461 Tutorial Week 2 Lab 2	B+-tree Exercise	Rosa KarimiAdl

Exercise 10.1 Consider the B+ tree index of order d = 2 shown in Figure 10.1.

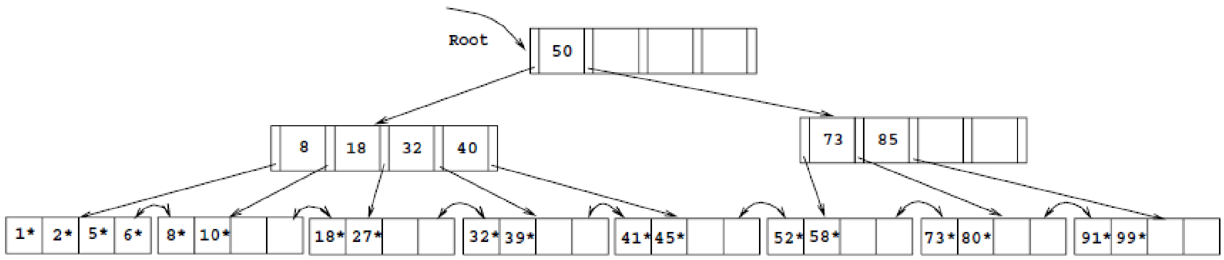
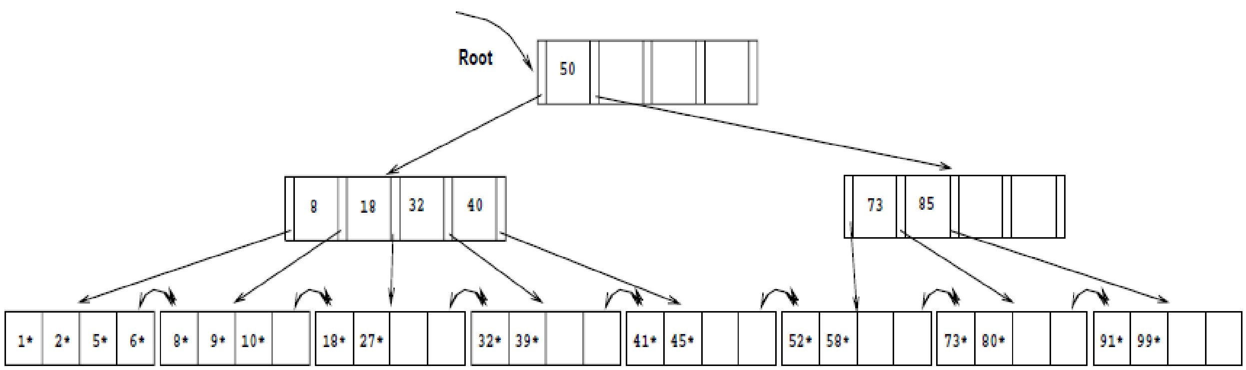


Figure 10.1 Tree for Exercise 10.1

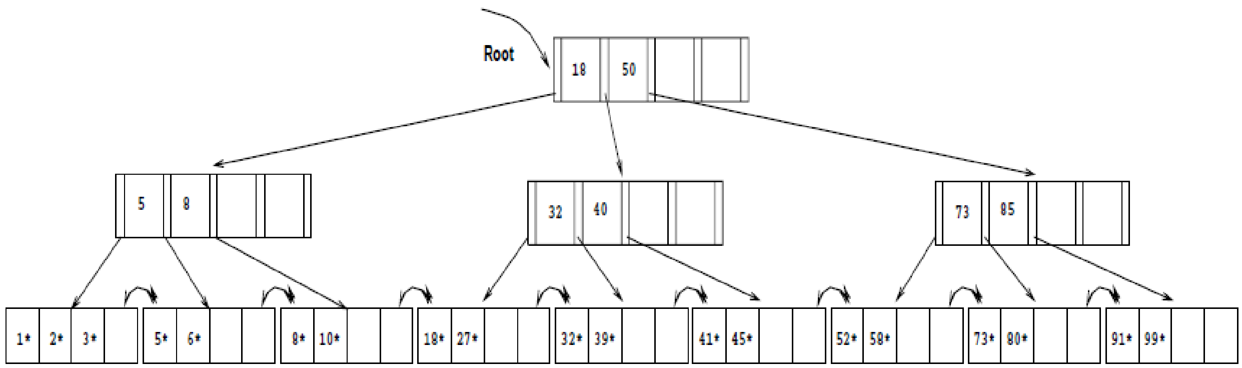
1. Show the tree that would result from inserting a data entry with key 9 into this tree.

