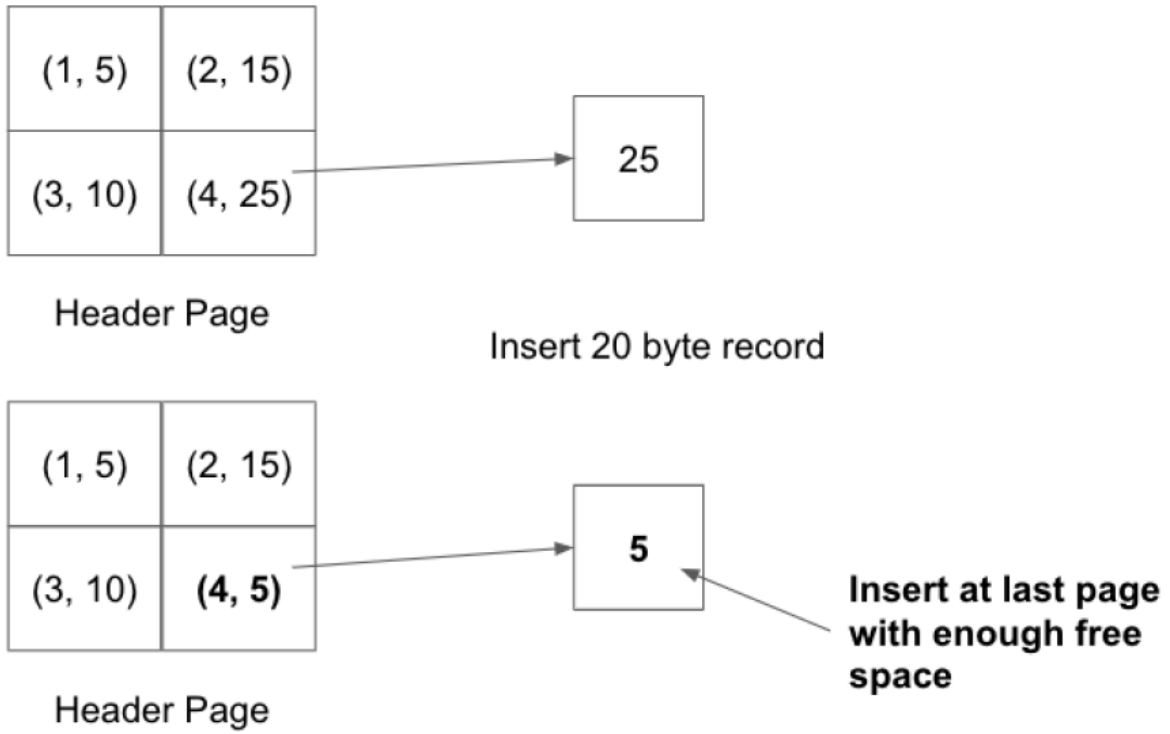
(#, #) within box = (page #, # of free bytes)

(1, 5)	(2, 15)
(3, 10)	(4, 25)

Header Page

Heap Files - Page Directory

(#, #) within box = (page #, # of free bytes)

