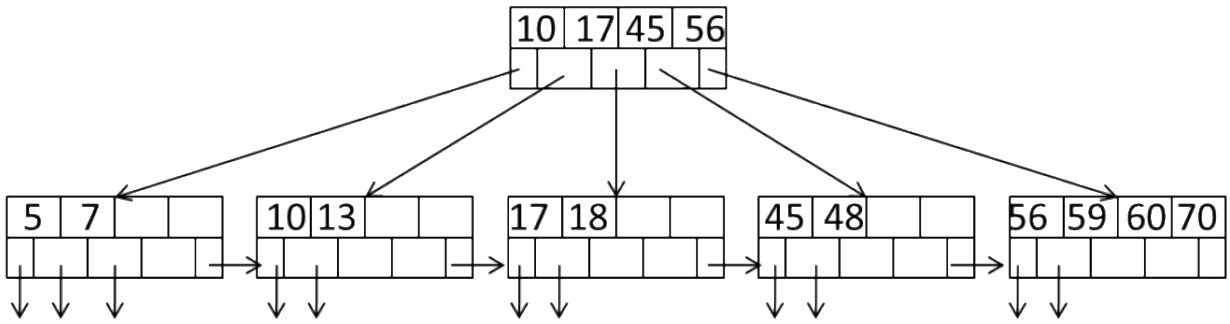
Homework #4

1. [40 points] Consider the following B+tree for the search key “age”. Suppose the degree d of the tree = 2, that is, each node (except for root) must have at least two keys and at most 4 keys.



a) Describe the process of finding keys for the query condition “age >= 20 and age != 45”. How many blocks I/O’s are needed for the process?

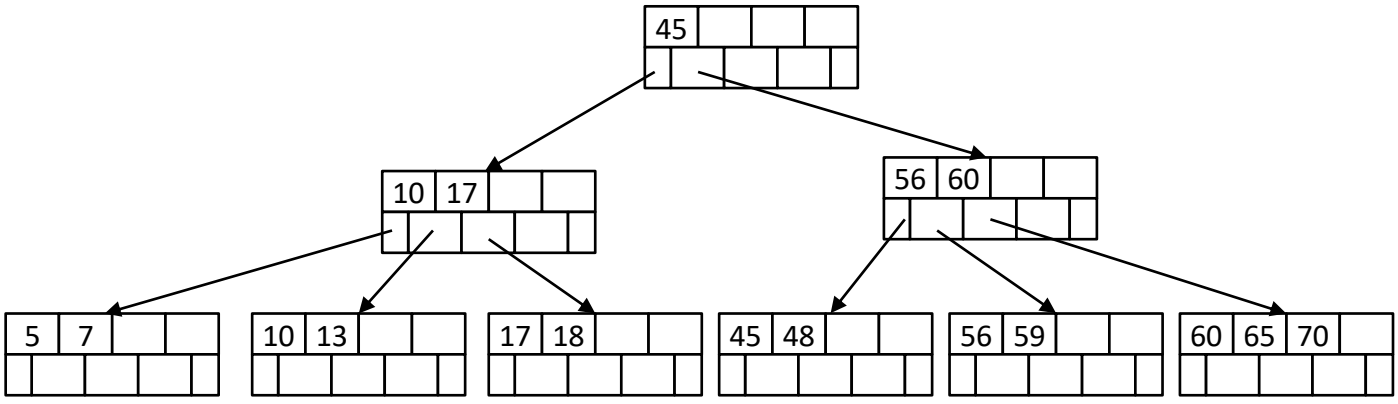
Answer:

- Start at root, try to find the leaf with 20. In this example, the corresponding leaf in the root node is the one down between 17 and 45 since 17 =< 20 <= 45. (1 block I/O)
- Then continue this procedure in the node pointed by the leaf, we find the leaf at the right-down of 18, but 18 < 20. (1 block I/O)
- So, we find the first leave in the next node who has the key = 45 which should be left out, so sequential traversal of the leaves after this one. (2 block I/O)

Totally 4 block I/O’s.

b) Draw the updated B+tree after inserting 65 into the tree.

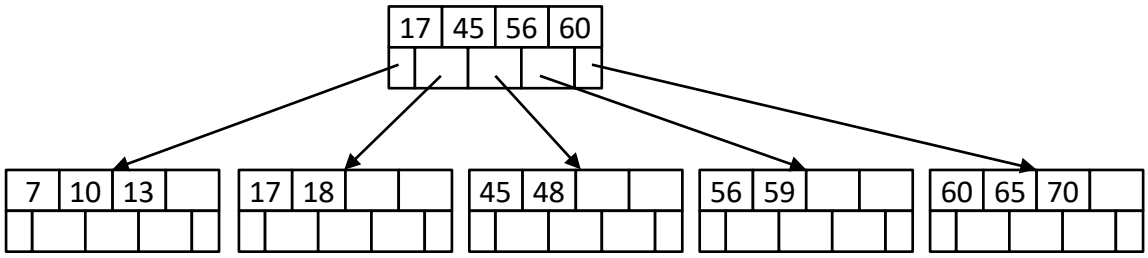
Answer:



