CSE4701 - Databases
HW2 2020-03-04

Problem 1.
(35 pts) Consider a disk with block size B = 512 bytes. A block pointer is P = 6 bytes long, and a record pointer is PR = 7 bytes long. A file has r = 100,000 EMPLOYEE records of fixed length. Each record has the following fields:

NAME (25 bytes),
SSN (9 bytes),
DEPARTMENTCODE (9 bytes),
ADDRESS (40 bytes),
PHONE (9 bytes),
BIRTHDATE (8 bytes),
SEX (1 byte),
JOBCODE (5 bytes),
SALARY (4 bytes).

a. **Calculate the record size R in bytes.**

Record size R = (25 + 9 + 9 + 40 + 9 + 8 + 1 + 5 + 4) = 110 bytes

b. **Calculate the blocking factor bfr and the number of file blocks b, assuming an unspanned organization.**

Blocking factor, "bfr" = floor(B / R) = floor(512 / 110) = 4 records/block
# of file blocks, "b" = ceiling(r / bfr) = (100,000 / 4) = 25000 blocks needed for file

c. **Suppose that the file is ordered by the key field SSN and we want to construct a primary index on SSN. Calculate**
   i. **the index blocking factor *bfri***

Index record size, R_i = (SSN + P) = (9 + 7) = 16 bytes
Index blocking factor, bfr_i = floor(B / R_i) = floor(512 / 16) = 32

   ii. **the number of first-level index entries and the number of first-level index blocks**

# of 1st-level index entries, r_1 = # of file blocks, b = 25000 entries
# of 1st-level index blocks, b_1 = ceiling(r_1 / bfr_i) = ceiling(25000 / 32) = 782 blocks

   iii. **the number of levels needed if we make it into a multilevel index**

# of 2nd-level index entries, r_2 = # of 1st-level blocks, b_1 = 782 entries
# of 2nd-level index blocks, b_2 = ceiling(r_2 / bfr_i) = ceiling(782 / 32) = 25 blocks
# of 3rd-level index entries, r_3 = # of 2nd-level blocks, b_1 = 25 entries
# of 3rd-level index blocks, b_3 = ceiling(r_3 / bfr_i) = ceiling(25 / 32) = 1 blocks

iv. the total number of blocks required by the multilevel index, and

Total # of blocks for the index $b_i = b_1 + b_2 + b_3 = 782 + 25 + 1 = 808$ blocks

v. the number of block accesses needed to search for and retrieve a record from the file—given its SSN value—using the primary index.

# of block access to search a record = x + 1 = 3 + 1 = 4 block accesses

Problem 2. (15 pts) Given the same specifications of Problem 1, consider this time you are building a primary index on SSN using B-tree. Calculate ...

i. the order p for the B-tree, ...

$$(block\ pointer \cdot p) + (key\ pointer + block\ pointer) \cdot (p - 1) \leq 512$$
$$(6 \cdot p) + (9 + 6) \cdot (p - 1) \leq 512$$
$$6p + 15p - 15 \leq 512$$
$$21p \leq 527$$
$$p = 25$$

ii. the number of levels needed if blocks are approximately 69% full (round up for convenience), and…

$p \cdot 0.69 = 25 * 0.69 = 17.25 \approx 18$

- need 25,000 blocks

	# of nodes	# of pointers	# of key values
Root:	1 node	18 ptrs	17 entries
Level 1:	18 nodes	18 x 18 ptrs	18 x 17 entries
Level 2:	18 x 18 nodes	18 x 18 x 18 ptrs	18 x 18 x 17 entries
Level 3:	18 x 18 x 18 nodes		18 x 18 x 18 x 17 = 99,144 entries

- $log_{18}(25000) = 3.50357 \rightarrow$ 4 levels

iii. the worst-case number of blocks needed to search for and retrieve a record from the file—given its SSN value—using the B-tree you are estimating.

