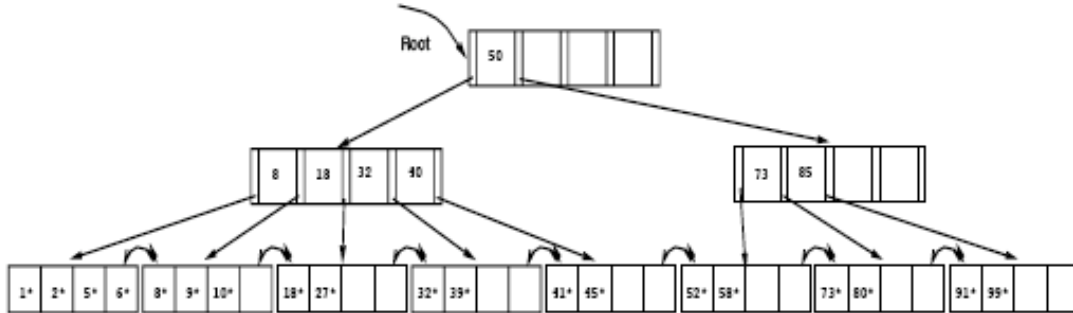


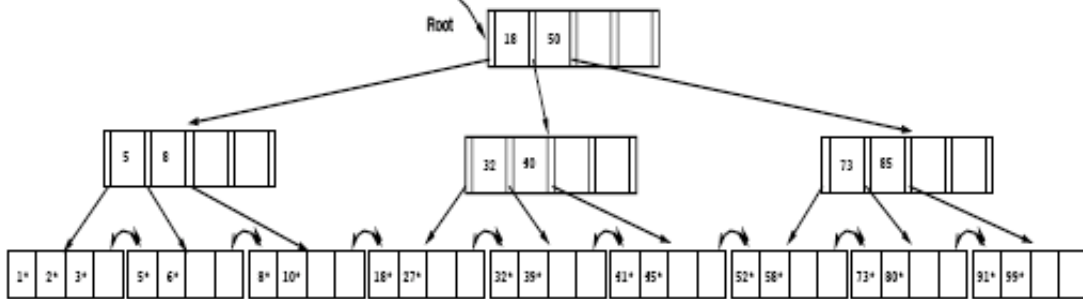
COP4710 Homework 4 solution

Problem I.

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

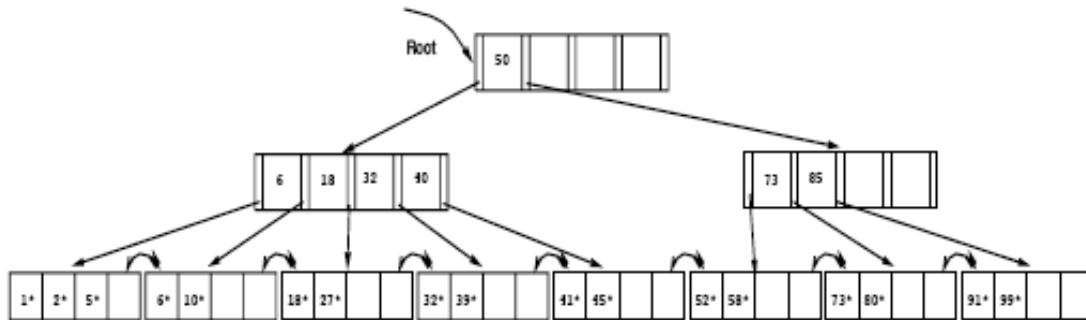


2. The data entry with key 3 goes on the first leaf page F . Since F can accommodate at most four data entries ($d = 2$), F splits. The lowest data entry of the new leaf is given up to the ancestor, which also splits. The result can be seen in the following figure.

