Sample Exercises: B+-Trees

Exercise 1 Consider the B+ tree index of order d = 2 shown in Figure 10.1.

1. Show the tree that would result from inserting a data entry with key 9 into this tree.
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?
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.
4. Show the B+ tree that would result from deleting the data entry with key 8 from the original tree, assuming that the right sibling is checked for possible redistribution.
5. Show the B+ tree that would result from starting with the original tree, inserting a data entry with key 46 and then deleting the data entry with key 52.
6. Show the B+ tree that would result from deleting the data entry with key 91 from the original tree.
7. Show the B+ tree that would result from starting with the original tree, inserting a data entry with key 59, and then deleting the data entry with key 91.
8. Show the B+ tree that would result from successively deleting the data entries with keys 32, 39, 41, 45, and 73 from the original tree.

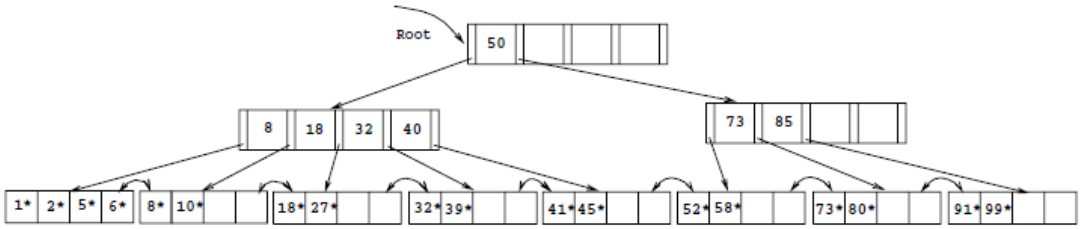


Figure 10.1 Tree for Exercise 10.1

Exercise 2 Answer the following questions:

1. What is the minimum space utilization for a B+ tree index?
2. If your database system supported both a static and a dynamic tree index, would you ever consider using the static index in preference to the dynamic index?

Exercise 3.

Consider a B+-tree of order two (d=2). Thus, the maximum number of pointers per node is 5 and the maximum number of entries is 4.

1. Show the results of entering one by one the keys that are three letter strings: (era, ban, bat, kin, day, log, rye, max, won, ace, ado, bug, cop, gas, let, fax) (in that order) to an initially empty B+-tree. Assume that you use lexicographic ordering to compare the strings. Show the state of the tree after every 4 insertions.
2. What is the utilization of the tree? The utilization of the tree is defined as the total number of entries in all the nodes of the tree (both leaf and non-leaf nodes) over the maximum number of entries that the same nodes can store.