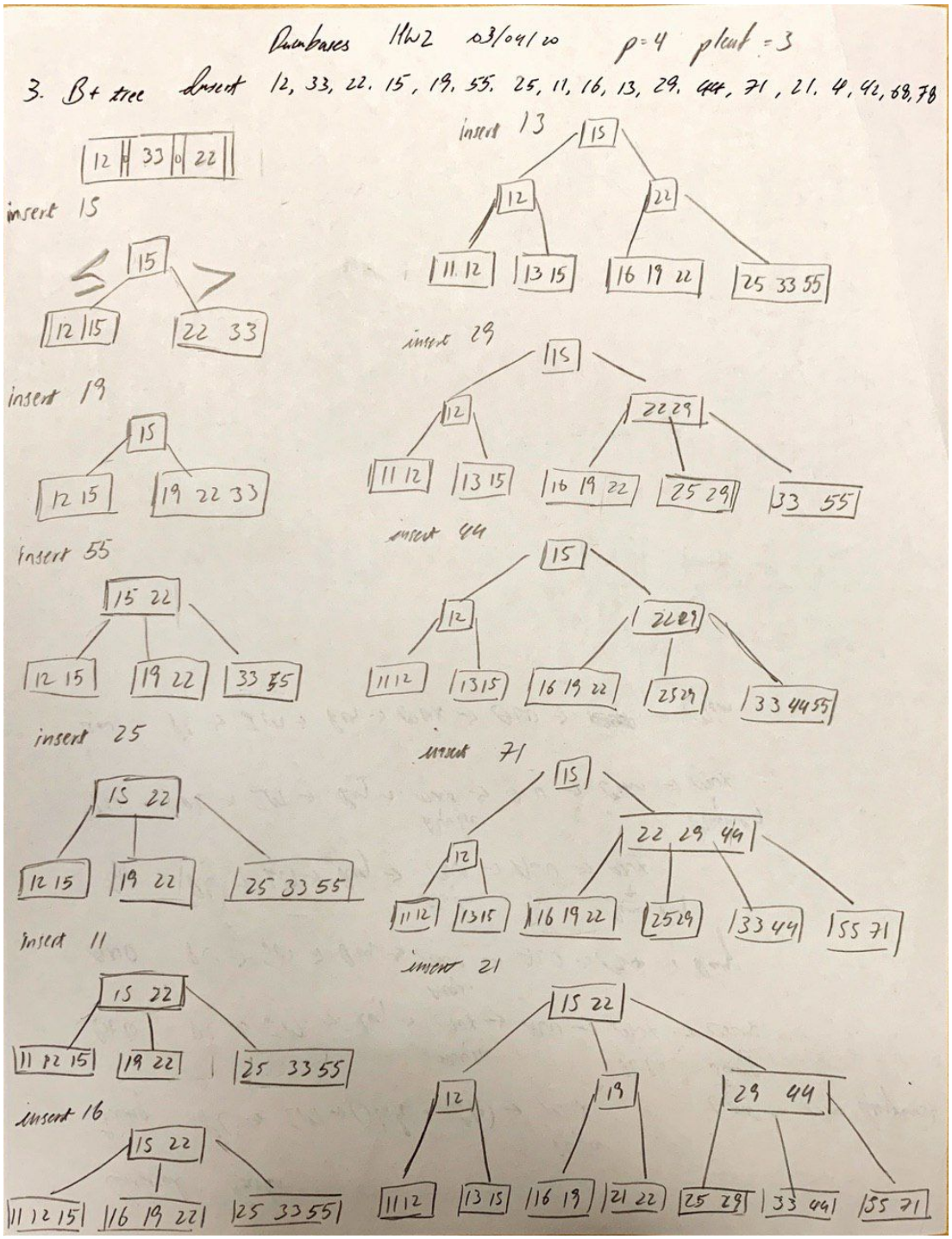
Worst-case block searches = # of levels + 1 = 4 levels + 1 = 5 blocks searches

