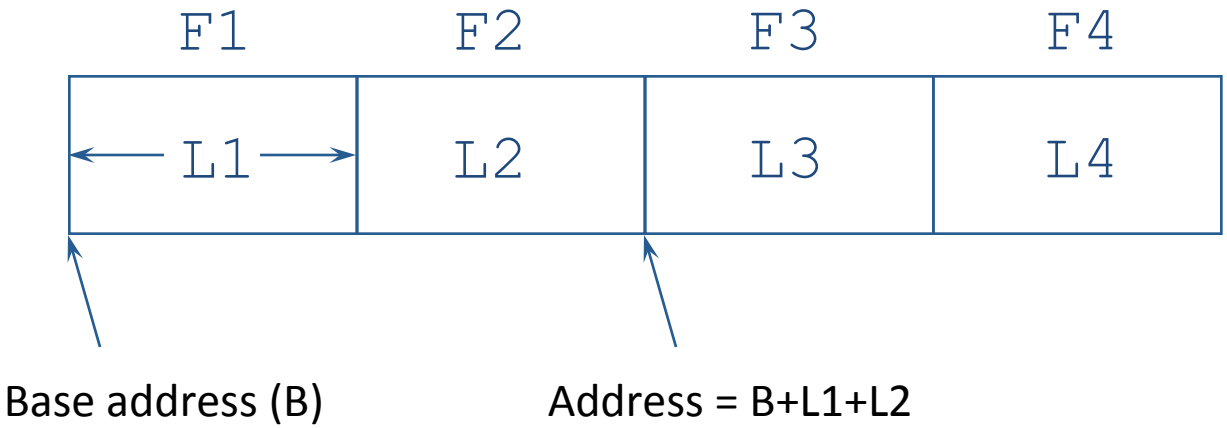
CS186 Discussion 4

(Record Formats, File Organization, Indexes, B+ Trees)