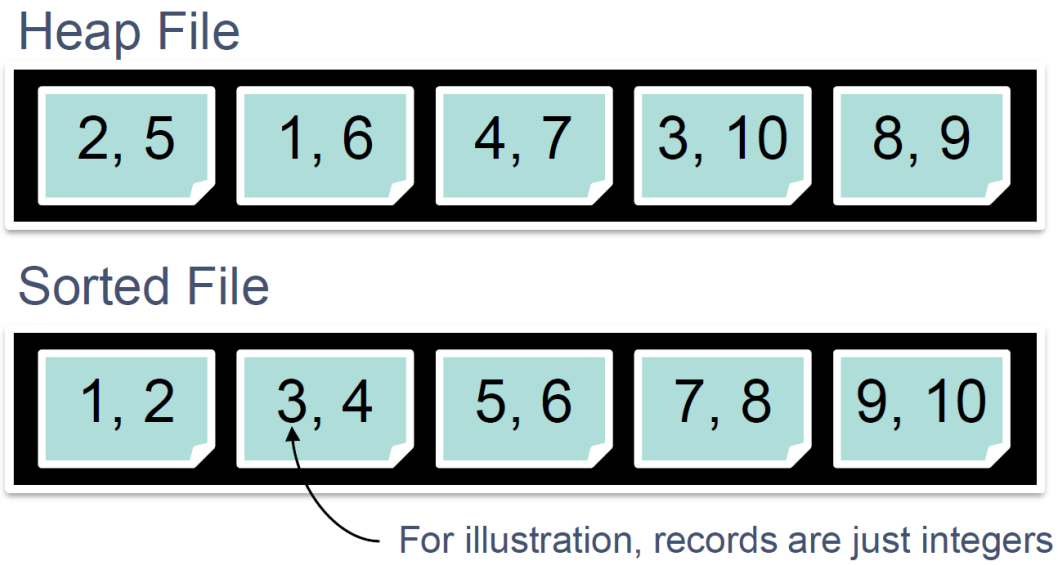


I/O Cost: 1 (read header) + 1 (read data) + 1 (write data) + 1 (write header) = **4 I/Os**

Heap Files - Note

- You should **ignore the I/O cost associated with reading/ writing the file's header pages** when the specific file implementation of heap files (i.e., heap file implemented with a linked list or page directory) **is not provided.**
- But if it is provided:
  - You must consider it

Heap Files vs Sorted Files

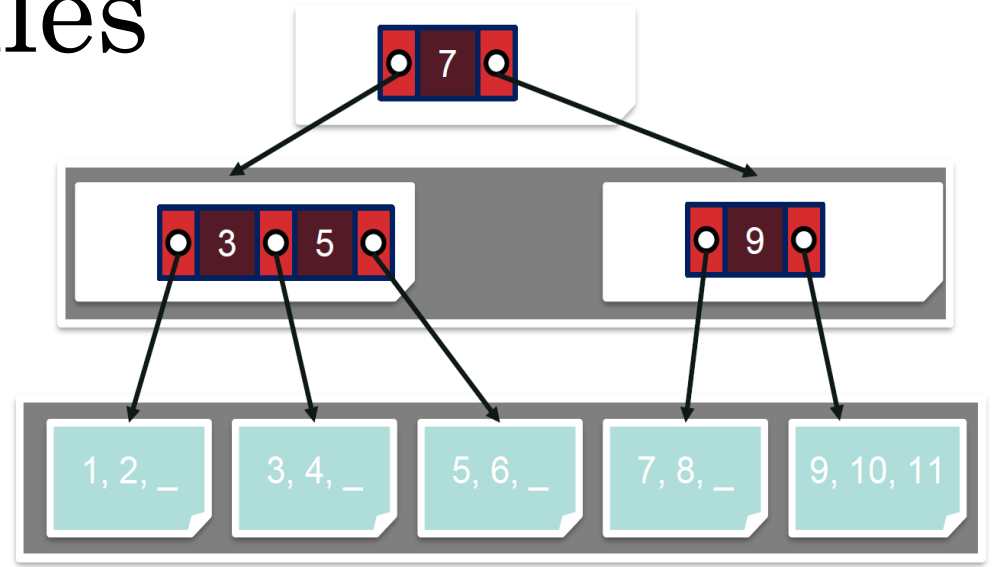


	Heap File	Sorted File
Scan all records	B*D	B*D
Equality Search	0.5*B*D	$(\log_2 B)*D$
Range Search	B*D	$((\log_2 B)+pages)*D$
Insert	2*D	$((\log_2 B)+B)*D$
Delete	(0.5*B+1)*D	$((\log_2 B)+B)*D$

