Week 14 Accessing and Indexing Data (2)

Assignment Solutions

1. Select all of the true statements regarding B+ Tree and ISAM.

- [ ] ISAM will automatically sort its overflow pages in order to minimize lookup time.
- [ ] ISAM, on average, requires fewer disk operations to visit a data page than a B+ tree if we consider that all of the ISAM data fits into a structure in which we do not have to access any overflow pages.
- [ ] B+ Tree is better in tables that grow at a very fast pace compared to an ISAM structure.
- [ ] Overflow pages in ISAM do not get deleted unless all records in the overflow pages are deleted.

Solution: BCD

Explanation:

Option A: is **INCORRECT** because ISAM does not sort overflow pages, and this makes lookup slow, especially if there are long chains of overflow pages. There are implementations in which you can sort overflow pages; however, these implementations will make inserts slower.

Option B: is **CORRECT** because ISAM trees are normally fully packed and appear fully optimized with the shortest possible height. In B+ trees, they are rarely fully packed and are usually only more than 50% full.

Option C: is **CORRECT** because tables that grow quickly may cause overflow in ISAM (for example, situations where there are ever-increasing keys). Though both data structures have the concept of overflow, B+ Trees perform automatic rebalancing. The lookup time for ISAM can become even worse in special cases where overflow pages are not deleted.

Option D: is **CORRECT** because the statement is true and overflow pages may cause very long overflow page chains, which quickly degrades the performance of ISAM– another reason why B+ Tree seems more attractive than ISAM.

2. Consider a B+ tree index with $n = 50$, where n is the maximum number of keys fitting in a block. The B+ tree indexes 100,000 records. What is the minimum number of nodes in the tree that we need to examine when searching for a record?

- ○ 2
- ○ 3
- ○ 4
- ○ 1

Solution: B

Explanation: To get the correct solution we have to use the following steps:

1) (Level 1) Read the root node.

2) (Level 2) Follow a child pointer from the root. A root node can have maximum n+1 = 51 children.

3) (Level 3) Follow a child pointer from a Level-2 node, each of which can again have maximum 51 children.

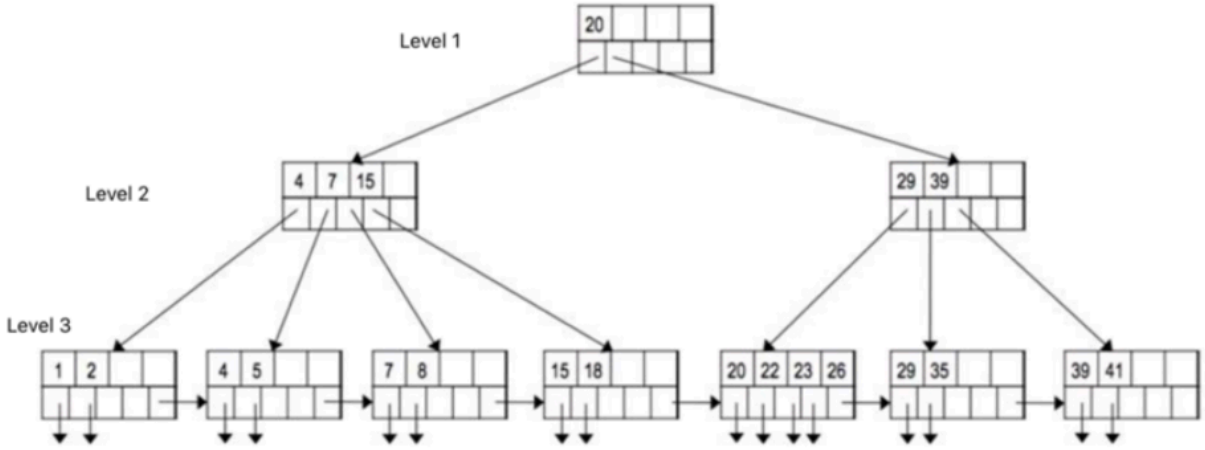
At this point, we have 51^2 leaf nodes. Each of these leaf nodes can have 50 pointers to records (the 51st pointer points to the block on its right), thus allowing it to index $51^2 \times 50 = 130050$ records. If you reduce the level by one, you can only index $51 \times 50 = 2550$ records, thus we need to have at least 3 levels in the B+tree. So, with 3 levels, we will need to read in 3 nodes.

3. Consider the following B+ tree in the figure. Each index node can hold at most 4 keys. A student from our CS411 class listed two solutions for inserting key 24 into the original tree:

Solution 1: Split the 5th leaf into [20, 22, 23] and [24, 26]. Add a new key 24 and a pointer to the rightmost node at level 2, which then has 3 keys and 4 pointers.

Solution 2: Split the 5th leaf into [20, 22] and [23, 24, 26]. Add a new key 23 and a pointer to the rightmost node at level 2, which then has 3 keys and 4 pointers.

Which of the above two solutions is correct?