Answer- The data entry with key 9 is inserted on the second leaf page:



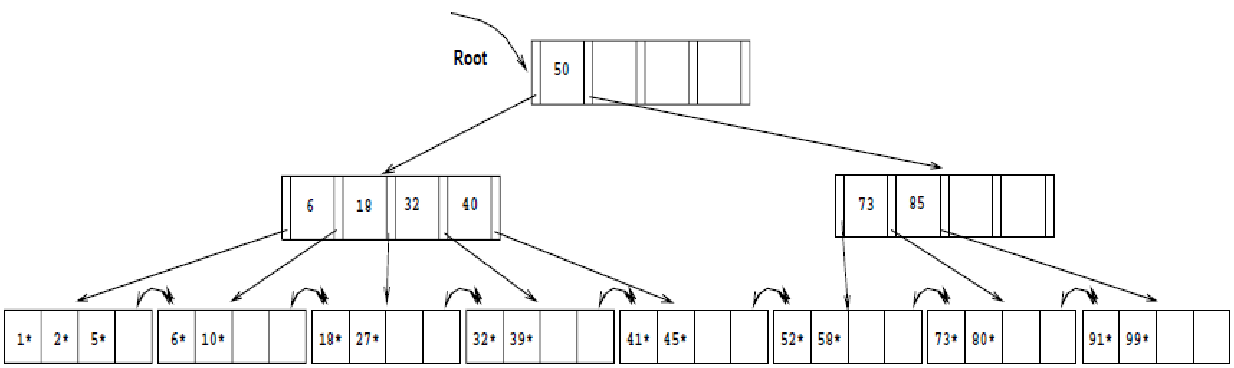
2. Show the B+ tree that would result from inserting a data entry with key 3 into the original tree. How many page reads and page writes does the insertion require?

Answer- The data entry with key 3 goes on the first leaf page F. Since F can accommodate at most four data entries (d = 2), F splits. The lowest data entry of the new leaf is given up to the ancestor, which also splits. Note how the middle key 18 is pushed up to the root node. The insertion will require 5 page writes, 4 page reads and allocation of 2 new pages.



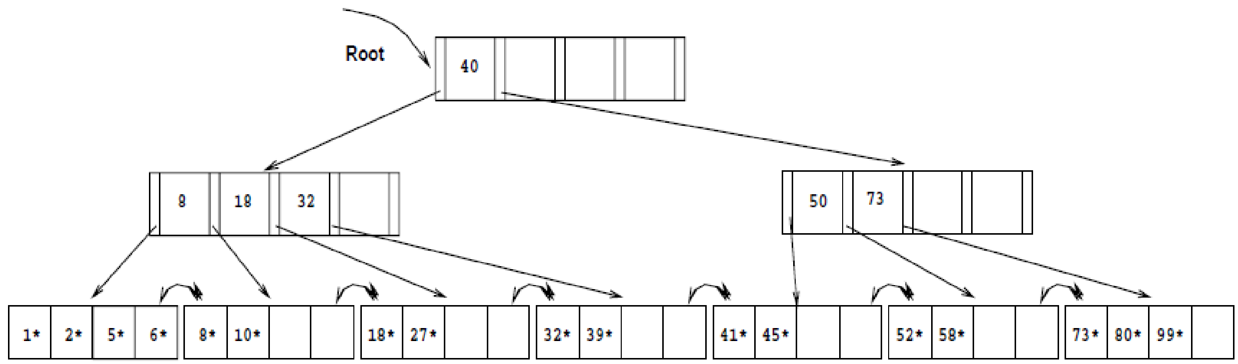
3. Show the B+ tree that would result from deleting the data entry with key 8 from the original tree, assuming that the left sibling is checked for possible redistribution.

Answer- The data entry with key 8 is deleted, resulting in a leaf page N with less than two data entries. The left sibling L is checked for redistribution. Since L has more than two data entries, the remaining keys are redistributed between L and N, resulting in the following tree:



6. Show the B+ tree that would result from deleting the data entry with key 91 from the original tree.

Answer- The data entry with key 91 is deleted from the leaf page N. N' s left sibling L is checked for redistribution, but L has the minimum number of keys. Therefore the two siblings merge. The key in the ancestor which distinguished between the newly merged leaves is deleted. This leaves us with an interior node with less than minimum entries. Consequently, it borrows one entry from its left sibling. This entry is 50, which has been pushed up to the root node. The last entry (40) in the first node is then pushed up to the root. The result can be seen in the following:



Exercise 10.3 Answer the following questions:

1. What is the minimum space utilization for a B+ tree index?

Answer- By the definition of a B+ tree, each index page, except for the root, has at least *d* and at most *2d* key entries. Therefore—with the exception of the root—the minimum space utilization guaranteed by a B+ tree index is 50 percent.

2. What is the minimum space utilization for an ISAM index?

Answer- The minimum space utilization by an ISAM index depends on the design of the index and the data distribution over the lifetime of ISAM index. Since an ISAM index is static, empty spaces in index pages are never filled (in contrast to a B+ tree index, which is a dynamic index). Therefore the space utilization of ISAM index pages is usually close to 100 percent by design. However, there is no guarantee for leaf pages' utilization.

3. If your database system supported both a static and a dynamic tree index (say, ISAM and B+ trees), would you ever consider using the static index in preference to the dynamic index?

Answer- A static index without overflow pages is faster than a dynamic index on inserts and deletes, since index pages are only read and never written. If the set of keys that will be inserted into the tree is known in advance, then it is possible to build a static index which reserves enough space for all possible future inserts. Also if the system goes periodically off line, static indices can be rebuilt and scaled to the current occupancy of the index. Infrequent or scheduled updates are flags for when to consider a static index structure.