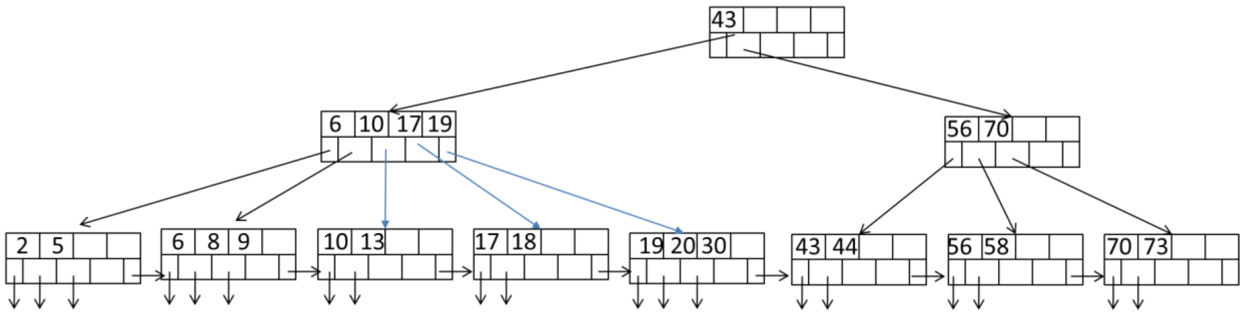
DSCI 551 – HW4

(Indexing and Query Execution)

Fall 2022 - Morning Sessions

100 points, Due Monday, 11/18 11:59 PM

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?

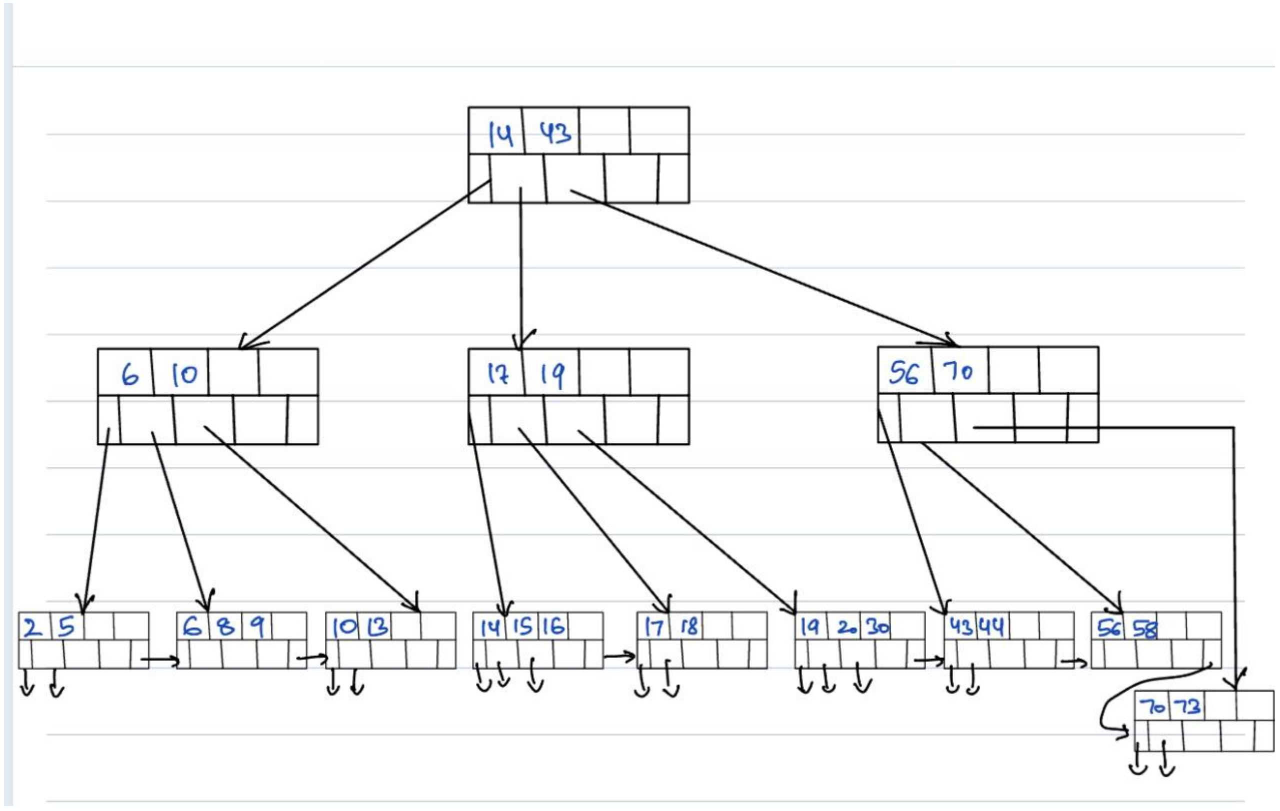
Total number on I/O blocks needed = 5

We start with the root node which has 43. Since we are finding the numbers between the range 35 and 65, we will traverse the left node first.

