CS186 Discussion 5

(Indexes, Tree-Structured Indexes)

Matthew Deng

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

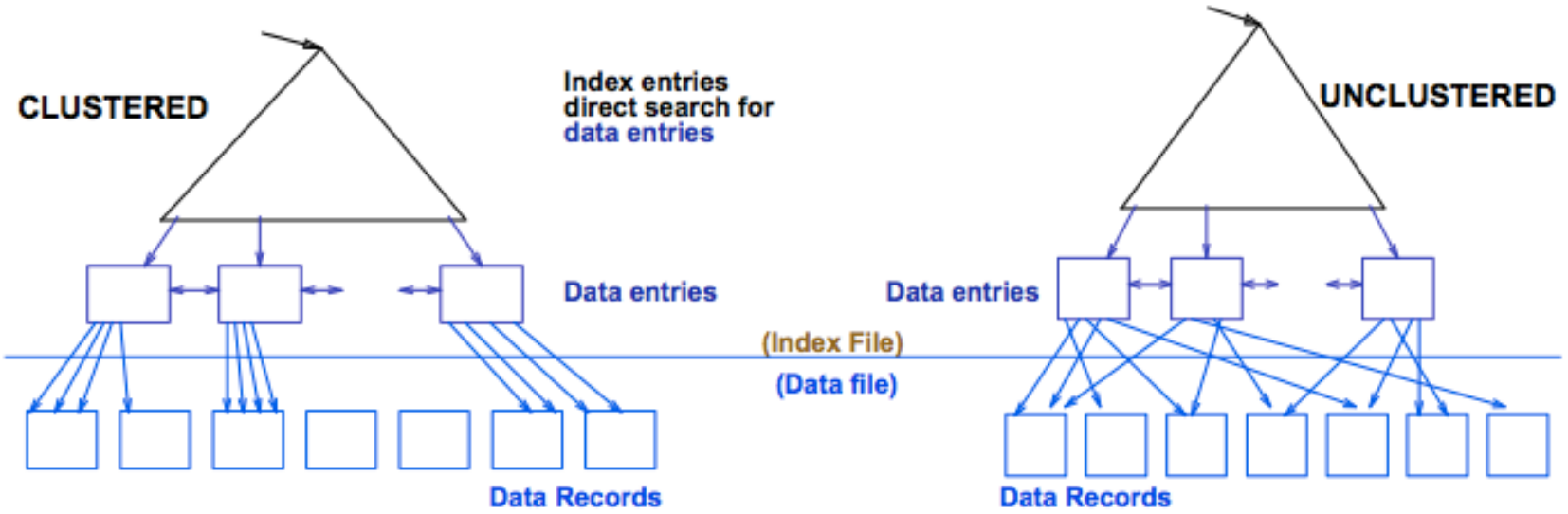
Alternatives

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	$\log_2(B) + 2 * .5B$	$\log_F(1.5B) + 2$

Indexing Worksheet

Indexing Exercises: Basics

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

Indexing Exercises: Basics

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

   An index is a data structure that organizes data records on disk to optimize certain kinds of retrieval operations. An index allows us to efficiently retrieve all records that satisfy search conditions on the search key fields of the index.

Indexing Exercises: Basics

2. Compare the two main types of indexes.

Indexing Exercises: Basics

2. Compare the two main types of indexes.

Hash based index: Cannot do range queries.
Slightly faster on equality search.

Tree based index:

Unclustered: Fast equality search, insertion, deletion.

Clustered: Fast search, insertion, deletion, scan,
range search. Needs space.

Indexing Exercises: Basics

3. What are important factors in determining whether or not you should add an index to add to a table?

Indexing Exercises: Basics

3. What are important factors in determining whether or not you should add an index to add to a table?

- Should know which field to cluster on
  - (calculate based on typical queries run)
- Decide if you even want to cluster
  - (high maintenance cost)

Indexing Exercises: Alternatives

1. What are ‘alternatives’?

Indexing Exercises: Alternatives

1. What are ‘alternatives’?

   Alternatives are the different ways of storing a data entry.

