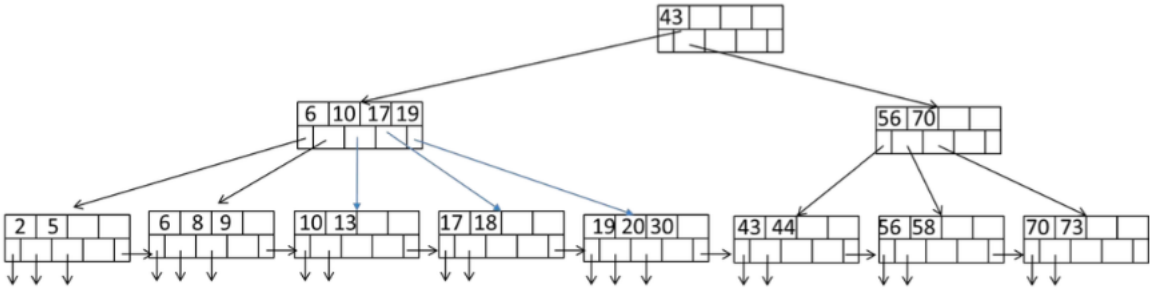
Shengtao Tony Hou
DSCI551
HW4
November 20th

Question 1:

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?

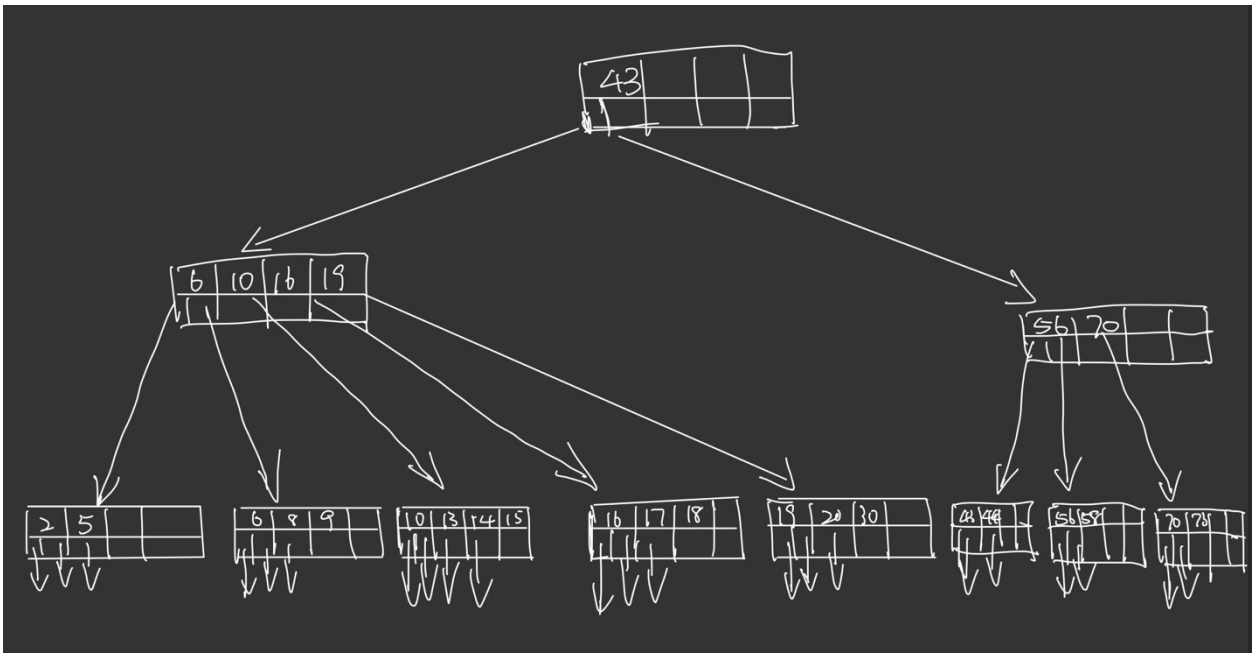
Answer :

Read the root and since 43 is >= 35 we move to left child internal node. We will then be moving to fifth leaf using the fifth pointer since 35 >=19. Since the largest value in fifth leaf is 30 which is less than 35, we will return to root and move down to the right child internal node to continue searching. Since 35 is less than 56, we will be moving to the first leaf node under. Because 35 <= 43 we know that 43 will be the first value within our query condition. We will then read 43,44 and use the pointer to go to second leaf node, where we read 56,58, since they are all less than 65. We then use the pointer to move to third leaf node under right child internal node and found ourselves value 70 which is larger than our upper bound of 65. Therefore the process of finding keys stop and we have our results as 43,44,56,58

Total blocks I/O needed are 6 blocks.