Solutions

Answer 1

1. The data entry with key 9 is inserted on the second leaf page. The resulting tree is shown in figure 10.2.

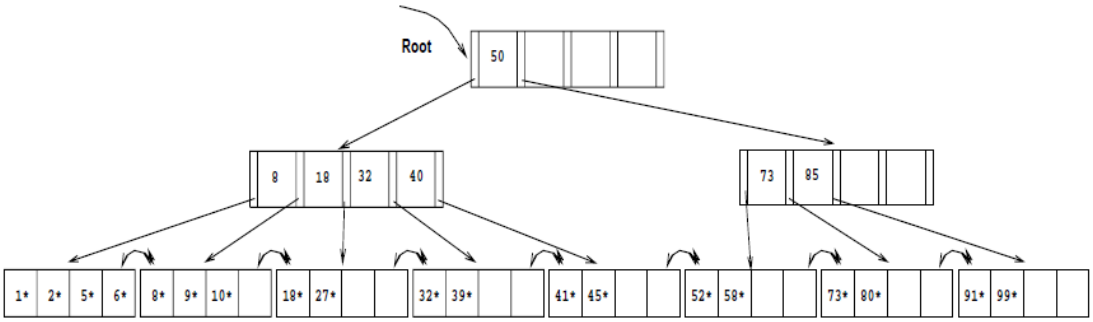


Figure 10.2

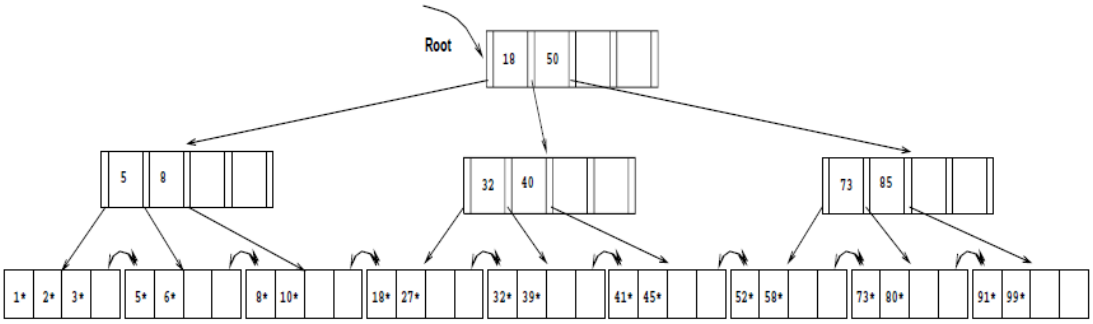


Figure 10.3

2. 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. The result can be seen in figure 10.3. The insertion will require 5 page writes, 4 page reads and allocation of 2 new pages.
3. 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 tree in figure 10.4.
4. As is part 3, the data entry with key 8 is deleted from the leaf page N. N's right sibling R is checked for redistribution, but R 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. The resulting tree is shown in figure 10.5.