Matthew Deng

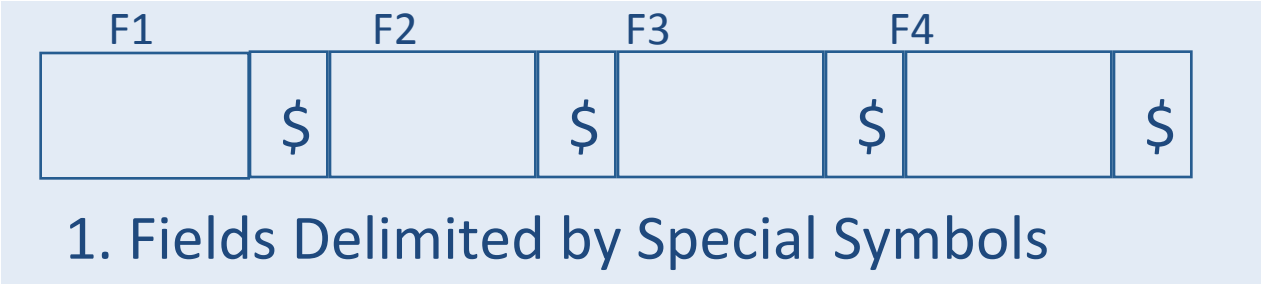
Record Formats

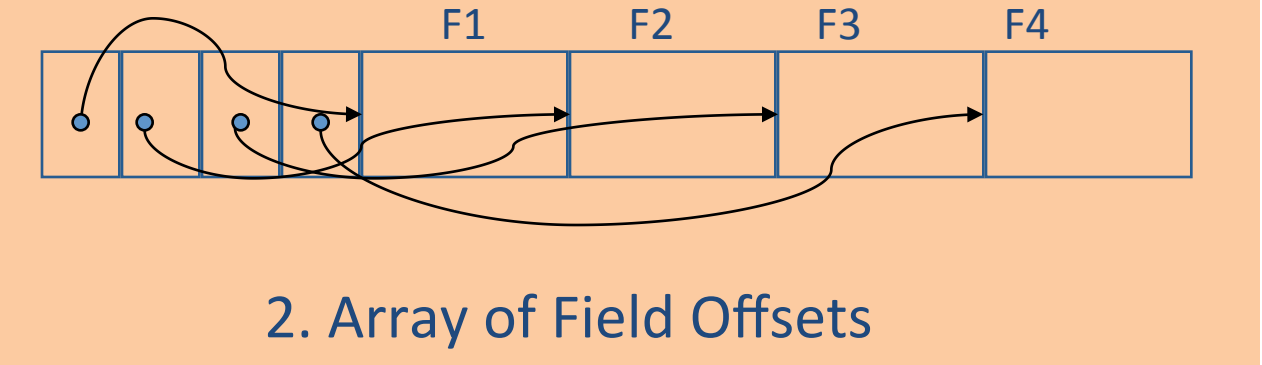
Record Formats: Fixed Length



use this when you can!