- ◯ Both answers are correct
- ◯ Solution 1
- ◯ Both answers are incorrect
- ◯ Solution 2

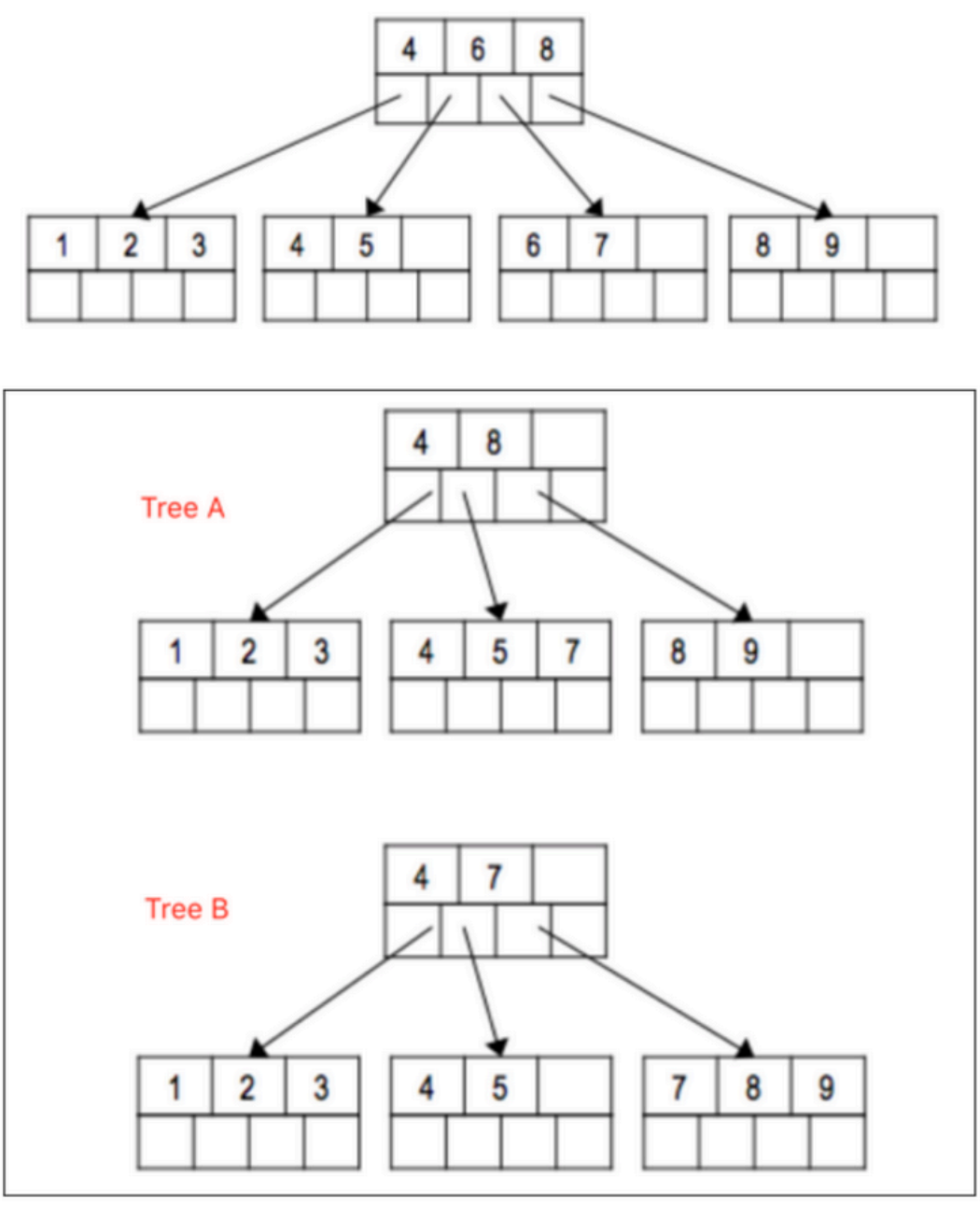
Solution: A

Explanation: A valid B+ tree needs to satisfy two conditions:
1) capacity requirement: each node has 2-4 keys (when degree is 2), and
2) key requirement: all keys correctly label the ranges of their subtrees.

Condition 1 is satisfied by both solutions.

Condition 2: In solution 1, with the new key 24, the ranges of the two children nodes are [20,24) and [24,29). In solution 2, with the new key 23, the range of the two nodes are [20,23) and [23,29). Both solutions satisfy this condition

4. Consider the following B+ tree in the figure, where each index node can hold 3 keys. What should be the resulting tree after deleting key 6 from the tree?



- ○ Tree B
- ○ Tree A
- ○ Neither tree is possible
- ○ Both trees are possible

Solution: D

Explanation: When we remove key 6 from the tree, key 7 becomes the only key in that node. However, for each node, we must have at least 2 keys, so we merge key 7 into another node. It does not matter which node we merge key 7 into, as long as the capacity and key/range requirements are satisfied, so both solutions will work.

5. Consider a B+ tree: The key size is 4 bytes, the pointer size is 4 bytes, and the degree d is 2. What is a possible **actual** size of data (i.e., keys and pointers) stored in a node in this tree?

- [x] 36
- [x] 28
- [ ] 22

Solution: AB

Explanation:

a) 36
Explanation: There can be at least d = 2 and at most 2d = 4 keys. If we have 4 keys then we have 4 + 1 = 5 pointers. Thus, since each key size is 4 bytes and each pointer size is 4 bytes, the total size becomes (4 * 4) + (5 * 4) = 36 bytes.

b) 28
Explanation: There can be at least d = 2 and at most 2d = 4 keys. If we have 3 keys, then we have 3 + 1 = 4 pointers. Thus, since each key size is 4 bytes and each pointer size is 4 bytes, the total size becomes (3 * 4) + (4 * 4) = 28 bytes.