Ryan Young

CSE 4701

3-3-2020

Homework 2

1.) *Consider a disk with block size B = 512 bytes. A block pointer p = 6 bytes long, and a record pointer is P_R = 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), and SALARY(4 bytes).*

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

The record size R in bytes would be the size of all the things that make up R. This means that 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.*

The blocking factor bfr is the number of records per block which means that with a block size of 512 bytes and a record size of 110 bytes there are 4.655 records per block. Since this disk is unspanned, and the not an integer then the bfr should be 4.

c.) *Suppose that the file is order by the key field SSN and we want to construct a primary index on SSN. Calculate*

i.) *The index block factor bfri*

The field size of SSN is 9 bytes, the record pointer is 7 bytes, 512 divided by 16 = 32 entries per block.

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

Since the first-level index entries in a primary index, number of file blocks = 100,000 records / 4 bfr = 25,000 blocks. The number of first-level index blocks = 25,000 blocks / 32 entries per block = 781 blocks.

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

Number of second level entries = number of first level blocks, number of second level blocks= 781 / 32 = 24. Which makes 24 third level entries hence we would have 3 levels because 24/32 is not a whole number and thus = 1.

iv.) *The total number of blocks required by the multilevel index*

781 + 24 + 1 = 806

v.) *The number of block accesses needed to search for a retrieve a record form the file---given its SSN value---using primary index.*

4 block accesses because there are three levels.

2.) *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*

Given:

Block Size = 512 bytes

Block Pointer = 6 bytes

Record Pointer = 7 bytes
Key Field of SSN = 9 bytes
We are using the block pointer because it is not dense.
Since a B-tree can have up to p block pointers, p – 1 record pointers and p -1 search field values that all must fit into one block:
(p-1) * (15) + 6p <= 512
15p – 15 + 6p <= 512
21p – 15 <= 512
P = 25
So, the order of b-tree is 25.

(ii) The number of levels needed if blocks are approximately 69% full (round up for convenience)

New order P will be 0.69 * p = 0.69 * 25 = 18 rounded up from 17.25.
Root: 1 node, 17 entries, 18 tree pointers
L1: 18 nodes, 18 * 17 entries, 18 * 18 tree pointers
L2: 18 * 18 nodes, 18 * 18 * 18 entries, 18 * 18 * 17 tree pointers
L3: 18*18*18 nodes, 18 * 18 * 18 * 17 entries = 99,144 entries
Number of levels = 4 ($\log_{18}(25{,}000) = 3.5 = 4$

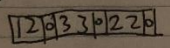
(iii) The worst-case number of blocks needed to search for and retrieve a record from the file—given its SSN value—using the B-tree you are estimating.

Worst case = # of levels + 1 so 5 block searches worst case.

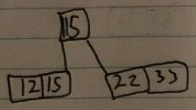
3.) A PARTS file with Part# as key field includes records with the following Part# values: 12, 33, 22, 15, 19, 55, 25, 11, 16, 13, 29, 44, 71, 21, 4, 42, 68, 78. Suppose that the search field values are inserted in the given order in a B+-tree of order p = 4 and pleaf = 3.

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

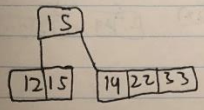
Insert first 3



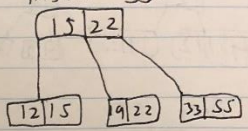
insert 15:



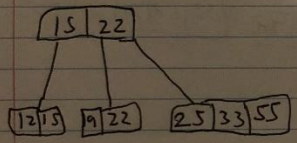
insert 19:



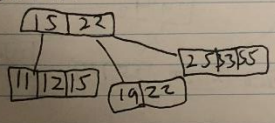
insert 55:



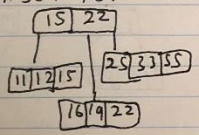
insert 25:



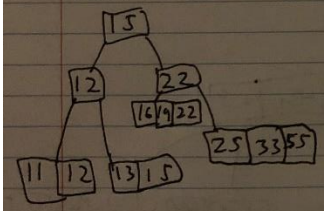
insert 11:



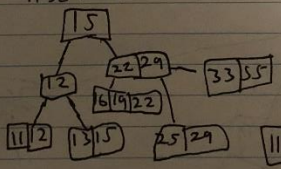
insert 16:



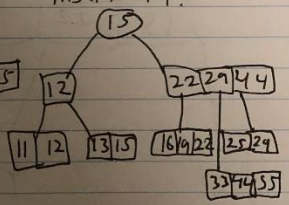
insert 13:



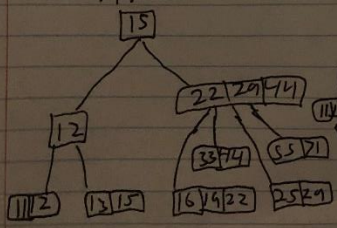
insert 29:



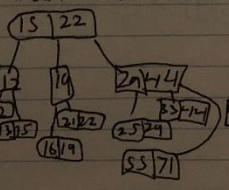
insert 44:



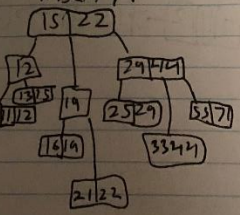
insert 71:

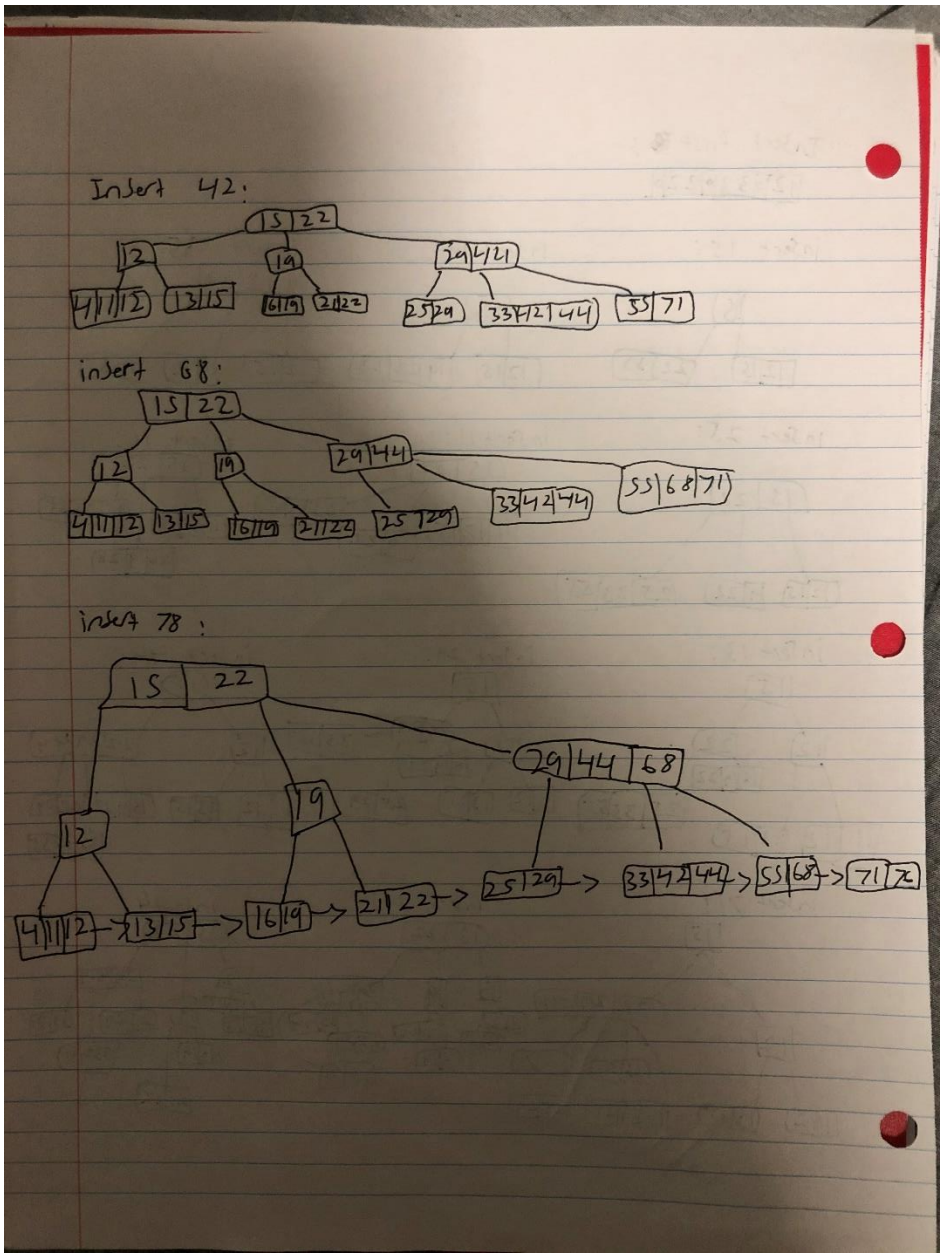


insert 21:



insert 4:





(ii) *(ii) 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.*

Since there are 25 nodes in the tree and there are 36 possible nodes we get a fill ratio of 69.44 percent.