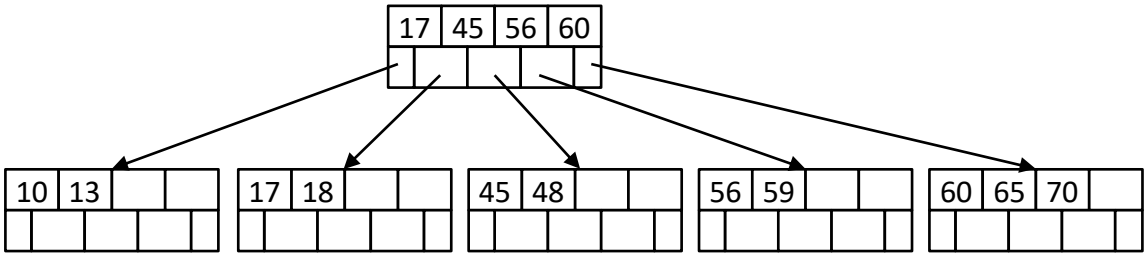
c) Draw the updated trees after deleting all odd ages from the leaf nodes of the tree obtained in part b. Assume the deletion goes from left to right. That is, first remove 5, then 7, 13, 17, 45, etc. Show the tree after each deletion.

Answer:

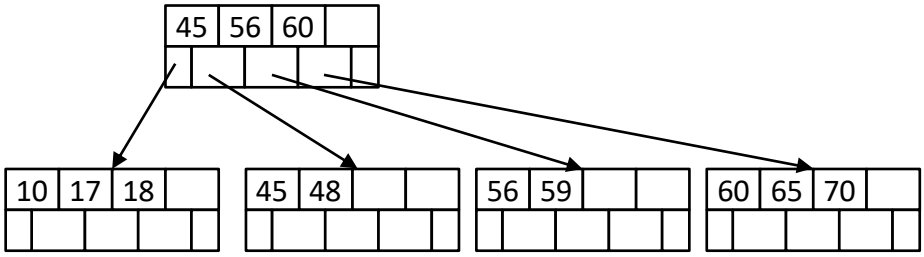
Step1: remove 5.



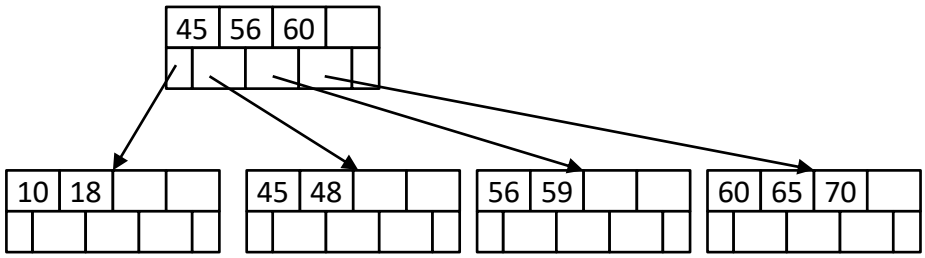
Step2: remove 7.



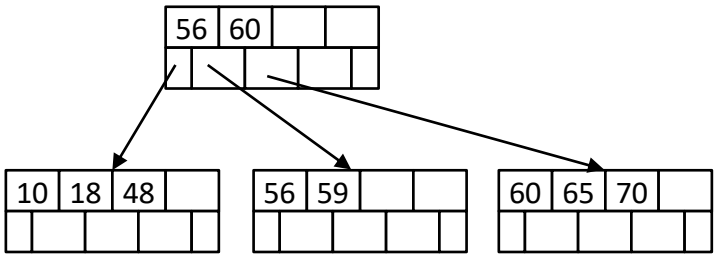
Step3: remove 13.



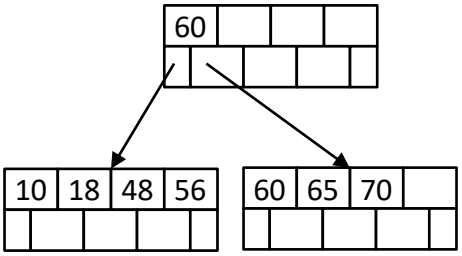
Step 4: remove 17.



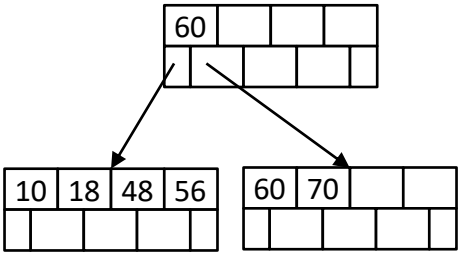
Step 5: remove 45.