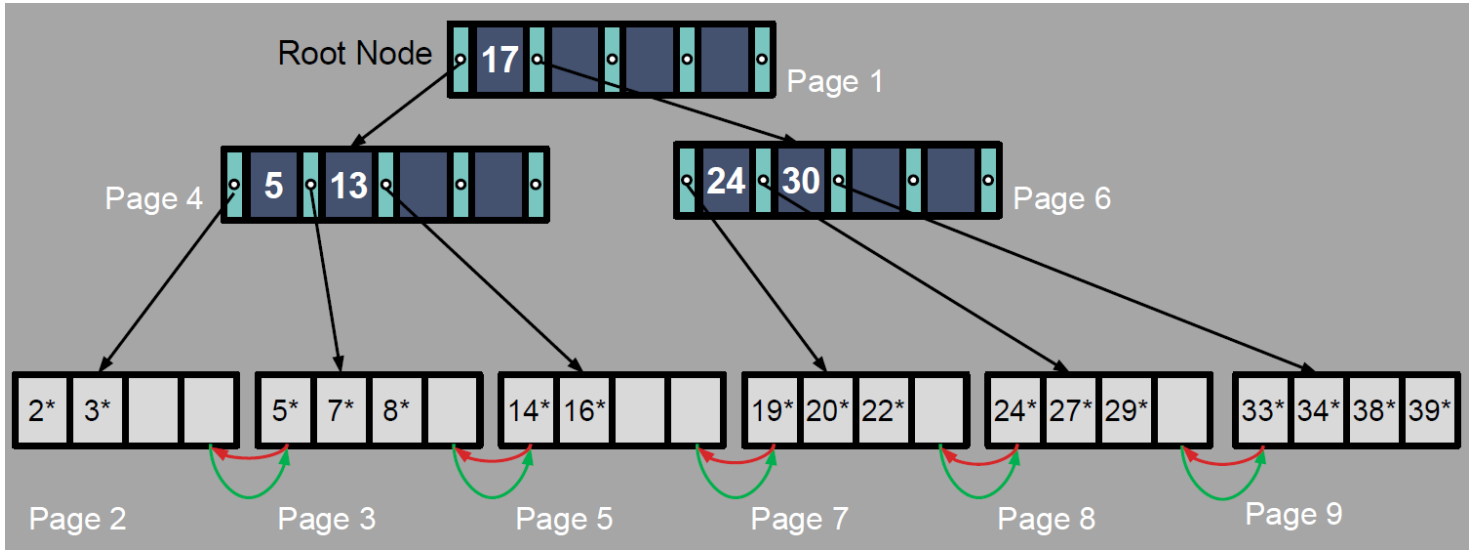
- **B:** The number of data blocks
- **R:** Number of records per block
- **D:** Average time to read/write disk block

