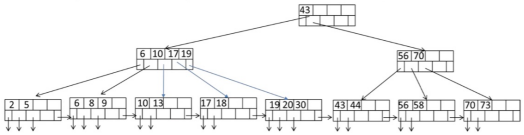
Homework 4

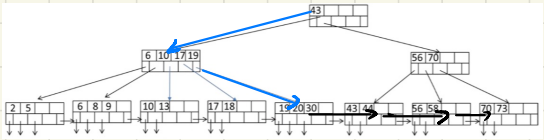
1. [40 points] Consider the following B+tree for the search key "age. Suppose the degree d of the tree = 2, that is, each node (except for root) must have at least two keys and at most 4 keys. Note that sibling nodes are nodes with the same parent.



   a. [10 points] Describe the process of finding keys for the query condition "age >= 35 and age <= 65". How many blocks I/O's are needed for the process?

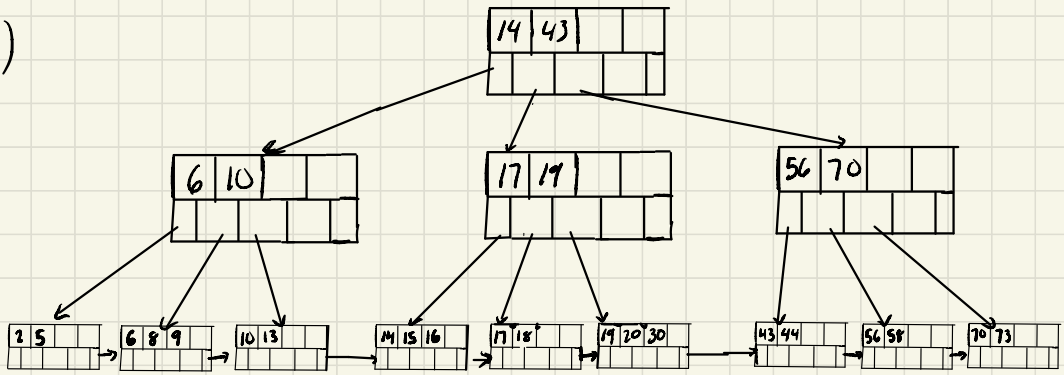
   b. [15 points] Draw the B+-tree after inserting 14, 15, and 16 into the tree. Only need to show the final tree after all insertions.

   c. [15 points] Draw the tree after deleting 2 from the original tree.

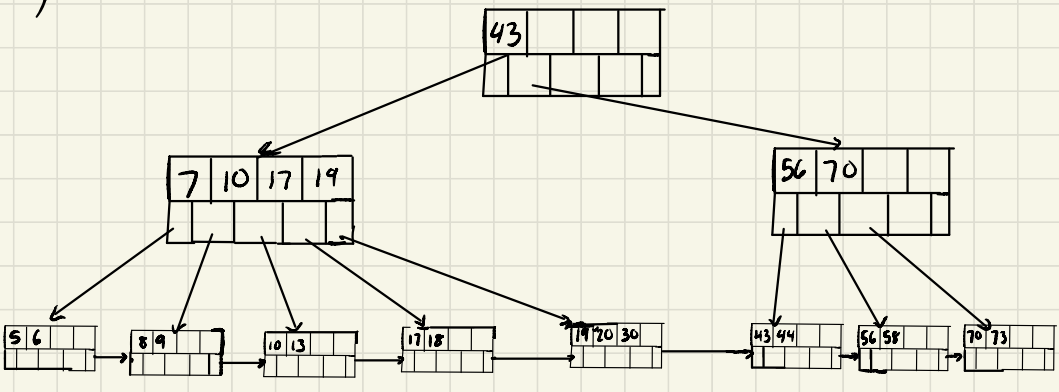


a) To find age >= 35 and age <= 65, you first will search for key equal to 35. We start from the root node and move to node with keys: 6, 10, 17, 19. Then we will move to the leaf node with keys 19, 20, 30. For searching for 35, we would need to access 3 nodes depicted in blue above. After we found 35, we move to the leaf node with keys 43 and 44 and also move to leaf node with keys 56 and 58. Since we want keys less than 65 we would need to load the next leaf node which begins with 70 Since this node begins with a key higher than 65, we know that we should stop with the node that contains the keys 56 and 58. Thus the total blocks I/O's needed for this process is 6 (3 from searching for 35 and 3 from searching for 65).

b)



c)



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 5,000 blocks.
   ii. S is a clustered relation with 20,000 blocks.
   iii. 102 pages available in main memory for the join.
   iv. Assume the output of join is given to the next operator in the query execution plan (instead of writing to the disk) and thus the cost of writing the output is ignored.

