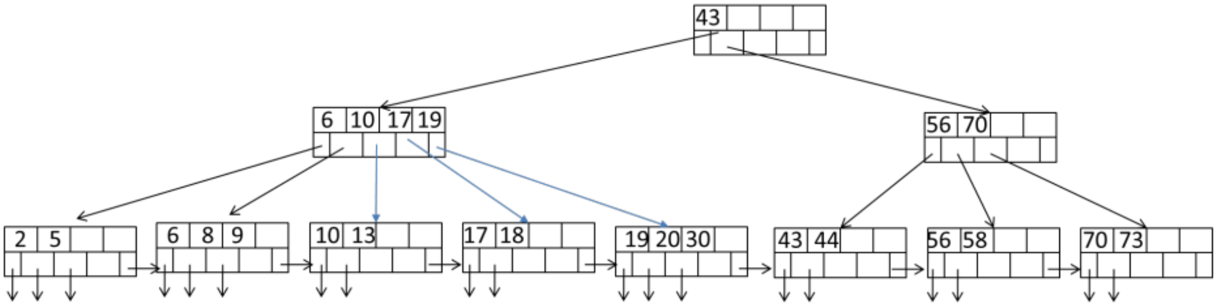
Lorenzo Scaturchio

Homework 4

1. 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 >= 35 and age <= 65”. How many blocks I/O’s are needed for the process?

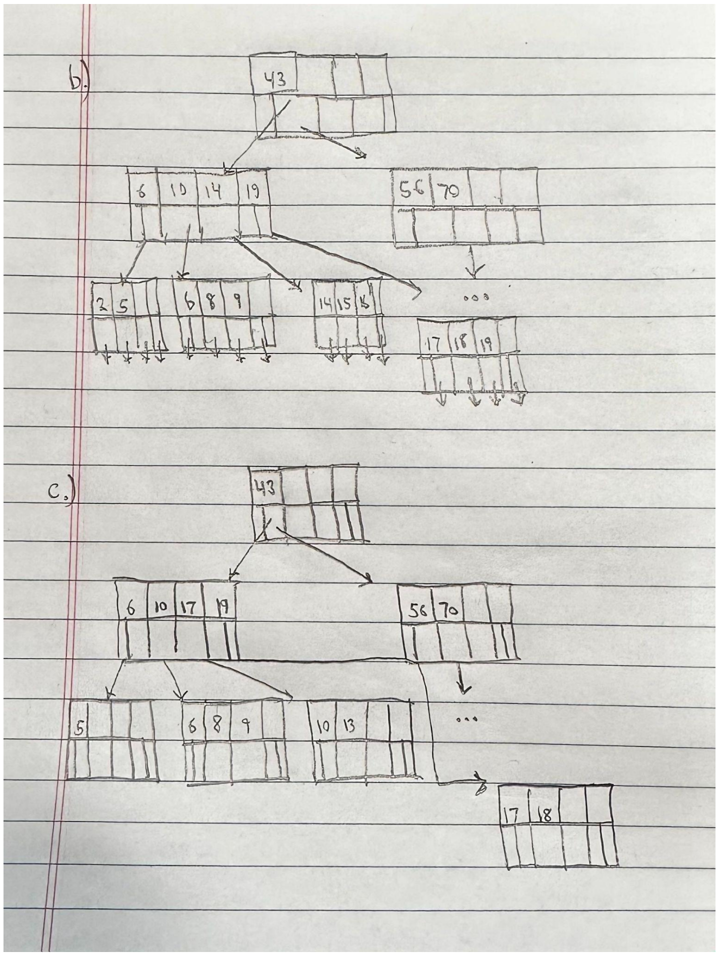
1. Read the root node of value 43, then find the second leaf node in this case on the left side.
2. Read the second leaf node on the left then find the start point in this case 43
3. Move the traversal up to 65 which would read the third and fourth leaf node
4. We stop at the fourth leaf node because the endpoint 58 is less than the fifth leaf node so we stop
5. It would cost 5 reads and 0 writes = 4 block I/O’s

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

- When insert 14: It cost 2 reads (root node and leaf node start with 10 = 3 block I/O’s
- When insert 15: It cost 2 reads (root node and leaf node start with 10 = block I/O’s
- When insert 16: It cost 2 reads (root node and leaf node start with 10 = … block I/O’s

   c. Draw the tree after deleting 2 from the original tree.

- When delete 2: It cost 4 reads (root node, internal node starts with… leaf node starts with… leaf node starts with…) and 2 write (create new leaf node with… and create new root node with…) = … block I/O’s



2. 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. (Block-based) nested-loop join with R as the outer relation.

```
for each 100 blocks br of R do
    for each block bs of S do
        if r and s join then output(r, s)
```

The outer loop runs 100 times and reads the costs multiplied by each other. Cost: B(R) + B(R)B(S) / 100 = 1,005,000 block I/O's

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

```
for each 100 blocks bs of S do
    for each block br of R do
        if r and s join then output(r, s)
```

The outer loop runs 100 times and reads the costs multiplied by each other. Cost: B(S) + B(S)B(R) / 100 = 1,020,000 block I/O's

c. 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

- Load 100 blocks of R each time, sort them and send them back to disk making 50 runs of size 100. Cost = 2 B(R) = 10000
- Load 100 blocks of S each time, sort them and send them back to disk making 200 runs of size 100. Cost = 2 B(S) = 40000
- Load 200 runs of R then merge them and send them back to disk which makes 2 runs of 10,000 pages. Cost = 2 B(R) = 10000
- Load 100 runs of S then merge them and send them back to disk which makes 1 run of 10,000 pages. Cost = 2 B(S) = 40000
- Mege 1 run from S and 2 runs from R by joining all sorted runs. Cost would be: B(R) + B(S) = 25000 Therefore total cost: 5 B(R) + 5 B(S) = 125,000 Block I/O's

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

- Hash R into 100 buckets and send them back to disk. Cost = 2 B(R) = 10000 200 blocks/bucket
- Hash S into 100 buckets and send them back to disk. Cost = 2 B(S) = 40000 100 blocks/bucket
- Join the buckets from R and S cost: B(S) + B(R) = 25000 Therefore, the total cost would be 3B(R) + 3B(S) = 75,000 Block I/O's