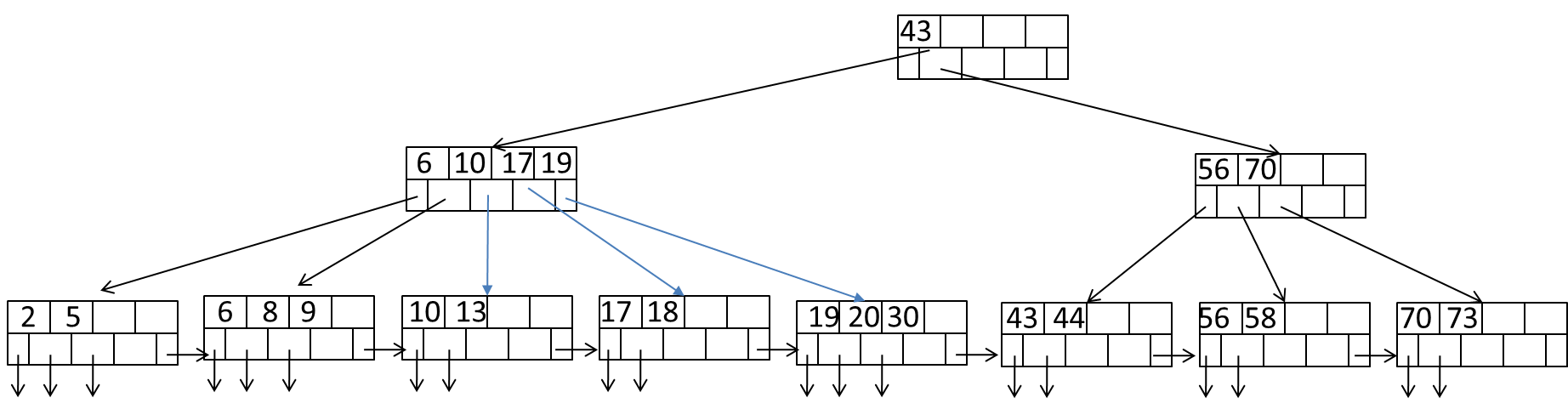
# DSCI 551 – HW4

# (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.**



1. [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?

Load root node, then go Left because 35 is less than 43, then load the fifth leaf from the left (19, 20, 30) because the next one is 43, so it must start somewhere between the two. Then load the next one (43, 44) since it is less than 65, then the next one as well (56, 58), then the next one (70, 73), because we need that last one to realize it is greater than 65 and to stop loading. 6 blocks total used.

1. [15 points] Draw the B+-tree after inserting ~~6, 7, and 8~~ 14, 15, and 16 into the tree. Only need to show the final tree after all insertions.

Diagram

Description automatically generated

1. [15 points] Draw the tree after deleting 2 from the original tree.

Diagram

Description automatically generated

1. [60 points] Consider natural-joining tables R(a, b) and S(a,c). Suppose we have the following scenario.
   * 1. R is a clustered relation with 5,000 blocks.
     2. S is a clustered relation with 20,000 blocks.
     3. 102 pages available in main memory for the join.
     4. 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?

* 1. [10 points] (Block-based) nested-loop join with R as the outer relation.

Cost1 = B(R)+B(R)/(M-2) \* B(S)

= 5000 + (5000/100 \* 20,000) = **1005000 Blocks**

Go through the R first as it is the outer loop. So because R is 5000 blocks, and 102 pages available, you end up with 5000/102 iterations of sorting R to do all the blocks. Then, for each iteration of R, you need to sort S by an input buffer of 1. So the inner loop will run B(R)/(M-2) iterations.

(See the slide for example)

* 1. [10 points] (Block-based) nested-loop join with S as the outer relation.

Cost2 = B(S)+B(S)/(M-2) \* B(R)

= 20,000 + (20,000/100 \* 5000) **= 1020000 Blocks**

Go through the S first as it is the outer loop. So because S is 20,000 blocks, and 102 pages available, you end up with 20,000/102 iterations of sorting S to sort all the blocks. Then, for each iteration of S, you need to sort R by an input buffer of 1. So the inner loop will run B(S)/(M-2) iterations.

* 1. [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.

So this is an External Sorting Problem

We sort R into B(R) / M number of runs, and Sort S into B(S) / M number of runs in the first pass. Then we merge them and they sort while merging. We have 101 blocks for merging, and with R/M = 50 and S/M = 200, we won’t have enough room to do it all in one merge, given that it’s by one buffer. So we do it again.

1.: B(R)/100 = 5000/100 = 50 runs

B(S)/100 = 20000/100=200

1. B(R)=50/100 = 0.5 runs

B(S)=200/100 = 2 runs

(2BR + 2BS for 1st round) + (2BS for 2nd round) + sort(1BR + 1BS)

3BR + 5BS

Pass1 sort and merge: 2br + 2bs + br + bs

3B(R) + 3 B(S)

* 1. [20 points] Partitioned-hash join (assume 101 pages used in partitioning of relations and no hash table is used to lookup in joining tuples).

As long as one is less than memory after doing BR/memory and BS/memory, then do one pass.

Because BR/memory = 50, it doesn’t matter that B(S) / 100 is greater than memory.

So total cost is 3BR + 3BS

Submit a pdf file: Lastname\_Firstname.pdf