Index Files

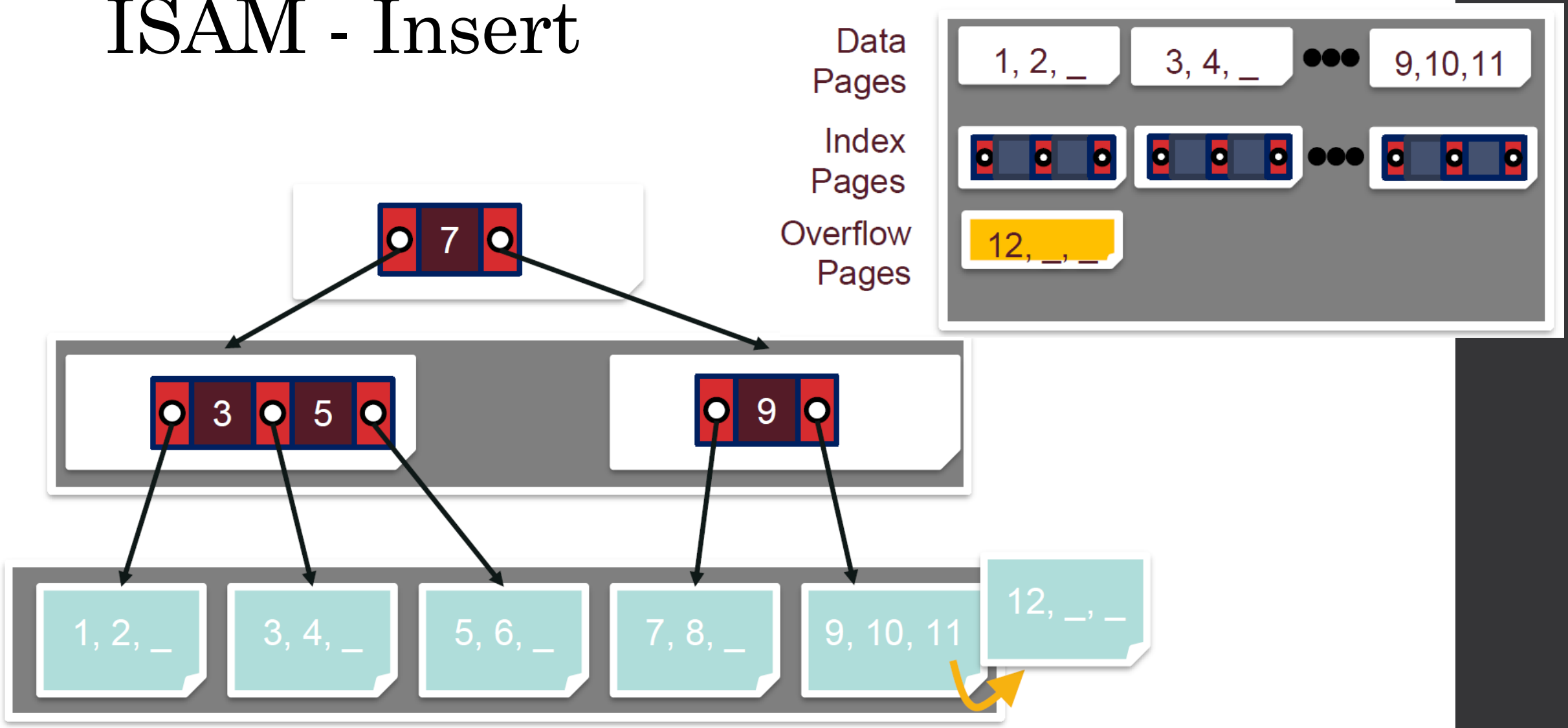
ISAM



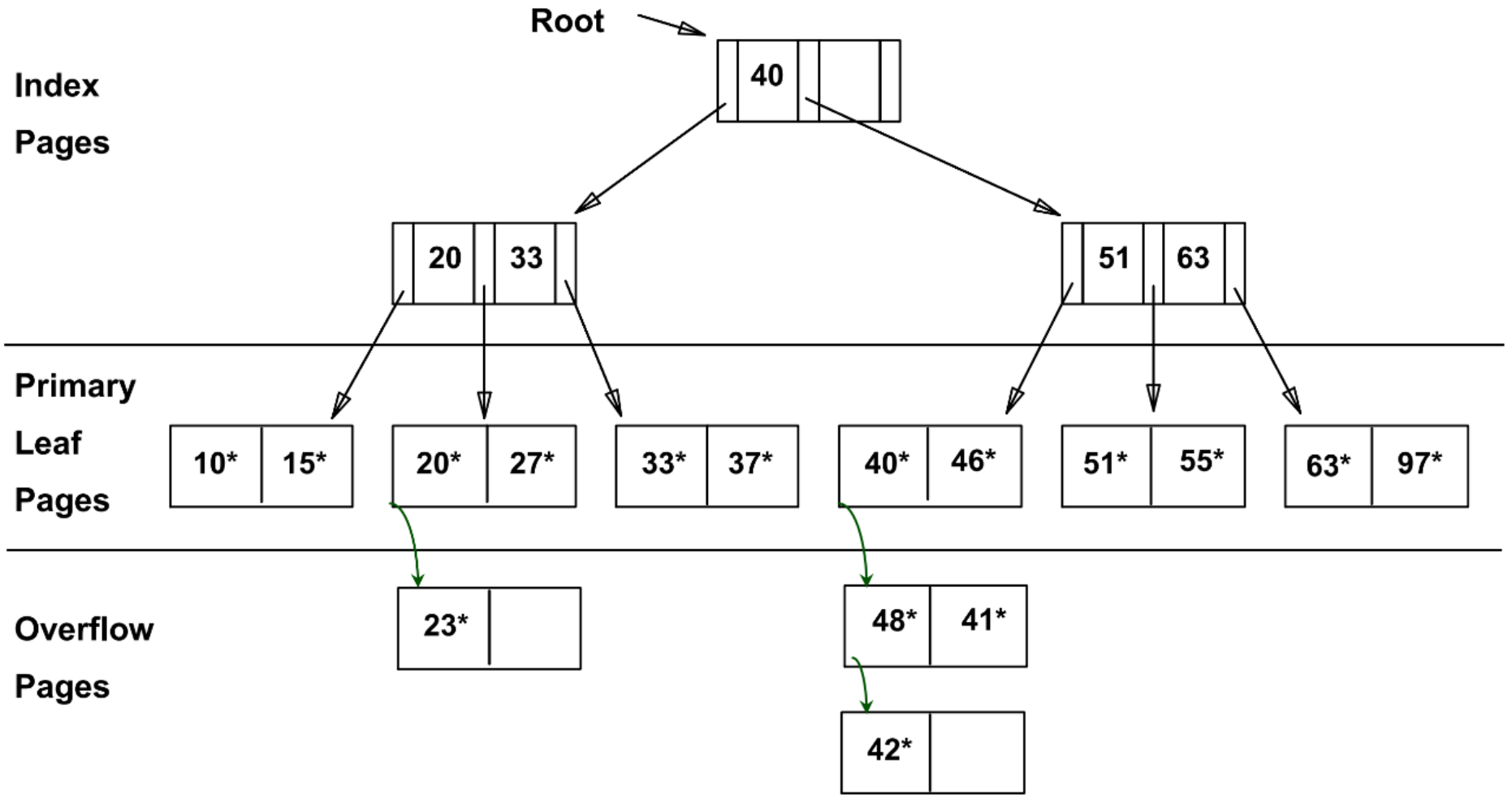
B+ Tree



ISAM - Insert

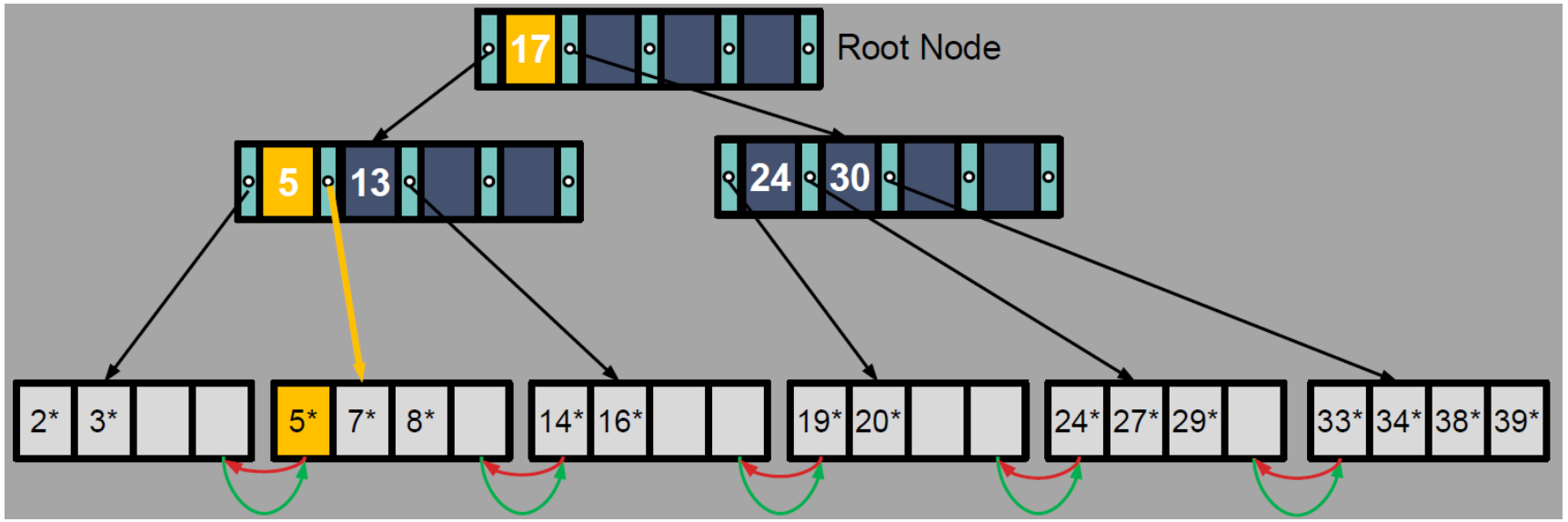

Data Pages
Index Pages
Overflow Pages
1, 2, _
3, 4, _
9,10,11
12, _, _
7
3
5
9
1, 2, _
3, 4, _
5, 6, _
7, 8, _
9, 10, 11
12, _, _