   Data entry - an index file representation of record.

   Three alternatives:

   1. Actual data record (with key value k)
   2. <k, rid of matching data record>
   3. <k, list of rids of matching data records>

Indexing Exercises: Alternatives

2. Say we have the following records. Use the boxes to show examples of each data entry alternative, assuming we build an index on age.

```
Clowns (Name text, SID int, Experience int, Age int)
RID: 022 - Casey | 134234 | 2 | 35
RID: 051 - Merk | 955551 | 2 | 35
RID: 101 - Test | 045671 | 11 | 15
```

Alternative 1	Alternative 2	Alternative 3

Indexing Exercises: Alternatives

2. Say we have the following records. Use the boxes to show examples of each data entry alternative, assuming we build an index on age.

```
Clowns (Name text, SID int, Experience int, Age int)
RID: 022 - Casey | 134234 | 2 | 35
RID: 051 - Merk | 955551 | 2 | 35
RID: 101 - Test | 045671 | 11 | 15
```

Alternative 1	Alternative 2	Alternative 3
Casey | 134234 | 2 | 35 Merk | 955551 | 2 | 35 Test | 045671 | 11 | 15 (RID is implicit)	<key, RID> <35, 022> <35, 051> <15, 101>	<key, [RID list]> <35, [022, 051]> <15, [101]>

Indexing Exercises: Clustering

1. What are clustered indexes?

Indexing Exercises: Clustering

1. What are clustered indexes?

   When a (data) file is organized so that the ordering of data records is the same as or close to the ordering of data entries in some index, we say that the index is clustered

Indexing Exercises: Clustering

2. How do they relate to data entry alternatives?

Indexing Exercises: Clustering

2. How do they relate to data entry alternatives?

   An index that uses Alternative (1) is clustered, by definition. An index that uses Alternative (2) or (3) can be a clustered index only if the data records are sorted on the search key field.

Indexing Exercises: Clustering

3. True or False? Given the table `Students(sid char(20), gpa float, age integer)`, a clustered tree based index on `gpa` will increase the performance of the following query: `SELECT * FROM Students where age > 20 AND gpa > 3.5;`

Indexing Exercises: Clustering

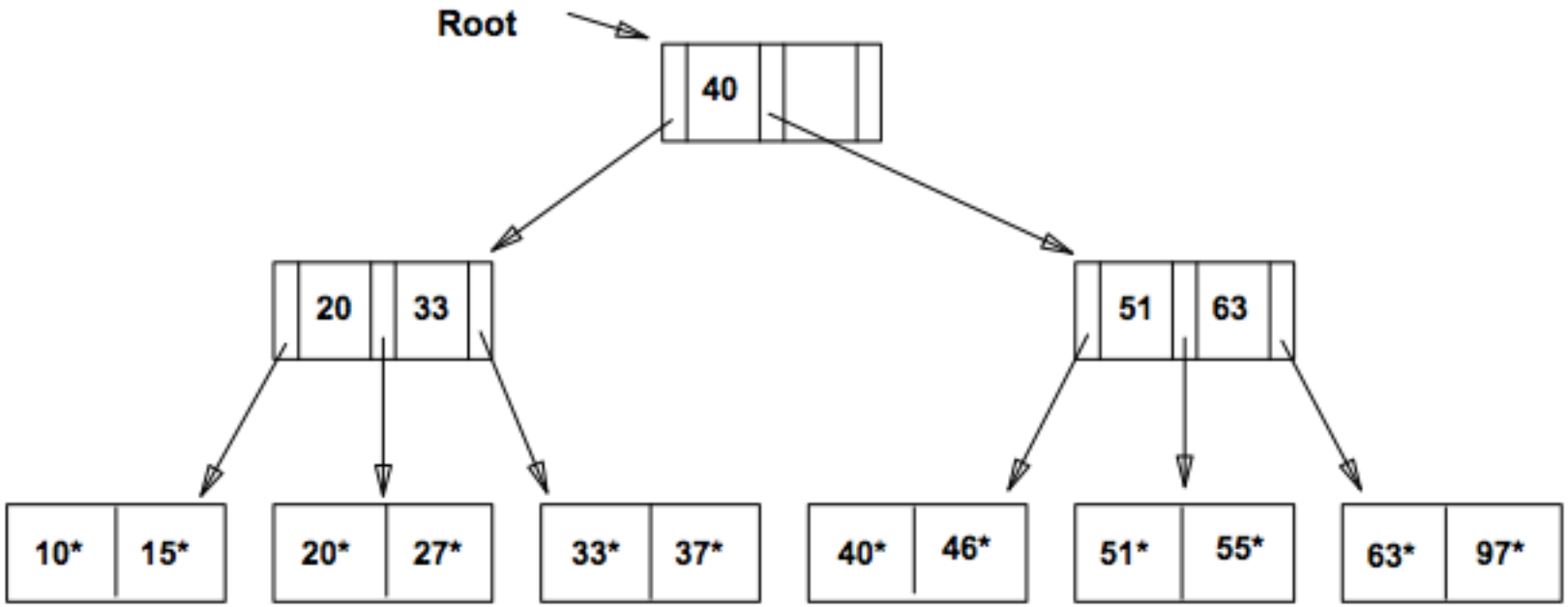
3. True or False? Given the table `Students(sid char(20), gpa float, age integer)`, a clustered tree based index on `gpa` will increase the performance of the following query: `SELECT * FROM Students where age > 20 AND gpa > 3.5;`