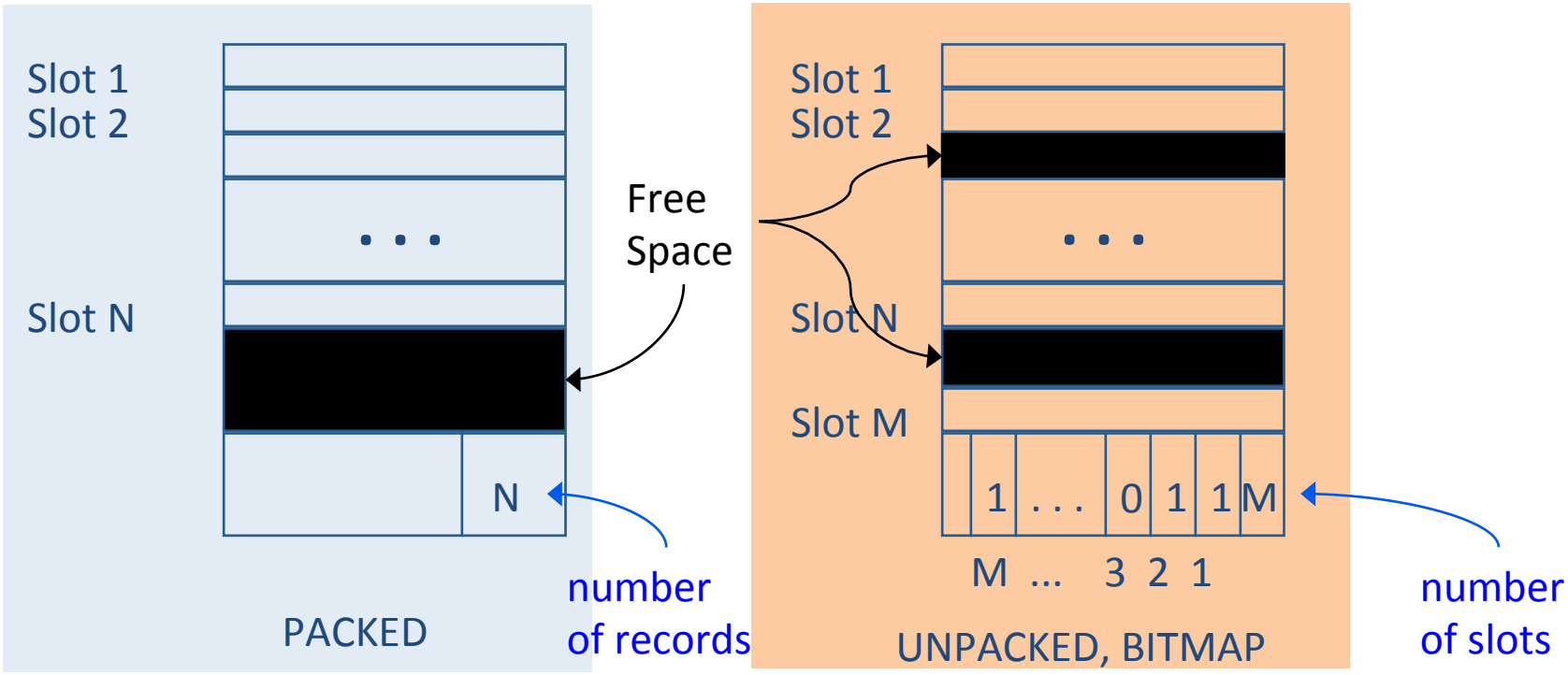
Record Formats: Variable Length





symbols can cause problems

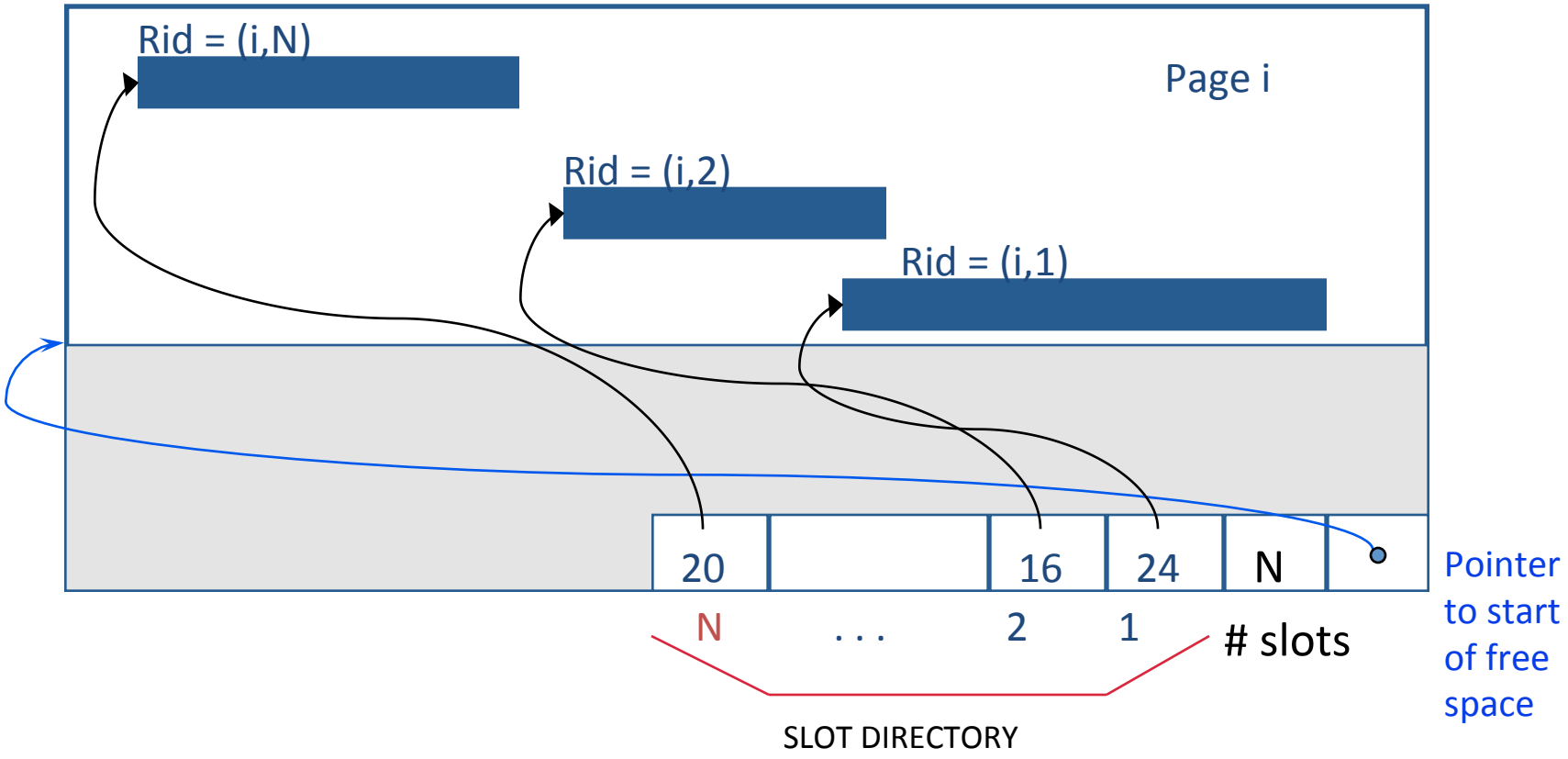
Page Formats: Fixed Length



record id = <page id, slot number>

free space in unpacked spread out

Page Formats: Variable Length



can move records without changing rid

Worksheet: Record Formats

Record Formats Exercises

1. What's the tradeoff between fixed length fields and variable length fields?

Record Formats Exercises

1. What's the tradeoff between fixed length fields and variable length fields?

   fixed length has less bookkeeping overhead, but can be memory-inefficient

Record Formats Exercises

2. What is fragmentation?

Record Formats Exercises

2. What is fragmentation?

   when free memory is divided into small chunks

Record Formats Exercises

3. What does slotted page header contain?

Record Formats Exercises

3. What does slotted page header contain?

   number of record entries
   end of free space in the block
   location and size of each record

File Organization

File Organization

	Heap File	Sorted File
Scan All		
Equality Search		
Range Search		
Insertion		
Deletion		

File Organization

	Heap File	Sorted File
Scan All	B	B
Equality Search	.5B	$\log_2(B)$
Range Search	B	$\log_2(B)$ + num_matches
Insertion	2	$\log_2(B)$ + 2 * .5B
Deletion	.5B + 1	$\log_2(B)$ + 2 * .5B

Worksheet:
File Organization

File Organization Exercises

1. What should we consider when optimizing our file organization?

File Organization Exercises

1. What should we consider when optimizing our file organization?

   How often we need to update records

   How fast the worst case search/scan is

File Organization Exercises

2. When should we use heap files? Sorted files?

File Organization Exercises

2. When should we use heap files? Sorted files?

   How often we need to update records

   How fast the worst case search/scan is

Indexes

Indexes

- Index – Disk-based data structure for fast lookup by value
- Data entry: k*
  - <k, {items}>
- Hash Indexes
  - Equality search
- Tree Indexes
  - Equality search
  - Range search