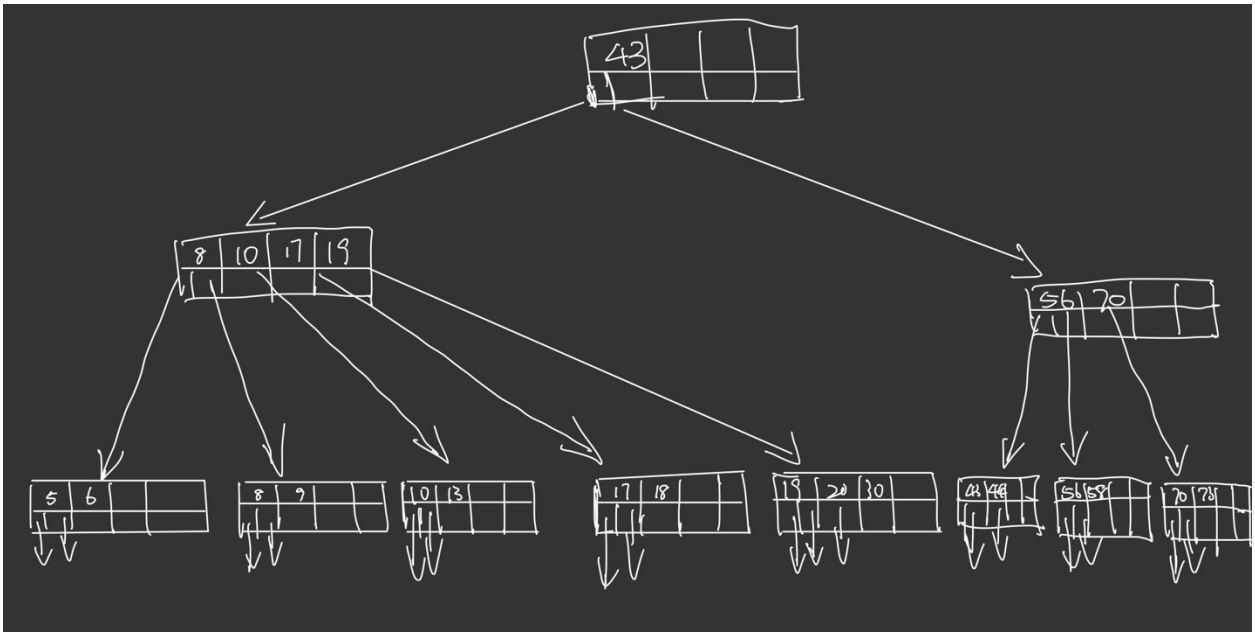
   b. [15 points] Draw the B+-tree after inserting 14, 15, and 16 into the tree. Only need to show the final tree after all insertions.

Answer:



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

Answer:



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.

$R \bowtie S$

for each 100 blocks of b_r of R do

for each block of 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)

- R as outer relation = 1 pass R, 50 passes through S

$$\frac{5000 b_r}{(102-2m)} = 50 \text{ passes through S}$$

$$\text{Cost: } R + \frac{(R*S)}{(M-2)} = 5000 b_r + \frac{5000 b_r}{(102-2m)} \times 2000 b_s$$

$$= 1005000 \text{ blocks I/O}$$

b. $S \bowtie R$ block

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

- S as outer relation: one pass S, 200 passes through R.

$\therefore \frac{20000 bs}{(102-2m)} = 200$ passes through R

Cost: $20000 bs + \frac{20000 bs}{(102-2m)} \cdot 5000 br =$

≈ 20000 I/O

C. R ⋈ S sort Merge

- Pass 1: sort
  - Split R into Runs of size M, then split S into runs of size M.
  - Sort R: 50 runs @ 100 blocks/run
    = 5000 blocks
    ∴ cost of $2b_r$ = 10000 blocks
    and 100 runs.
  - Sort S = 200 runs @ 100 blocks/run
    = 20000 blocks
    ∴ Cost of $2b_s$ = 40000 blocks
    and 400 runs

Pass 2 : Merge

- Run of R > 100 and S > 100 therefore take larger number to begin merging due to too large of runs

S runs = 400

R runs = 200

Merge R & S

$$R = \frac{400 \text{ (br runs)}}{100} = 4 \text{ runs}$$

$$S = \frac{200 \text{ (br runs)}}{100} = 2 \text{ runs}$$

$$2br + 2bs = 500002 \text{ blocks}$$

∴ Total cost $5br + 5bs = 125000$ 2blocks

I/O

d) R M = [illegible] blocks

d) $R \bowtie S$ partitioned hash

Pass 1

- Hash R into 100 buckets.

  50 blocks per bucket (R_i)

  $2b_r$ = 10000 blocks

- Hash S into 100 buckets.

  200 blocks per bucket (S_i)

  $2b_s$ = 40000 blocks

Pass 2: Join R_i with S_i

$b_r + b_s$ = 25000 blocks

∴ Total cost = $3b_r + 3b_s$ = 75000 blocks

I/O