   Describe the steps for each of the following join algorithms. For sorting and hashing-based algorithms, also indicate the sizes of output from each step. What is the total number of block I/O's needed for each algorithm? Which algorithm is most efficient in terms of block's I/O?

a. [10 points] (Block-based) nested-loop join with R as the outer relation.

b. [10 points] (Block-based) nested-loop join with S as the outer relation.

c. [20 points] Sort-merge join (assume only 100 pages are used for sorting and 101 pages for merging). Note that if join can not be done by using only a single merging pass, runs from one or both relations need to be further merged, in order to reduce the number of runs. Select the relation with a larger number of runs for further merging first if both have too many runs.

d. [20 points] Partitioned-hash join (assume 101 pages used in partitioning of relations and no hash table is used to lookup in joining tuples).

a) Total Number of Block I/O's:

Router: $B(R) + B(R)/M-2 * B(S)$

$5000 + 5000/102-2 \cdot 20000 = \boxed{1,005,000}$

- First load 100 blocks of R (102-2). This is the outer loop.
- Next load input Buffer which is one block of S at a time, 20,000 times. This is the inner loop.
- The outer loop runs $B(R)/M-2$ times and each time the it needs to read S (20,000) to the output buffer. Once output buffer is full the next operator in the query begins
- Altogether R is read once so this is added to the product of the outer and inner loop

b) Total Number of Block I/O's:

Souter: $B(S) + B(S)/(M-2) * B(R)$

$20000 + 20000/102-2 \cdot 5000 = \boxed{1,020,000}$

- First load 100 blocks of S (102-2). This is the outer loop.
- Next load input Buffer which is one block of R at a time, 5,000 times. This is the inner loop.
- The outer loop runs $B(S)/M-2$ times and each time the it needs to read R (5,000) to the output buffer. Once output buffer is full the next operator in the query begins
- Altogether S is read once so this is added to the product of the outer and inner loop

c) Total Number of Block I/O's:

given: 100 pages for sorting

- Pass 1: sort R ⇒ 50 runs, 100 pages/run
  sort S ⇒ 200 runs, 100 pages/run

$2B(R) + 2B(S) = 50000$

given: 100 pages for merging

- Pass 2 (merge): $B(R) + B(S) \rightarrow 3B(R) + 3B(S) = 75{,}000$
- Need to remerge 200 runs → 2 runs → $B(R) + 2B(S)$

Total Number of Block I/O's: $3(5000) + 5(20000) = \boxed{115000}$

- Organize into two passes: pass 1: sort, pass 2: merge
- Sort R and S where there are 100 pages per run, 50 and 200 respectively
- Pass 1 cost is $2B(R) + 2B(S)$ because it is reading and writing the sorted data
- Pass two cost: $B(R) + B(S)$, add to pass 1 cost = $3B(R) + 3B(S) = 75000$
- Need to remerge: 200 runs → 2 runs
- Thus total number of Block I/O's $3B(R) + 5B(S) = 115000$

d) Total Number of Block I/O's:

given: 101 pages in partitioning of relations

- Pass 1: hash R into 100 buckets, 50 blocks/bucket
  hash S into 100 buckets, 200 blocks/bucket

$2B(R) + 2B(S) = 50000$

- Pass 2: join R_i with S_i → $B(R) + B(S) = 25000$

Total Number of Block I/O's: $3(5000) + 3(20000) = \boxed{75000}$

- Hash R into 100 (101-1) buckets
- Hash S into 100 (101-1) buckets
- Pass one cost is $2B(R) + 2B(S) = 50000$
- After this we join every pair of corresponding buckets via joining R_i with S_i
- Thus pass two cost is $B(R) + B(S) = 25000$
- Total # block I/O's $3B(R) + 3B(S) = 75000$

The partitioned-hash join is the most efficient algorithm out of all of them b/c it has the least number of block I/O's 75000.