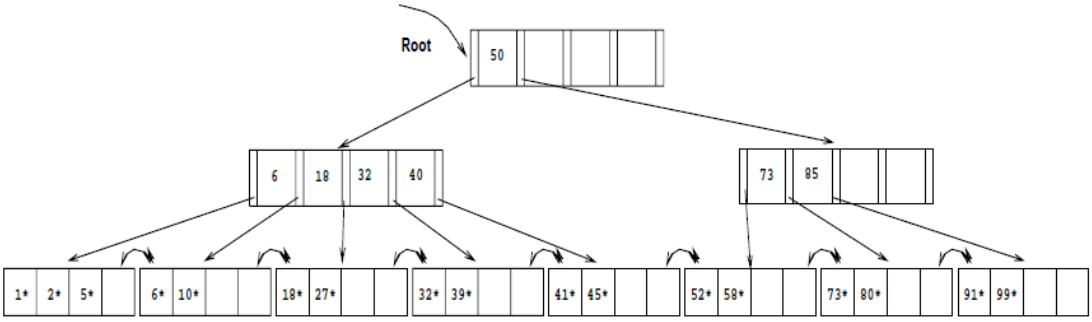


Figure 10.4

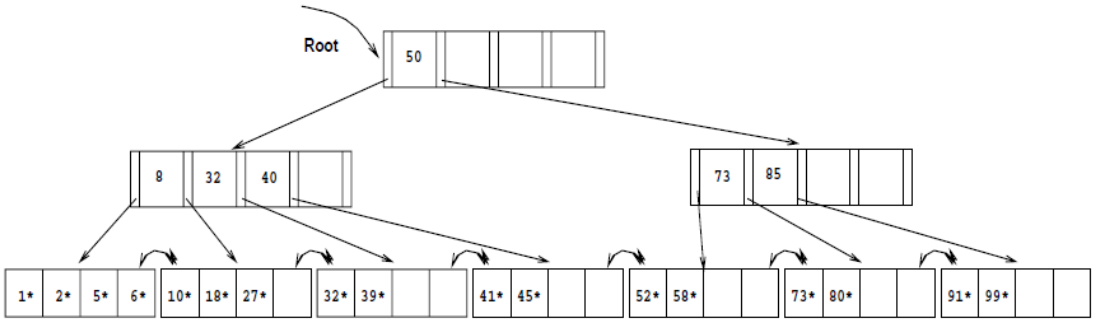


Figure 10.5

5. The data entry with key 46 can be inserted without any structural changes in the tree. But the removal of the data entry with key 52 causes its leaf page L to merge with a sibling (we chose the right sibling). This results in the removal of a key in the ancestor A of L and thereby lowering the number of keys on A below the minimum number of keys. Since the left sibling B of A has more than the minimum number of keys, redistribution between A and B takes place. The final tree is depicted in figure 10.6.
6. Deleting the data entry with key 91 causes a scenario similar to part 5. The result can be seen in figure 10.7.
7. The data entry with key 59 can be inserted without any structural changes in the tree. No sibling of the leaf page with the data entry with key 91 is affected by the insert. Therefore, deleting the data entry with key 91 changes the tree in a way very similar to part 6. The result is depicted in figure 10.8.
8. Considering checking the right sibling for possible merging first, the successive deletion of the data entries with keys 32, 39, 41, 45 and 73 results in the tree shown in figure 10.9.

