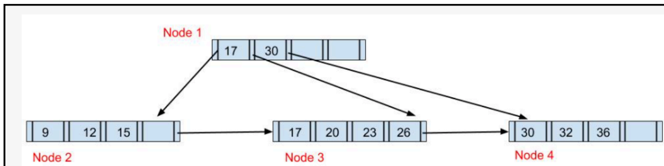
Reading Homeworks, cont……

HW 7

- Leaf
  - Copy $T.P_1$ through $T.K_{\lceil \frac{n}{2} \rceil}$ into L
  - Copy $T.P_{\lceil \frac{n}{2} \rceil + 1}$ ... into L'
  - K' = smallest key in L'
  - Insert K', L' into parent after L
- Interior
  - Copy $T.P_1$ through $T.P_{\lceil \frac{n+1}{2} \rceil}$ into P
  - $K'' = T.K_{\lceil \frac{n+1}{2} \rceil}$
  - Copy $T.P_{\lceil \frac{n+1}{2} \rceil + 1}$ ... into P'
  - Insert K'', P' into parent after P'



Q10.1
1 Point

Consider deleting the tuple with the key "30". After deleting the key from Node 4, how would Node 1 be affected (per the algorithm in the book)?

- ○ Delete "30" and the corresponding pointer from Node 1.
- ○ Replace "30" with "32" in Node 1.
- ◉ No change -- leave "30" as is in Node 1.

Q10.3
1 Point

Consider inserting a new key "22". What is the final set of keys in Node 1 (root)? As above, pick the answer that best fits.

- ○ 17, 30
- ○ 17, 22, 30
- ◉ 17, 23, 30
- ○ 17, 26, 30

HW 8

Q3
2 Points

Say we have a primary B+-tree of height 4 on attribute "zipcode" (not a candidate key) in a "person" table. So there will be many records with the same zipcode, but they are consecutive because the relation is sorted by zipcode.

Consider a query to find all people in a specific zipcode, and let's say there are 100 records with that zipcode. Further, let's say a single relation block can hold 10 records. Estimate the cost of executing this query. Assume t_S = 4ms, and t_T = 0.1ms.

How many milliseconds would this take? Enter nothing but the number, e.g., "49.3".

21.4

Explanation

This is primary, but not key, height 4. 10 blocks of tuples.

4 seeks + 4 blocks for tree = 16.4
1 seek + 10 blocks for tuples = 5

Q4
2 Points

Do the same but with the assumption that the index is a secondary index (i.e., the relation data is not sorted by zipcode). Assume the order of the B+-tree is 500, i.e, 500 search keys/pointers can fit on a single B+-tree node.

426.4

Explanation

Since only 100 ptrs and each leaf contains 500, assume all on one.

4 seeks + 4 blocks for tree = 16.4
100 * (1 seek + 1 block for tuples) = 100*4.1 = 410

HW 8

Q5
3 Points

Consider a query with a conjunctive predicate:

```
select * from R where a = 10 and b = 20.
```

- R occupies 1 million blocks on disk, and
- there are secondary indexes of height 4 on both R.a and R.b.
- Assume number of tuples in R with R.a = 10 is 1000, with R.b = 20 is 3000, and with both R.a = 10 & R.b = 20 is 200.

For all the indexes, assume the number of pointers in each leaf (to the actual records) is 500, and number of records of R per block is 100.

Q5.1
1 Point

How many blocks are transferred when using the index on R.a to fetch tuples matching R.a = 10, and then checking the condition in memory.

1005

Explanation

4 for tree (including leaf), 1 for extra leaf (1000 ptrs needed, 500 per leaf), 1000 for tuple blocks

Q5.2
1 Point

The same as the above but using the index on R.b:

3009

Explanation

4 for tree, 5 for extra leaves, 3000 for tuple blocks

Q5.3
1 Point

Same as above, but instead using "index-anding" to identify tuples w/o reading them, and then reading only those that match the entire predicate:

214

Explanation

4 for tree, 1 extra leaf for *a*, 4 for tree, 5 extra leaves for *b*, 200 tuple blocks

HW 8

Q6
2 Points

Consider a query with a disjunctive predicate:

```
select * from R where a = 10 OR b = 20
```

- R occupies 1 million blocks on disk
- secondary indexes of height 4 on both R.a and R.b
- 1000 tuples match R.a = 10, 1500 match R.b = 20, 2000 tuples match the entire predicate.

What are the total number of disk I/Os (i.e., # blocks transferred) for each of the following options?

For all the indexes, assume the number of pointers on the leaf level (to the actual records) is 500 per block, and number of records of R per block is 100.

Q6.1
1 Point

Specify the minimal number of disk I/Os (i.e., # blocks transferred) required to *identify* all matching tuples.

Explanation
4 for tree and 1 extra leaf for *a*, 4 for tree and 2 extra leaves for *b* = 11

11

Q6.2
1 Point

Using this approach (OR-ing the ptr sets), what would then be the total number of disk accesses to identify *and load* all matching tuples?

2011

Explanation
Just the above, plus 2000 blocks for the matching tuples = 2011

HW 9

Q4 Block nested loop join
2 Points

Compute the cost of a block nested-loop join between *R* and *S* in terms of seeks and block transfers.

