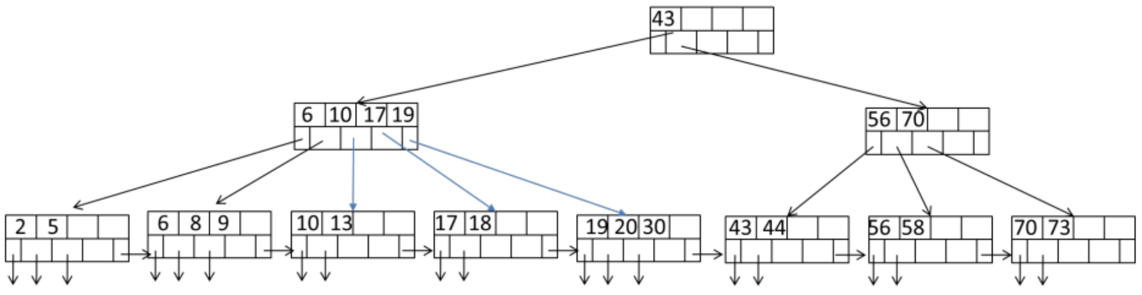
Homework 4

Question 1



A

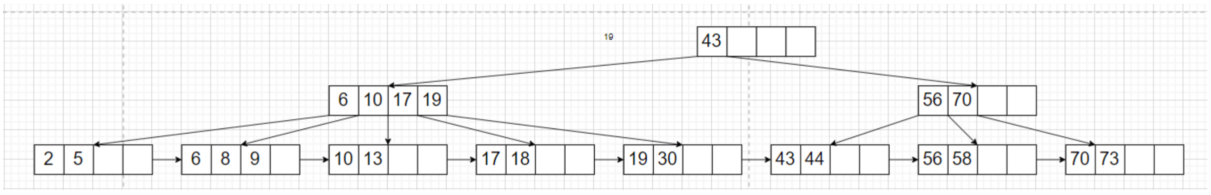
1. Start at the root node [43].
2. Since 15 is less than the smallest key in the root node (which is 19), we go to the left child [6, 10, 17, 19].
3. Since 50 is greater than the largest key in this node (which is 19), we continue to its right sibling [56, 70].
4. Since neither 15 nor 50 is in this node, we need to go down to its leaf nodes.
5. Traverse the leaf nodes to find all keys between 15 and 50 (inclusive).

Assuming each node in the tree fits in one disk block, the process requires reading **7** block I/O's

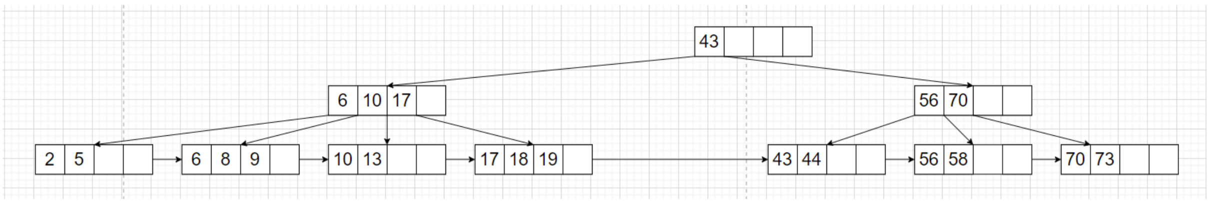
1. The root node [43]
2. Both index nodes [6, 10, 17, 19] and [56, 70]
3. The leaf node [10, 13], [17, 18], [19, 20, 30] & [43, 44]