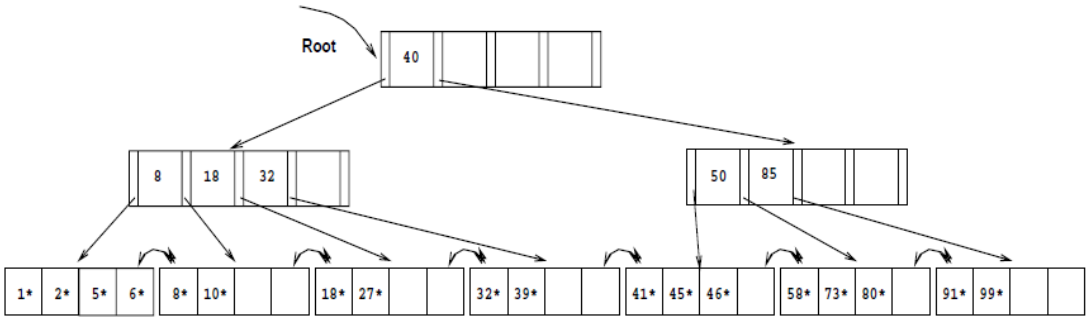


Figure 10.6

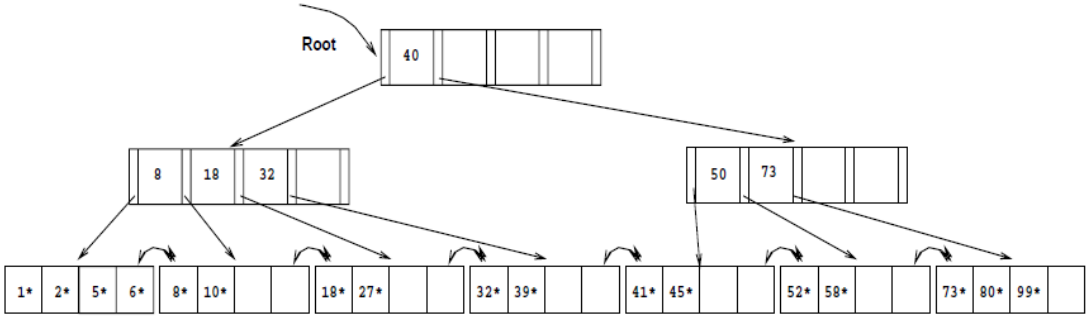


Figure 10.7

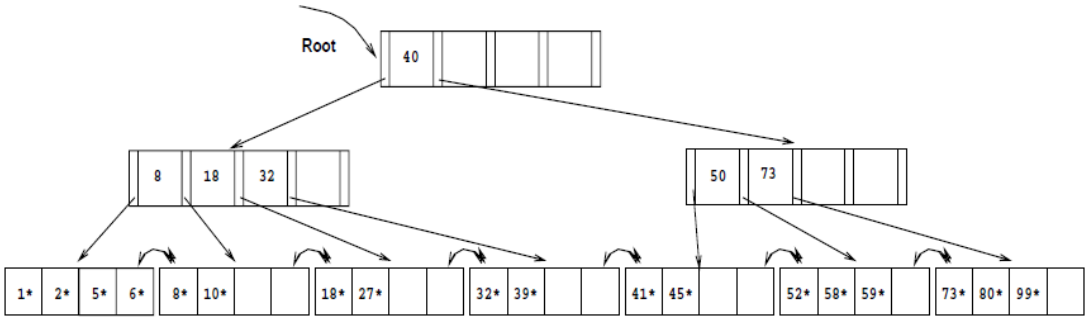


Figure 10.8

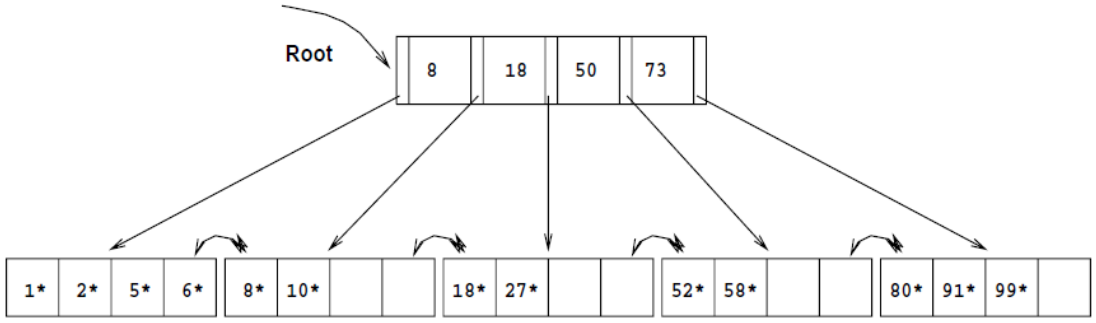


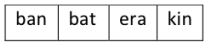
Figure 10.9

Answer 2 The answer to each question is given below.

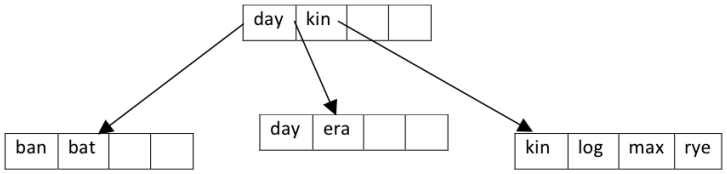
1. By the definition of a B+ tree, each index page, except for the root, has at least d and at most 2d key entries. Therefore—with the exception of the root—the minimum space utilization guaranteed by a B+ tree index is 50 percent.
2. A static index without overflow pages is faster than a dynamic index 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.

Answer 3

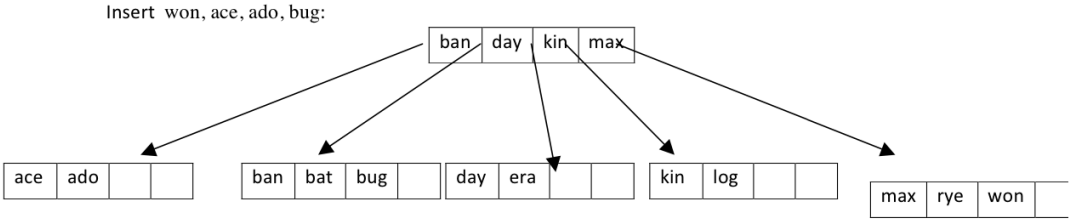
1. Insert era, ban, bat, kin:



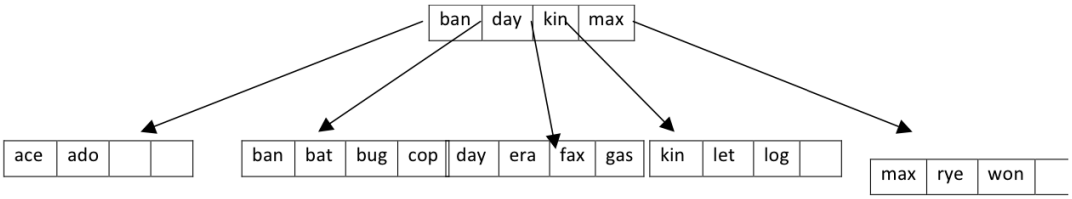
Insert day, log, rye, max:



Insert won, ace, ado, bug:



Insert cop, gas, let, fax:



2. The utilization of the tree is 20 / 24 = 0.833