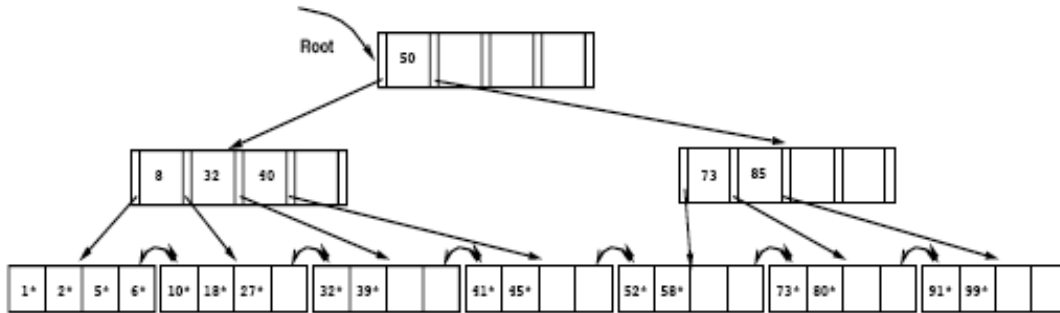


The insertion will require 5 page writes, 4 page reads and allocation of 2 new pages.

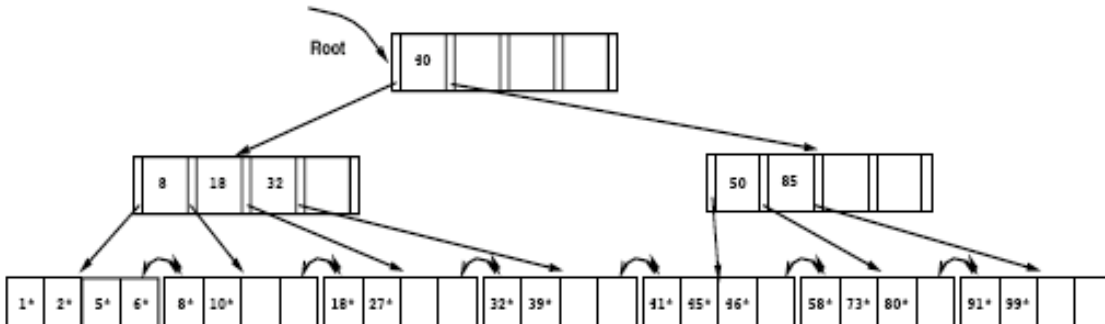
3. The data entry with key 8 is deleted, resulting in a leaf page N with less than two data entries. The left sibling L is checked for redistribution. Since L has more than two data entries, the remaining keys are redistributed between L and N , resulting in the following tree.



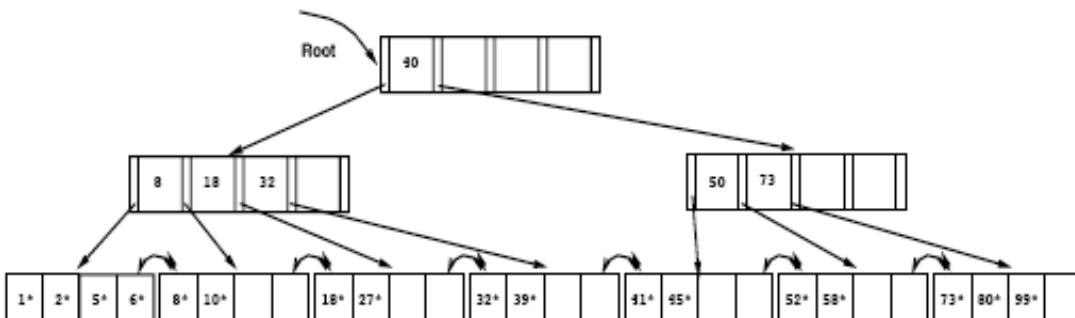
4. As is part 3, the data entry with key 8 is deleted from the leaf page N . N 's right sibling R is checked for redistribution, but R has the minimum number of keys. Therefore the two siblings merge. The key in the ancestor, which distinguished between the newly merged leaves, is deleted. The resulting tree is shown in the following figure.



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 the following figure.



6. Deleting the data entry with key 91 causes a scenario similar to part 5. The resulting tree is:



Problem II.

