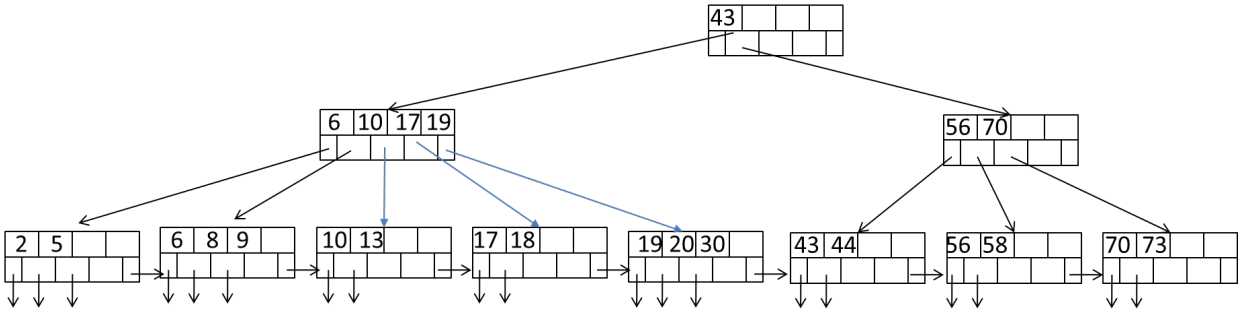
DSCI 551 – HW4

(Indexing and Query Execution)

(Fall 2020)

100 points, Due 4/18, Sunday

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. Describe the process of finding keys for the query condition "age >= 10 and age <= 20". How many blocks I/O's are needed for the process?
   b. Draw the B+tree after inserting 31 and 32 into the tree. Only need to show the final tree after the two insertions.
   c. Draw the tree after deleting 43 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 2,000 blocks and 10,000 tuples
   ii. S is a clustered relation with 50,000 blocks and 100,000 tuples
   iii. S has a clustered index on the join attribute a
   iv. V(S, a) = 5 (recall that V(S, a) is the number of distinct values of a in S
   v. 102 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 thus the cost of writing the output is ignored.

   Describe the steps (including input, output, and their sizes at each step, e.g., **sizes of runs or buckets**) for 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
   b. Sort-merge join (assume only 100 pages 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.

c. Partitioned-hash join (assume 101 pages used in partitioning of relations and no hash table used for lookup in joining buckets)
d. Index join (ignore the cost of index lookup)