ISAM - Insert


Root
Index
Pages
40
20
33
51
63
Primary
Leaf
Pages
10*
15*
20*
27*
33*
37*
40*
46*
51*
55*
63*
97*
Overflow
Pages
23*
48*
41*
42*

B+ Tree - Insert

- If there is room in the leaf just add the entry
  - **Sort** the leaf page by key
- If no room:
  - Split-> Redistribute entries evenly -> **leaf:** Fix next/ prev pointers -> **Copy (leaf) / Push (parent)** up from leaf the middle key



Index Files

ISAM

- Data pages at bottom stored in logical order
- Insert into overflow pages
  - Overflow chains can degrade performance unless size of data set and data distribution stay constant
- Only leaf nodes are allowed to be modified, not tree nodes.

B+ Tree

- Data pages at bottom:
  - Use next and prev pointers
  - Don't need to be stored in logical order
- Dynamic Tree Index
  - Always Balanced
  - No overflow pages, more efficient
  - Require Occupancy Invariant (except root node): **d <= #entries <= 2d**
  - High fanout (F) means depth rarely more than 3 or 4
- Can dynamically modify root node, tree node, and leaf nodes.

Quiz 3

Q1: Suppose that all nodes in our B+ tree have an order of 1500. What's the MAXIMUM number of records we can index with a B+ tree of height 2?

Assume our B+ trees are laid out as in lecture.

Quiz 3

Q2: We want to bulk-load a B+ tree, and we increase the fill factor of this bulk load. Which of the following applies, in general?

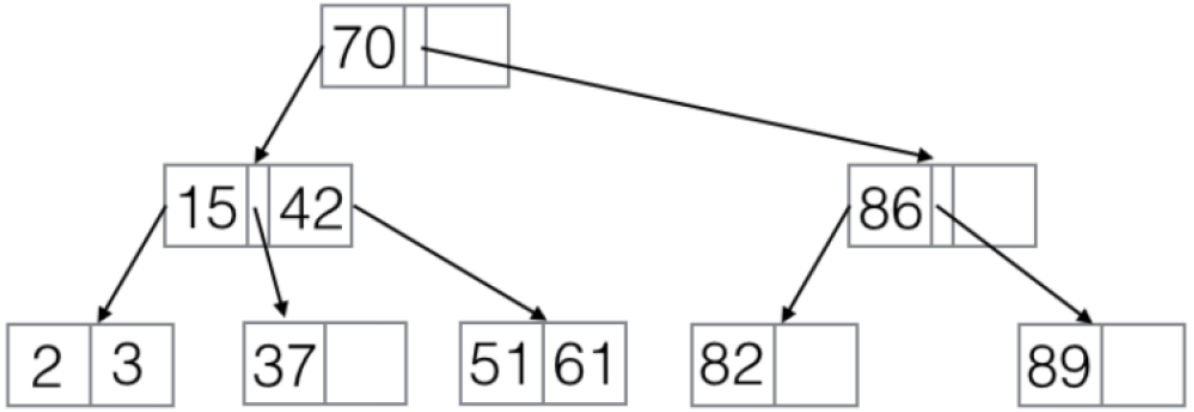
Check all that apply.

A. The bulk loading operation is faster
B. The bulk loading operation is slower
C. We consume more disk space
D. A sequence of many consecutive record lookups is faster
E. A sequence of many consecutive insertions requires fewer disk writes

Quiz 3

Q3: We insert the key 60 into the B+ tree in Figure A. How many I/Os (page reads and writes) does this operation take?

Assume we require zero page reads and one page write to create a new page from scratch. Also assume that we do no key redistribution. Exclude disk I/Os done to data pages. Finally, assume we have 20 pages of memory available for caching pages in memory after reading them.



Quiz 3