Step 6: remove 59.



Step 7: remove 65.



2. [60 points] Consider natural-joining tables R(a, b) and S(a, c). Suppose we have the following scenario.
   i. R is a clustered relation with 12,000 blocks and 120,000 tuples
   ii. S is a clustered relation with 5,000 blocks and 100,000 tuples
   iii. has a clustered index on the join attribute a
   iv. V(S, a) = 100 (recall that V(S, a) is the number of distinct values of a in S)
   v. 100 pages available in main memory for the join
   vi. Assume the output of join is given to the next operator in the query execution plan (instead of writing to the disk) and hence the cost of writing the output is ignored.

   Describe the steps (including input, output at each step, and their sizes) in each of the following join algorithms. What is the total number of block I/O's needed for each algorithm? Which algorithm is most efficient?

   a. Nested-loop join with R as the outer relation

   Answer:

   Since R and S are clustered, do block-based nested loop join.
   i) Read (M -2 = 98) blocks of R into memory buffer, read 1 block of S into read buffer.
   ii) Traverse tuples in the read buffer and tuples in the block of R in the memory buffer.
   iii) Join tuples and output into write buffer block. If buffer is full, write the block into storage, clear the buffer and put record in the buffer.
   iv) Read another block of s into read buffer, do ii) and iii) again until all blocks of S are traversed.
   v) Read another 98 blocks of R into memory buffer, do ii), iii) and iv) again until all blocks of R are traversed.

   $$\text{Total block IO cost} = \mathrm{B(R)} + \left\lceil \frac{B(R)}{M-2} \right\rceil * B(S) = 12000 + \left\lceil \frac{12000}{100-2} \right\rceil * 5000 = 627000$$

b. Nested-loop join with S as the outer relation

Answer:

It is similar to the answer of a except R and S and inversed.

$$\text{Total block IO cost} = \mathrm{B(S)} + \left\lceil \frac{B(S)}{M-2} \right\rceil * B(R) = 5000 + \left\lceil \frac{5000}{100-2} \right\rceil * 12000 = 629000$$

c. Sort-merge join

Answer:

i) Read 100 blocks of R into memory at a time, sort all blocks of R into (12000/100 = 120) sorted runs and write to storage.

ii) Merge (M-1 = 99) runs of R into one sorted run and write to storage. Merge other (120-99 = 21) runs into one sorted run and write to storage.

iii) Read 100 blocks of S into memory at a time, sort all blocks of S into (5000/100 = 50) sorted runs and write to storage.

iv) Read blocks from runs with tuples having the same value on join attribute in sorted order. Do join for those tuples, write to write buffer. If write buffer is full, write buffer to storage then write join result to the buffer. When join on one join value is done, clear buffers with only those join value, read new blocks in order from the runs and join until runs are all traversed.

$$\text{Total block IO cost} = 5 * \mathrm{B(R)} + 3 * B(S) = 5 * 12000 + 3 * 5000 = 75000$$

d. Simple sort-based join

Answer:

It's the same with merge-sort join with step i) ~ iii).

iv) Merge two sorted runs of R into one sorted run.

v) Merge 50 sorted runs of R into one sorted run.

vi). Read 98 blocks from sorted run of S in order and one block from R. Join tuples with same join value, put it into write buffer and write to storage. Traverse throughout sorted runs of R and S, join tuples until the end of every run.

$$\text{Total block IO cost} = 7 * \mathrm{B(R)} + 5 * B(S) = 7 * 12000 + 5 * 5000 = 109000$$

e. Partitioned-hash join

Answer:

i) Use one block of memory as read buffer and (M-1 = 99) buckets as write buffer. Read a block from R, apply hash function h to its tuples' join value. Send it to the corresponding write buffer block. When buffer block is full, write it to storage. After, R should be hashed to 99 buckets with each average (12000/99 = 121) blocks.

ii) Hash S to 99 buckets with each average (5000/99 = 51) blocks with the same hash function h.

iii) For corresponding R_i and S_i with same hash value, if neither is empty, read the smaller one S and apply hash function h' to make join more efficient. Read the other relation R block by block and do the join. Write join result to write buffer block in memory then write to storage.

$$\text{Total block IO cost} = 3 * \mathrm{B(R)} + 3 * B(S) = 3 * 12000 + 3 * 5000 = 51000$$

f. Index join (ignore the cost of index lookup)

Answer:

i) Read R blocks into (M-2 = 98) blocks of memory each time, traverse every tuple.

ii) Use clustered index on S(a) to read blocks of S with same join value a block by block into memory and do the join. Send it to write buffer block in memory then write to storage.

$$\text{Total block IO cost} = \mathrm{B(R)} + T(R) * \frac{B(S)}{V(S,a)} = 12000 + 120000 * \frac{5000}{100} = 6012000$$

Judge from above IO cost, partitioned-hash join is the most efficient in this task.