   True

Tree-Structured Indexes

ISAM

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



ISAM Insertion

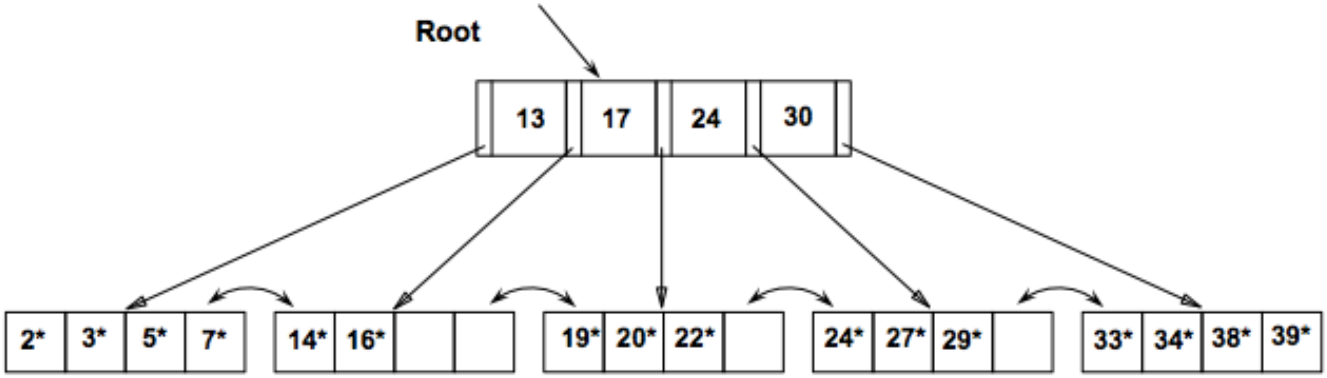
```
find leaf node L

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
```

Tree-Structured Indices Worksheet

Tree-Structured Indices Exercises

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

Tree-Structured Indices Exercises

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

ISAM Tree: Static structure. Consists of root, primary leaf pages and overflow pages. Long overflow chains can develop.

B+ Tree: Dynamic structure. Height balanced. Usually preferable to ISAM.

Order d: Each node contains entries

Height: Length of Path from the root to a leaf node

Fanout: The number of pointers out of the node

Tree-Structured Indices Exercises

2. We are using a B+ tree with alternative 1 (actual data records in leaf pages) to store one billion records. Each records is 200 bytes, each disk page has 16kB (16,384 Bytes) and will always be at most 67% full.

   a. Approximately many leaf pages are required?