I will draw the tree without the pointers as follows:

Step 1. Insertion of 23, 65, 37, 60
 (23, 37, 60, 65)

Step 2. Insertion of 46, 92, 46 causes a split, note that there is a “copy over” of 46
 (46, , ,)
 (23, 37, ,) (46, 60, 65, 92)

Step 3. Insertion of 48, 71, 56, 59, 18, 21, split happens after insertion of 48

(46, 60, ,)
 (18, 21, 23, 37) (46, 48, 56, 59) (60, 65, 71, 92)

Step 4. Insertion of 10, split happens after insertion of 10

(21, 46, 60,)
 (10, 18, ,) (21, 23, 37,) (46, 48, 56, 59) (60, 65, 71, 92)

Step 5. Insertion of 74, 78, 15, 16, split happens after insertion of 74

(21, 46, 60, 71)
 (10, 15, 16, 18) (21, 23, 37,) (46, 48, 56, 59) (60, 65, ,) (71,74, 78, 92)

Step 6. Insertion of 20, 24, split happens after insertion of 20 and the split propagates to generate a new root with value 46. Note that 46 is push over to the root.

(46, , ,)
 (16, 21, ,) (60, 71, ,)
 (10, 15, ,) (16, 18, 20,) (21, 23, 24, 37) (46, 48, 56, 59) (60, 65, ,) (71,74, 78, 92)

Step 7. Insertion of 28, 39, split happens after insertion of 28

(46, , ,)
 (16, 21, 24,) (60, 71, ,)
 (10, 15, ,) (16,18,20,) (21,23, ,) (24,28,37,39) (46,48,56,59) (60,65, ,) (71,74,78,92)

Step 8. Insertion of 43, split happens after insertion of 43

(46, , ,)
 (16, 21, 24, 37) (60, 71, ,)
 (10, 15, ,) (16,18,20,) (21,23, ,) (24,28, ,) (37,39,43,) (46,48,56,59) (60,65, ,) (71,74,78,92)

Step 9. Insertion of 47, 50, 69, split happens after insertion of 47

(46, , ,)
 (16, 21, 24, 37) (48, 60, 71, ,)
 (10, 15, ,) (16,18,20,) (21,23, ,) (24,28, ,) (37,39,43,) (46,47, ,) (48,50,56,59) (60,65,69,)
 (71,74,78,92)

Step 10. Insertion of 75, 8, split happens after insertion of 75

(46, , ,)
 (16, 21, 24, 37) (48, 60, 71, 75)
 (8,10,15,) (16,18,20,) (21,23, ,) (24,28, ,) (37,39,43,) (46,47, ,) (48,50,56,59) (60,65, 69,) (71,74, ,)
 (75,78,92,)

Step 11. Insertion of 49, split happens after insertion of 49, split propagates and ends up pushing 60 over to the root.

(46, 60, ,)
 (16, 21, 24, 37) (48, 50, ,) (71, 75, ,)
 (8,10,15,) (16,18,20,) (21,23, ,) (24,28, ,) (37,39,43,) (46,47, ,) (48, 49, ,) (50,56,59,) (60,65, 69,)
 (71,74, ,) (75,78,92,)

Problem III.

Let $M = 1000$ be the number of pages in R , $N = 200$ be the number of pages in S , and $B = 52$ be the number of buffer pages available.

1. Basic idea is to read each page of the outer relation, and for each page scan the inner relation for matching tuples. Total cost would be

$$\#pagesinouter + (\#pagesinouter * \#pagesininner)$$

which is minimized by having the smaller relation be the outer relation.

$$TotalCost = N + (N * M) = 200,200$$

The minimum number of buffer pages for this cost is 3.

2. This time read the outer relation in *blocks*, and for each block scan the inner relation for matching tuples. So the outer relation is still read once, but the inner relation is scanned only once for each outer block, of which there are $\lceil \#pagesinouter / B - 2 \rceil = \lceil 200/50 \rceil = 4$.

$$TotalCost = N + M * \lceil N / (B - 2) \rceil = 4,200$$

If the number of buffer pages is less than 52, the number of scans of the inner would be more than 4 since $\lceil 200/49 \rceil$ is 5. The minimum number of buffer pages for this cost is therefore 52.

3. According to what we learned in class, the cost of sort-merge join with the sorting phase is

$$TotalCost = 3 * (M + N) = 3,600$$

The minimum number of buffer pages required is 25. With 25 buffer pages, the initial sorting pass will split R into 20 runs of size 50 and split S into 4 runs of size 50 (approximately). These 24 runs can then be merged in one pass, with one page left over to be used as an output buffer. With fewer than 25 buffer pages the number of runs produced by the first pass over both relations would exceed the number of available pages, making a one-pass merge impossible.