B

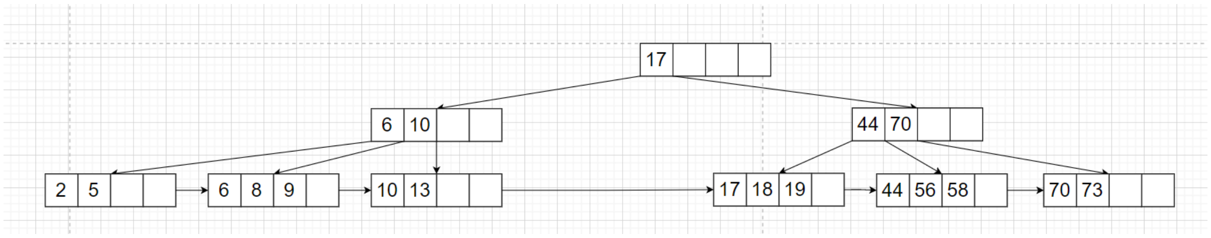
Deleting 20



Deleting 30

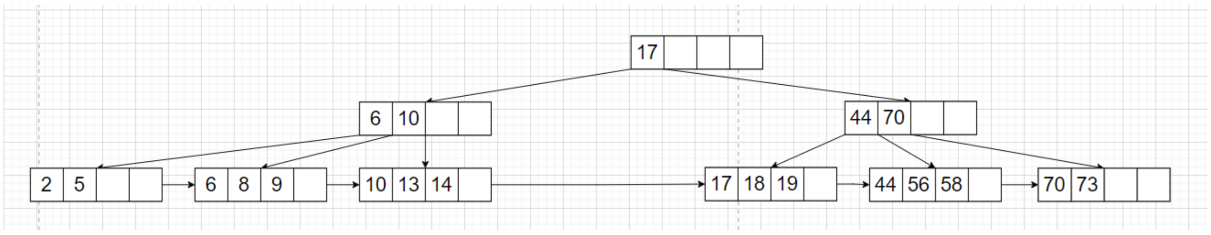


Deleting 43

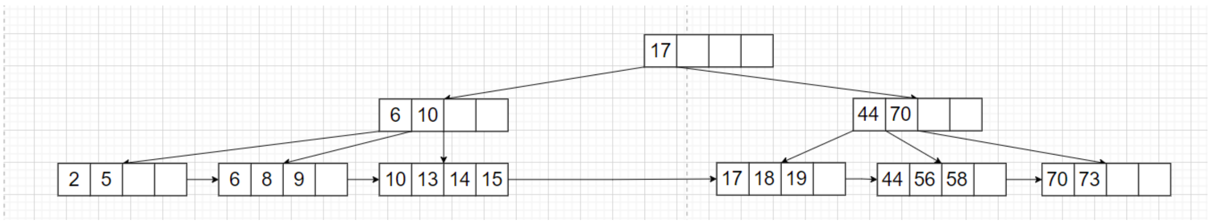


C

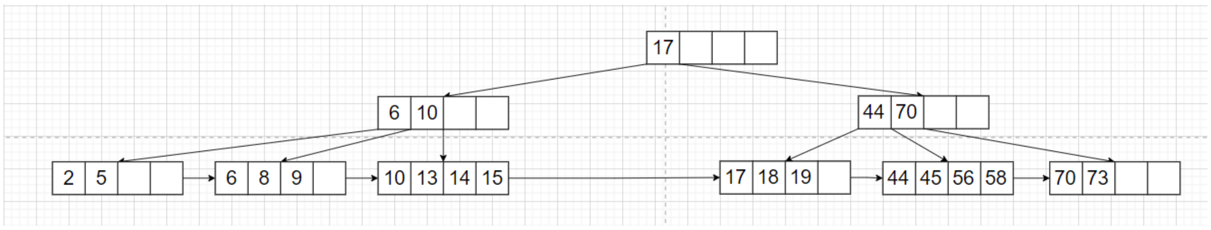
Inserting 14



Inserting 15



Inserting 45



Question 2

$M = Number\ of\ blocks/pages\ available\ in\ memory = 102$
$B(R) = Number\ of\ blocks\ holding\ R = 20000$
$B(S) = Number\ of\ blocks\ holding\ S = 50000$

Assumption: 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.

A

Block-based nested-loop joins with R as the outer relation:

```
for each (M-2) blocks $b_r$ of R do
    for each block $b_s$ of S do
        for each tuple r in $b_r$ do
            for each tuple s in $b_s$ do
                if r and s join then output(r,s)
                    ...
```

Cost Computation:

- Read R once = B(R)
- Outer loop runs over S = [B(R)*B(S)]/(M-2)
- Number of chunks of R = 200
- Chunk Size = 100
- R is the outer relation
  - 1 pass R
  - 200 passes through S
- Total Cost = B(R) + [B(R)*B(S)]/(M-2) = 20000 + (20000*50000)/100 = **10,020,000** blocks

B

Block-based nested-loop join with S as the outer relation:

```
for each (M-2) blocks $b_s$ of S do
  for each block $b_r$ of R do
    for each tuple s in $b_s$ do
      for each tuple r in $b_r$ do
        if r and s join then output(r,s)
          ...
```

Cost Computation:

- Read S once = B(S)
- Outer loop runs over R = [B(R)*B(S)]/(M-2)
- Number of chunks of R = 500
- Chunk Size = 100
- S is the outer relation
  - 1 pass S
  - 500 passes through R
- Total Cost = B(S) + [B(R)*B(S)]/(M-2) = 50000 + (20000*50000)/100 = **10,050,000** blocks

C

Sort-merge join:

Assumption:

- 100 pages are used for sorting.
- 101 pages for merging.

Pass 1 (Sorting):

- 200 runs with 100 blocks/run to sort R
- 500 runs with 100 blocks/run to sort S

Cost for read and write {R, S} = 2B(R) + 2B(S)

Since number of runs > M, an additional step will be required. Selecting relation with larger number of runs to reduce the number of runs.

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

Pass 2 (Merging):

Cost for merging {R, S} = B(R) + B(S)

Total Cost = 5B(R) + 5B(S) = 5 * (20000 + 50000) = **350,000** blocks

D

Partitioned-hash join:

Assumption: 101 pages used in partitioning of relations and no hash table is used to lookup in joining tuples

- Partition tuples in R and S using join attributes as key for hash.
- Tuples in partition Ri only match tuples in partition Si

Pass 1:

- Hash R into M-1 buckets
  - 100 buckets
  - 200 blocks/bucket

  Read and Write Cost = 2B(R)
- Hash S into M-1 buckets
  - 100 buckets
  - 500 blocks/bucket

  Read and Write Cost = 2B(S)
- Blocks in each bucket > M
  - We need to perform the partitioning algorithm again and hash each bucket into each consequent hash block before joining.

  Additional Cost for read and write {R, S} = 2B(R) + 2B(S)

  Final Cost for read and write {R, S} = 4B(R) + 4B(S)

Pass 2:

- Join R_i with S_i using Block based nested loop join method.

Merge Cost: B(R) + B(S)

Total Cost = 5B(R) + 5B(S) = 5* 20,000 + 5* 50,000 = **350,000** blocks

Partitioned-hash join algorithm and sort-based algorithm are equally efficient in terms of block's I/O.

- The hash join only needs to read each relation once to partition the data into buckets, and then read each bucket only once during the join phase.
- This can result in a significant reduction in the number of block I/Os compared to other algorithms like nested loop join.
- However, note the actual performance of each algorithm can depend on various factors such as the data distribution, available memory, and hardware specifications.