Tree-Structured Indices Exercises

2. We are using a B+ tree with alternative 1 (actual data records in leaf pages) to store one billion records. Each records is 200 bytes, each disk page has 16kB (16,384 Bytes) and will always be at most 67% full.

   a. Approximately many leaf pages are required?

   16384 * 0.67 / 200 = ~54 Entries per page.

   10^9 / 54 = ~18.5 * 10^6 pages.

Tree-Structured Indices Exercises

2. We are using a B+ tree with alternative 1 (actual data records in leaf pages) to store one billion records. Each records is 200 bytes, each disk page has 16kB (16,384 Bytes) and will always be at most 67% full.

   b. Assume each index entry takes 32 bytes. What is approximately the maximum fanout of the index?

Tree-Structured Indices Exercises

2. We are using a B+ tree with alternative 1 (actual data records in leaf pages) to store one billion records. Each records is 200 bytes, each disk page has 16kB (16,384 Bytes) and will always be at most 67% full.

   b. Assume each index entry takes 32 bytes. What is approximately the maximum fanout of the index?

   16384 * 0.67 / (32) = ~343

Tree-Structured Indices Exercises

2. We are using a B+ tree with alternative 1 (actual data records in leaf pages) to store one billion records. Each records is 200 bytes, each disk page has 16kB (16,384 Bytes) and will always be at most 67% full.

   c. What is the height (# levels of non-leaf nodes) of the tree How many I/O operations are required to insert a new record (assuming there is enough space in the leaf page)?

Tree-Structured Indices Exercises

2. We are using a B+ tree with alternative 1 (actual data records in leaf pages) to store one billion records. Each records is 200 bytes, each disk page has 16kB (16,384 Bytes) and will always be at most 67% full.

   c. What is the height (# levels of non-leaf nodes) of the tree How many I/O operations are required to insert a new record (assuming there is enough space in the leaf page)?

   Height = $\log_{343}(18.5 * 10\text{^}6) = \sim 3$.

   3 non-leaf reads + 1 leaf read + 1 write = 5 I/O's.

Tree-Structured Indices Exercises

2. We are using a B+ tree with alternative 1 (actual data records in leaf pages) to store one billion records. Each records is 200 bytes, each disk page has 16kB (16,384 Bytes) and will always be at most 67% full.

   d. How many pages are required to store the non-leaf nodes?

Tree-Structured Indices Exercises

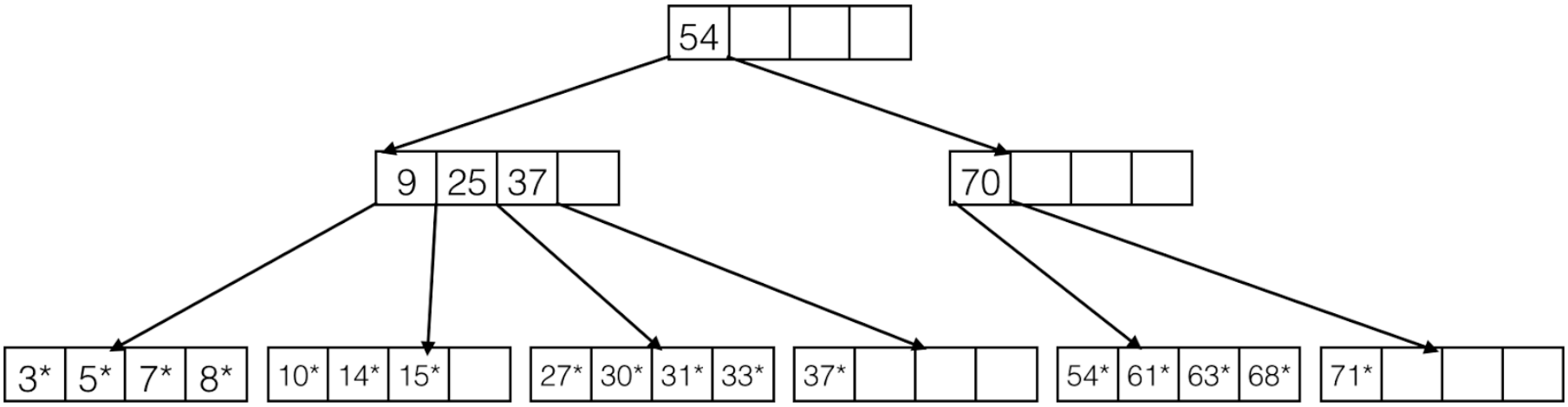
2. We are using a B+ tree with alternative 1 (actual data records in leaf pages) to store one billion records. Each records is 200 bytes, each disk page has 16kB (16,384 Bytes) and will always be at most 67% full.