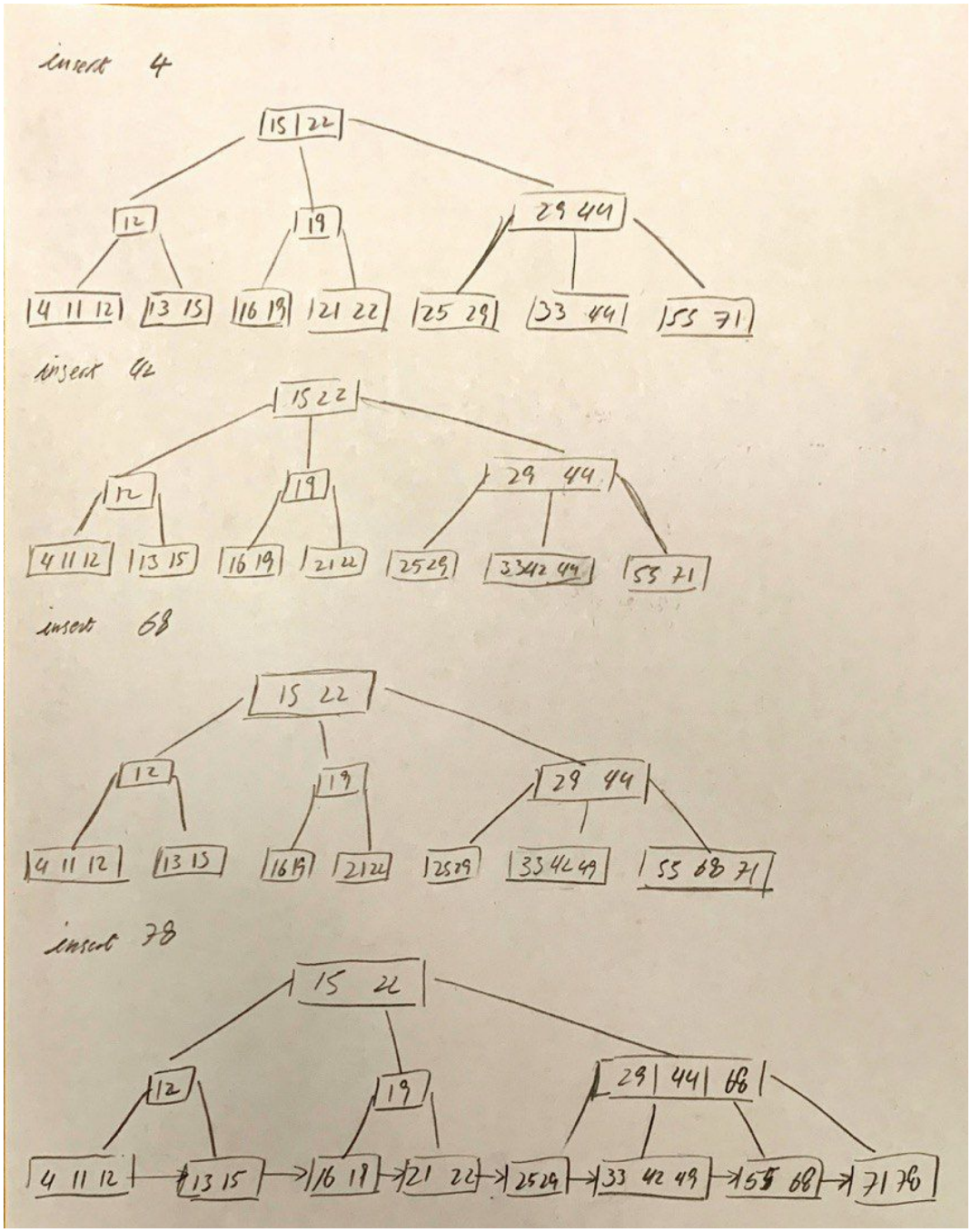
Problem 3. (25 pts) A PARTS file with Part# as key field includes records with the following Part# values: 12, 33, 22, 15, 19, 55, 25, 11, 16, 13, 29, 44, 71, 21, 4, 42, 68, 78.

CSE4701 - Databases
HW2 2020-03-04

Suppose that the search field values are inserted in the given order in a B+-tree of order p = 4 and pleaf =

iv. Show how the tree will expand (show all steps as in Fig 18.12 (6th ed)) and what the final tree will look like.





v. What is the fill ratio of the final B+-tree you created? (Note: we learned 69% is the average fill ratio in class.) Here the fill ratio is defined as (# of filled key values)/(# of maximally allowed key values) regardless each key value resides in index node or sequence set node.

$$Fill\ Ratio = \frac{\#\ of\ nodes\ in\ tree}{\#\ of\ possible\ nodes} = \frac{25}{36} = 69.44\%$$