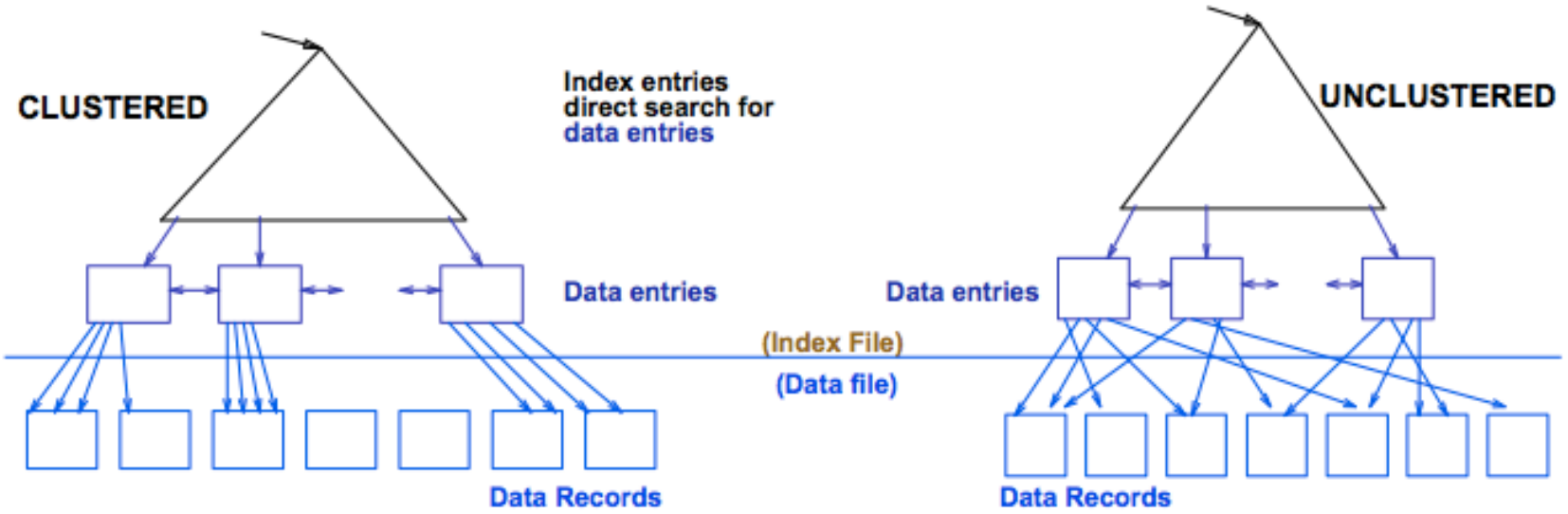
Alternatives

There are three alternatives for storing data entries in index:

1. actual data
   - there can only be one alternative 1 index per table
2. key and record ID
   - <k, rid>
3. key and list of record IDs
   - <k, [rids]>

Clustering

- Clustered Index:
  - index data entries are stored in (approximate) order by value of search key in data records
  - A file can be clustered on at most one search key
  - Alternative 1 is always clustered
  - Heap file usually packed to 2/3 capacity



File Organization

	Heap File	Sorted File
Scan All		
Equality Search		
Range Search		
Insertion		
Deletion		

File Organization

	Heap File	Sorted File
Scan All	B	B
Equality Search	.5B	$\log_2(B)$
Range Search	B	$\log_2(B)$ + num_matches
Insertion	2	$\log_2(B)$ + 2 * .5B
Deletion	.5B + 1	$\log_2(B)$ + 2 * .5B

File Organization

	Heap File	Sorted File	Clustered File
Scan All	B	B	
Equality Search	.5B	$\log_2(B)$	
Range Search	B	$\log_2(B)$ + num_matches	
Insertion	2	$\log_2(B)$ + 2 * .5B	
Deletion	.5B + 1	$\log_2(B)$ + 2 * .5B	

File Organization

	Heap File	Sorted File	Clustered File
Scan All	B	B	1.5B
Equality Search	.5B	$\log_2(B)$	$\log_F(1.5B) + 1$
Range Search	B	$\log_2(B)$ + num_matches	$\log_F(1.5B)$ + num_matches
Insertion	2	$\log_2(B) + 2 * .5B$	$\log_F(1.5B) + 2$
Deletion	.5B + 1	$\log_2(B) + 2 * .5B$	$\log_F(1.5B) + 2$

Worksheet: Indexes

Indexes Exercises

1. What are indexes and why do we have them?

Indexes Exercises

1. What are indexes and why do we have them?

   to find records faster over a variety of workloads

Indexes Exercises

2. What are clustered indexes?

Indexes Exercises

2. What are clustered indexes?

   the index whose search key specifies the sequential order of the file

Indexes Exercises

3. What is a unclustered index and how is it different from a clustered index?

Indexes Exercises

3. What is a unclustered index and how is it different from a clustered index?

   the table file is not ordered by the unclustered index field, so range queries on a unclustered index are a lot worse than those on a clustered index

Indexes Exercises

4. How is data stored in a index? Describe the three alternatives.

Indexes Exercises

4. How is data stored in a index? Describe the three alternatives.

   (A1) By value: entire row in leaf

   (A2) By reference: (key, rid) pairs, with each key occuring once

   (A3) By reference: (key, [rid]) pairs -- difference from A2 is that all rids with the same key are in one index entry

Indexes Exercises

5. What's the difference between multiple indexes and a multi-column index?

Indexes Exercises

5. What's the difference between multiple indexes and a multi-column index?

   Multiple indexes are constructed on different columns individually, while a multi-column index is constructed on a concatenation of columns.