   d. How many pages are required to store the non-leaf nodes?

   1 + 343 + 343^2 = 117993

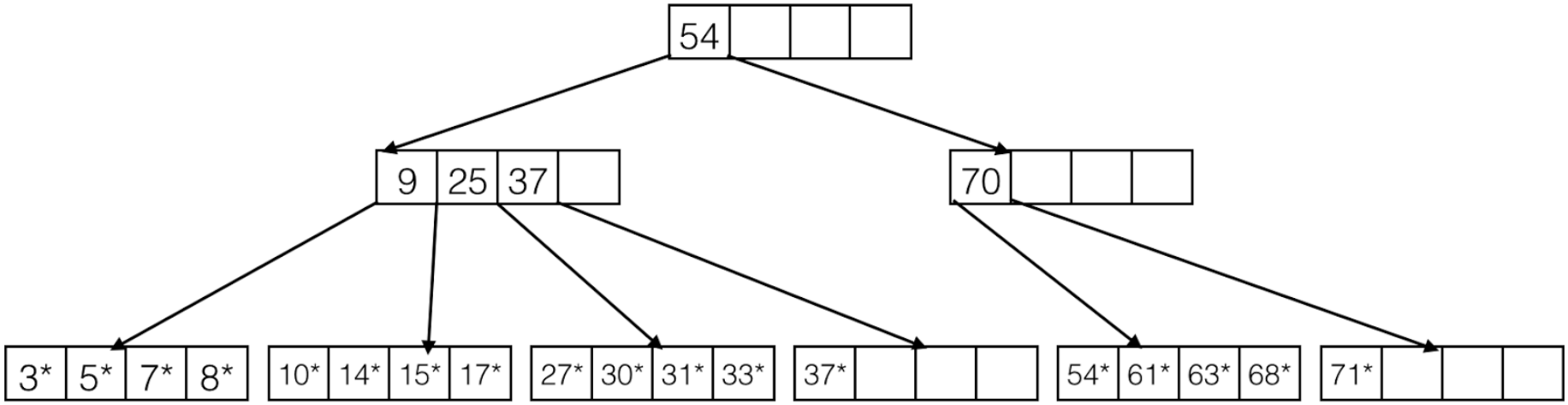
B+ Trees Exercises

Consider the B+ Tree below and insert the following in order:
17, 18, 29



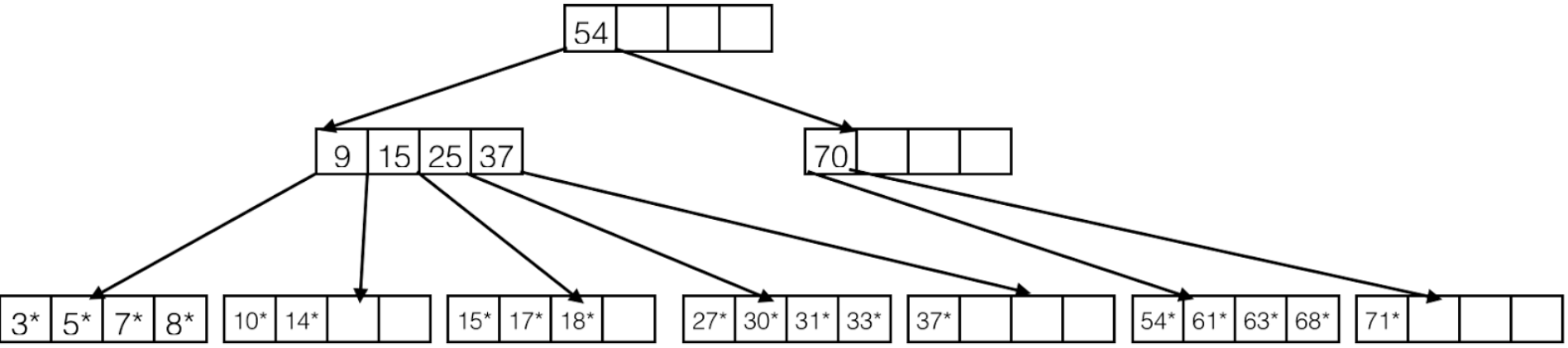
B+ Trees Exercises

Consider the B+ Tree below and insert the following in order:
17, 18, 29



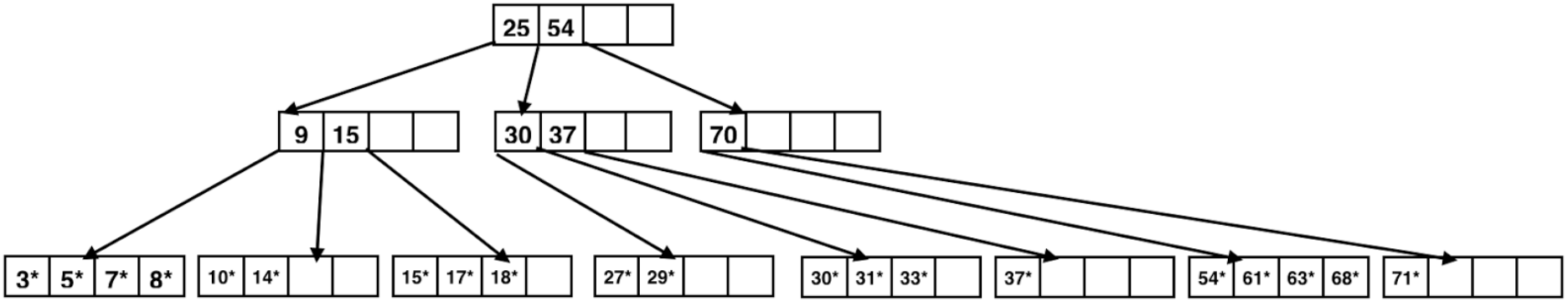
B+ Trees Exercises

Consider the B+ Tree below and insert the following in order:
17, 18, 29



B+ Trees Exercises

Consider the B+ Tree below and insert the following in order:
17, 18, 29



Counting IOs Exercises

Assume that there are 2 million users in your database. Page size is 16kB. Each user entry is 2kB in size. You are performing range queries based on a user's age. Answer the following questions:

a. You have decided to create a clustered B+ Tree on the age field. The tree has a fanout of 200 and a height of 3. Assume that you are on average returning 50,000 users per query. On average, how many I/O's are performed by such a query?

Counting IOs Exercises

Assume that there are 2 million users in your database. Page size is 16kB. Each user entry is 2kB in size. You are performing range queries based on a user's age. Answer the following questions:

a. You have decided to create a clustered B+ Tree on the age field. The tree has a fanout of 200 and a height of 3. Assume that you are on average returning 50,000 users per query. On average, how many I/O's are performed by such a query?

3 + (50,000 * 2 / 16) = 6,253

Counting IOs Exercises

b. Assume your B+ tree is unclustered. In the worst case, how many I/O's do you need now? Assume that you are still returning 50,000 users per query on average, and that an index entry is 3 times smaller than a user entry.

Counting IOs Exercises

b. Assume your B+ tree is unclustered. In the worst case, how many I/O's do you need now? Assume that you are still returning 50,000 users per query on average, and that an index entry is 3 times smaller than a user entry.

3 I/Os to descend non-leaf index pages

ceil(50,000 * 2/3 / 16) = 2084 I/Os to read leaf index pages

50,000 I/Os to read unordered data pages

So 3 + 2084 + 50,000 = 52,087 I/Os.