- num blocks: $b_r = 500$, $b_s = 200$
- num tuples: $n_r = 2000$, $n_s = 100$
- R is the outer relation

Q4.1
1 Point

Assume M = 3.

How many blocks will be transferred?

100500

Explanation
Need to stream through S for each block of R. blocks= $b_r + b_r * b_s = 100500$

How many seeks will be required?

1000

Explanation
For each R block, need to seek to beginning of S, and seek to the correct R block. $2 * b_r = 1000$

Q4.2
1 Point

Assume M = 52. Seeks should be minimized.

How many blocks will be transferred?

2500

Explanation
Devote one block to S, one to output, and 50 to R. Stream through S for each chunk of R. blocks $b_r + (b_r/50) * b_s = 2500$

How many seeks will be required?

20

Explanation
seeks $2 * (br/50) = 20$

HW 9

Q5 External sorting
3 Points

Assume you need to sort 2000 tuples. 5 tuples fit into a block, and the system has 100 blocks of memory available. Count *all* reads and writes.

Q5.1
1 Point

How many runs are created?

4

Explanation
Runs are size of memory $(2000/5)/100 = 4$

How many blocks is each?

100

Explanation
500

Q5.2
1 Point

How many seeks for run creation?

8

Explanation
4 runs, 2 seeks each

How many blocks transfers for run creation?

800

Explanation
Each block has to be read once and written once, so 2*400=800

HW 9

Q5 External sorting
3 Points

Assume you need to sort 2000 tuples. 5 tuples fit into a block, and the system has 100 blocks of memory available. Count *all* reads and writes.

Q5.3
1 Point

How many merge steps (phases) are needed?

1

Explanation
Just one step. Only 4 runs, can easily have the minimum of 1 output block, and 1 input block for each run.

Q5.4
0 Points

Assume 60 blocks are used for the output buffer.

How many seeks for merging?

47

How many block transfers for merging? NOT ON TEST

800

Explanation
Each run will have 10 memory blocks for an input buffer: $(100-60)/4 = 10$. Each run is 100 blocks and will need to be filled $100/10 = 10$ times. So 40 seeks and 100 blocks for each run to be read in merging step.
The output buffer is 60 blocks, and so will need to be emptied $\lceil 400/60 \rceil = 7$ times. So 7 more seeks and 400 block transfers for the output buffer.
In total we need $4 * 10 + 60 = 47$ seeks, and $400 + 400 = 800$ block transfers.

HW 9

Q6
6 Points

Consider an equi-join on attribute B of relations R and S. Assume:

- b_r and b_s are 1000 and 2000, respectively
- we have an B+tree index on relation S on attribute B, of height 3.
- leaf nodes hold ptrs to 500 records
- B is a key in R, but not in S.
- 100 tuples in R each have 4 matches in S
- Each block of R or S holds 50 tuples.
- The below totals should reflect both identifying the matches, and returning the corresponding tuples.

Q6.1
1 Point

Compute blocks transferred assuming the index is a **primary**.

R blocks transferred:

1000

Explanation
Must read through entire relation R to find matches, no index.

Q6.2
1 Point

Compute blocks transferred assuming the index is a **primary**.

Index blocks:

150000

Explanation
1000 blocks * 50 tuples per block means 50,000 probes of S's index, each of which costs 3 block transfers.

Q6.3
1 Point

Compute blocks transferred assuming the index is a **primary**.

S blocks transferred:

100

Explanation
Need to read 400 tuples from S, though this is a primary, the entire 400 are not consecutive. Instead, the four tuples that match each specific tuple of the 100 that have a match, are on the same page.

For example, the tuples in R that have a (actually 4) matches are tuples 100, 200, etc., the 4 tuples of S that match B=100 are located on a single page. The 4 matching B=200 are on a single (but different than the previous) page. The 4 matching B=300 are on yet another page. So there are 100 distinct pages of S that have matches.

HW 9

- b_r and b_s are 1000 and 2000, respectively
- we have an B+tree index on relation S on attribute B, of height 3.
- leaf nodes hold ptrs to 500 records
- B is a key in R, but not in S.
- 100 tuples in R each have 4 matches in S
- Each block of R or S holds 50 tuples.
- The below totals should reflect both identifying the matches, and returning the corresponding tuples.

Q6.4
1 Point

Compute blocks transferred assuming the index is a **secondary**.

R blocks transferred:

1000

Explanation
no change

Q6.5
1 Point

Compute blocks transferred assuming the index is a **secondary**.

Index blocks transferred:

150000

Explanation
no change

Q6.6
1 Point

Compute blocks transferred assuming the index is a **secondary**.

S blocks transferred:

400

Explanation
secondary means that each tuple assumed on distinct page, so 400 pages

Exam #2

- Functional dependences (extraneous attributes, covers)
- ~~Storage manager~~
- ~~RAID~~
- File organization (heap, sorted, hash)
- Indexes (primary / secondary, dense sparse, hash)
  - B+-trees: height, cost of access, including xtra leaves
  - insertions, deletions
- Query execution (including costs)
  - selections
  - joins (block nested, hash, merge, index nested..)
  - sorts (in-memory, external)
- Query estimation
  - histograms
  - uniformity
  - using attribute stats
- Query optimization (general questions only)
  - execution trees
  - materialization/pipelining