Indexes Exercises

6. What if you want two clustering indices on separate columns?

Indexes Exercises

6. What if you want two clustering indices on separate columns?

   Either:

   create a multi-column index (must do lexicographic search; can't search on second column alone)

   or

   duplicate the table and have two clustering indices (must duplicate writes -- less efficient).

Indexes Exercises

7. What is data locality and why do we care?

Indexes Exercises

7. What is data locality and why do we care?

   Spatial locality: good for finding data in shorter time (less movement)

   Cache locality: good for I/O

Indexes Exercises

8. What disadvantages are there to using indexes?

Indexes Exercises

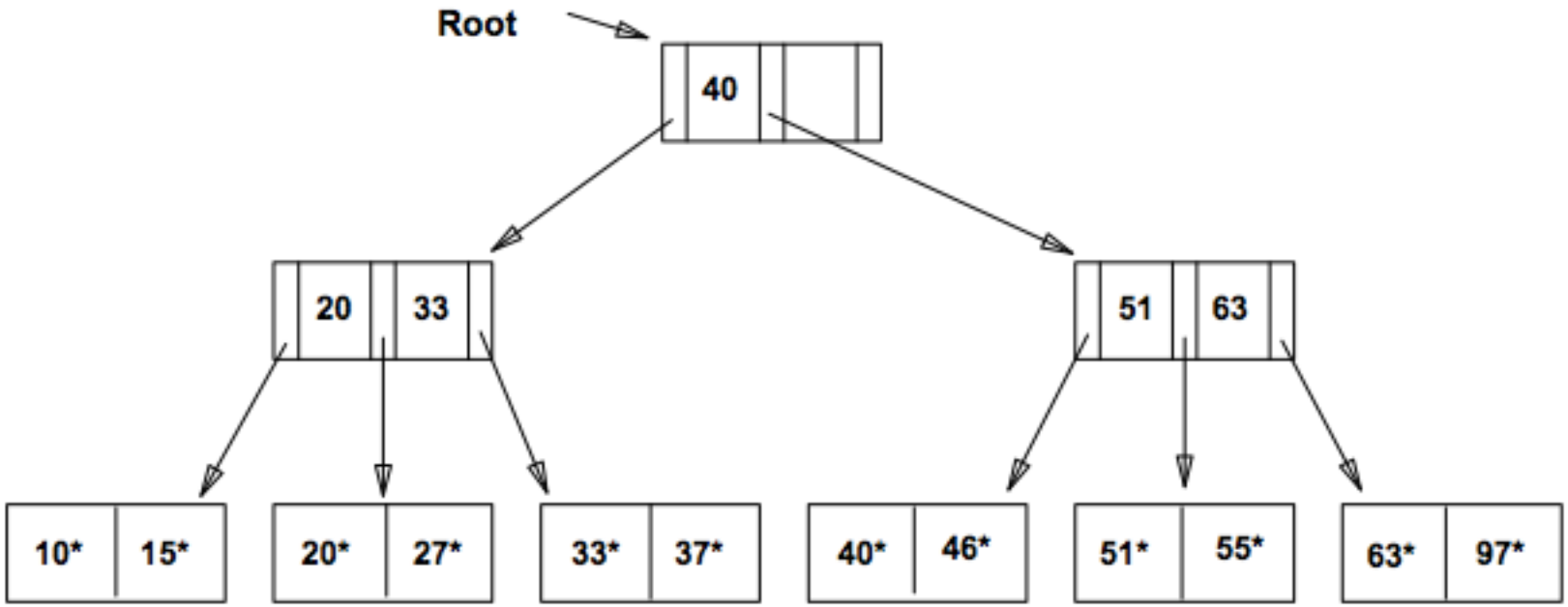
8. What disadvantages are there to using indexes?