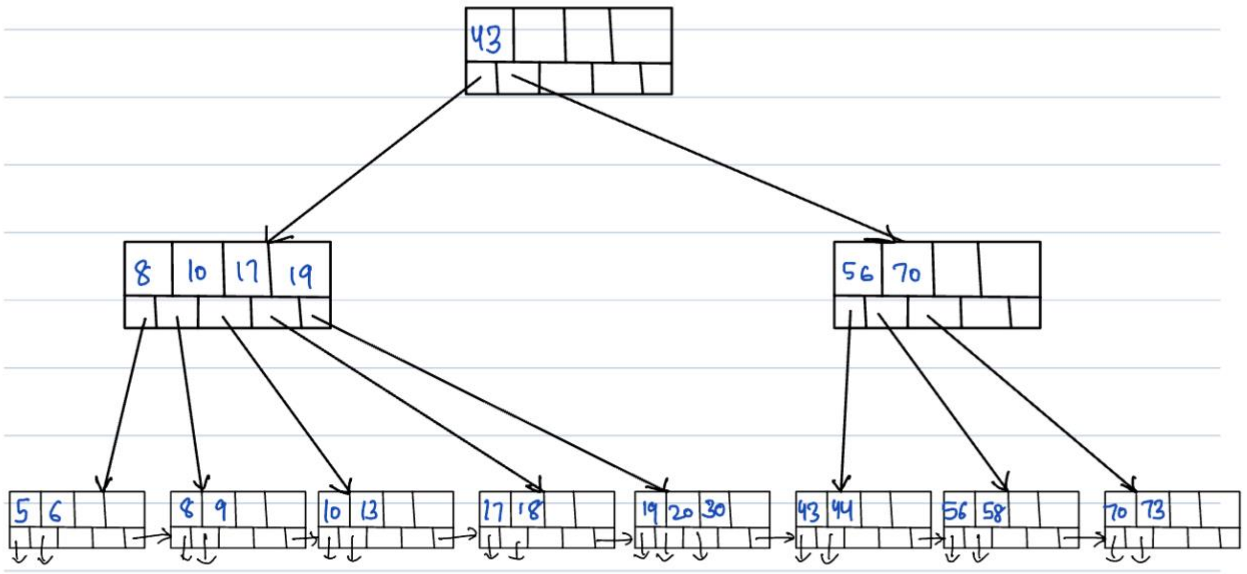
Traversing the left node, we need numbers greater than 35, so we read all the blocks greater than 19. So far, we have traversed the root node, its right node and all nodes greater than 19 making a total of 3 nodes.

Next, we move forward reading the nodes to the right and stop as soon as we get a value greater than 65. We stop right when we come across 70 making it a total of 5 blocks.

b. [15 points] Draw the B+-tree after inserting 6, 7, and 8 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.



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.
      Chunk size = 100 blocks
      Number of chunks from R = 5000/100 = 50
      If R is the outer relation, 1 pass for R and 50 passes for S.
      Total blocks needed = 5000 + 5000/(102-2) * 20000 = 1005000

   b. [10 points] (Block-based) nested-loop join with S as the outer relation.
      Chunk size = 100 blocks
      Number of chunks from S = 20000/100 = 200
      If R is the outer relation, 1 pass for S and 200 passes for R.
      Total blocks needed = 20000 + 20000/(102-2) * 5000 = 1020000

   c. [20 points] Sort-merge join (assume only 100 pages are used for sorting and 101 pages for merging). Note that if join cannot 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.
      B(R) = 5000
      B(S) = 20000
      Number of blocks used in sorting = 100
      Number of blocks used in merging = 101

      PASS 1 Sort
      Sort R => 50 runs (100 blocks/run)
      Sort S => 200 runs (100 blocks/run)
      Cost = 2 B(R) + 2 B(S)

      PASS 2 Merge:
      200 runs => 2 runs
      Cost = B(R) + 3 B(S)

      Total Cost = 3 B(R) +5 B(S) = 115000

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).

B(R) = 5000

B(S) = 20000

M = 102

PASS 1 Sort

Hashing R into 100 buckets = 5000/100 => 50 runs

Hashing S into 100 buckets => 20000/100 =>200 runs

Cost = 2 B(R) + 2 B(S)

PASS 2 Join

200 runs => 2 runs

Cost = B(R) + B(S)

Total cost = 3 B(R) + 3 B(S) = 75000

Partition-hash join is the most efficient algorithm in terms if block I/O.