   Maintenance cost
   clustered indexes force us to maintain a sorted file

Tree-Structured Indexes

ISAM

- Indexed Sequential Access Method
- Static data structure
- No longer used



ISAM Insertion

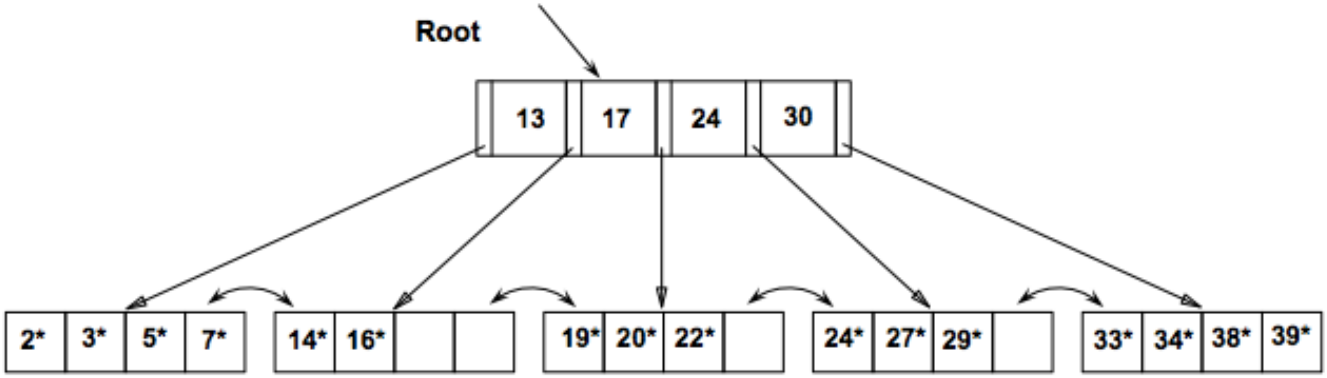
```
find leaf node L

if L not full:
    insert data entry

else:
    if overflow page not full:
        insert data entry
    else:
        create overflow page
        insert data entry
```

B+ Trees

- Dynamic data structure
- Most popularly used index
- Height balanced
- Terminology:
  - Order d: every node has between d and 2d entries
  - Fanout F: branches from a node (max: 2d + 1, average: 2/3 * 2d)
  - Number of leaf pages N



B+ Tree Insertion

```
find leaf node L

if L not full:
    insert data entry

else:                                          // 2d+1 in L
    split L into L and L2                      // d in L, d+1 in L2
    copy up middle key to parent P

    while P past capacity:                     // 2d+1 in L
        split P into P and P2                  // d in P, d in P2
        push up middle key to parent P
```

Worksheet:
B+ Trees

B+ Trees Exercises

1. What is the difference between an ISAM and B+ Tree Index?

B+ Trees Exercises

1. What is the difference between an ISAM and B+ Tree Index?

   ISAM Tree: Static structure

   - Consists of root, primary leaf pages, overflow pages.
   - Long overflow chains can develop.

   B+ Tree: Dynamic structure.

   - Height balanced.
   - Usually preferable to ISAM.
   - Order d: Each node contains $d \leq m \leq 2d$ entries (except for the root!)
   - Height: Length of longest path from the root to a leaf node
   - Fanout of a node: The number of pointers out of the node

B+ Trees Exercises

2. What seems to be the tradeoff between index designs?

B+ Trees Exercises

2. What seems to be the tradeoff between index designs?

   static vs. dynamic maintenance

   impact on data lookup and data insertion at runtime

   impact on storage requirements

B+ Trees Exercises

3. Why do you think B+ trees are so popular?

B+ Trees Exercises

3. Why do you think B+ trees are so popular?

   Adaptive to writes and balanced

B+ Trees Exercises

4. In the insert algorithm for a B+tree, what happens when the inserted leaf doesn't have enough space?

B+ Trees Exercises

4. In the insert algorithm for a B+tree, what happens when the inserted leaf doesn't have enough space?

   must split L (into L and a new node L2)

   then redistribute entries evenly

   copy up middle key

   insert index entry pointing to L2 into parent of L

   This can happen recursively!

B+ Trees Exercises

5. True/False

   Leaves can have different distance from root.

   Every non-leaf, non-root node has at least floor(d / 2) children.

   Leaf nodes cannot be full.

B+ Trees Exercises

5. True/False

   Leaves can have different distance from root. F

   Every non-leaf, non-root node has at least floor(d / 2) children.

   Leaf nodes cannot be full.

B+ Trees Exercises

5. True/False

   Leaves can have different distance from root. F

   Every non-leaf, non-root node has at least floor(d / 2) children. T

   Leaf nodes cannot be full.

B+ Trees Exercises

5. True/False

   Leaves can have different distance from root. F

   Every non-leaf, non-root node has at least floor(d / 2) children. T

   Leaf nodes cannot be full. F

B+ Trees Exercises

6. Can you think of a better way to load data with a B+ tree than inserting one by one?

B+ Trees Exercises

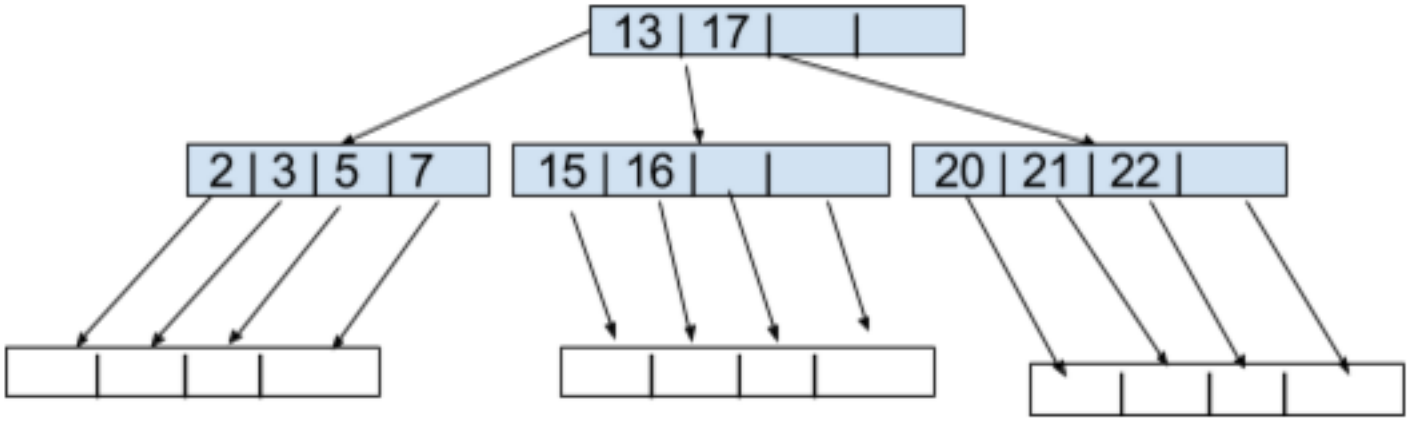
6. Can you think of a better way to load data with a B+ tree than inserting one by one?

   Sort the data entries, then build the tree up layer by layer.

   Really good for concurrency and space utilization.
   (read about bulk inserts if you want to learn more)

B+ Trees Exercises

7. Draw the tree after we insert the key 9.



B+ Trees Exercises

7. Draw the tree after we insert the key 9.