**CS/SE 6360.004/005 Assignment-5**

**Due Date:** December 7, 2015, 11:59 PM

**1.** Consider a disk with block size B=512 bytes. A block pointer is P=6 bytes long,

and a record pointer is P R =7 bytes long. A file has r=30,000 EMPLOYEE records

of fixed-length. Each record has the following fields: NAME (30 bytes), SSN (9

bytes), DEPARTMENTCODE (9 bytes), ADDRESS (40 bytes), PHONE (9 bytes),

BIRTHDATE (8 bytes), GENDER (1 byte), JOBCODE (4 bytes), SALARY (4 bytes, real

number). An additional byte is used as a deletion marker.

**(50 points)**

(a) Calculate the record size R in bytes.

Solution:

R (Record length) = 30+9+9+40+9+8+1+4+4+ 1 (deletion marker) = 115 bytes

(b) Calculate the blocking factor bfr and the number of file blocks b assuming an

unspanned organization.

Solution:

Here,

B(Block size) = 512 bytes

bfr (blocking factor) = floor(B/R) = floor(512/115) = 4 records per block  
b (No of blocks needed for file) = ceiling(r/bfr) = ceiling(30000/4) = 7500 blocks

(c) Suppose 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 bfr i;

Solution:

Ri (Index record size) = (Vssn + P) = (9 + 6) = 15 bytes  
bfri (Index blocking factor) = floor(B/Ri) = floor(512/15) = 34

(ii) the number of first-level index entries and the number of first-level index blocks;

Solution:

No of first level index entries r1 = no of file blocks b = 7500 entries  
No of first level index blocks b1 = ceiling(r1/bfri) = ceiling(7500/34) = 221 blocks

(iii) the number of levels needed if we make it into a multi-level index;

Solution:

No of second level index entries r2 = no of first level blocks b1 = 221 entries  
No of second level index blocks b2 = ceiling(r2/bfri) = ceiling(221/34) = 7 blocks  
r3 = b2 = 7 entries  
b3 = ceiling(r3/bfri) = ceiling(7/34) = 1  
Since the third level has only one block, it is the top index level. Therefore, there are 3 index levels.

(iv) the total number of blocks required by the multi-level index; and

Solution:

Total no of blocks bi = b1+b2+b3 = 221+7+1 = 229 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.

Solution:

No of block accesses to search for a record = x+1 = 3+1 = 4.

(d) Suppose the file is not ordered by the key field SSN and we want to construct a

secondary index on SSN. Repeat the previous exercise (part c) for the secondary

index and compare with the primary index.

Solution:

1. Ri = (Vssn + P) = (9+6) = 15 bytes  
   bfri = fo = floor(B/Ri) = floor(512/15) = 34 index records per block
2. R1 = no of file records r = 30000  
   b1 = ceiling(r1/bfri) = ceiling(30000/34) = 883 blocks
3. R2 = b1 = 883 entries  
   b2 = ceiling(r2/bfri) = ceiling(883/34) = 26 blocks  
   r3 = b2 = 26 entries  
   b3 = ceiling(r3/bfri) = ceiling(26/34) = 1 block  
   Hence, the index has 3 levels.
4. Total no of blocks bi = b1+b2+b3 = 883+26+1 = 910 blocks
5. No of block accesses to search for a record = x+1 = 4.

(e) Suppose the file is not ordered by the non-key field DEPARTMENTCODE and we want to construct a secondary index on SSN using Option 3 of Section 18.1.3, with an extra level of indirection that stores record pointers. Assume there are 1000 distinct

values of DEPARTMENTCODE, and that the EMPLOYEE records are evenly distributed among these values.

Calculate

(i) the index blocking factor bfr i;

Solution:

Ri = (Vdepartmentcode + P) = (9+6) = 15 bytes  
bfri = floor(B/Ri) = floor(512/15) = 34 index blocks

(ii) the number of blocks needed by the level of indirection that stores record pointers;

Solution:

Avg no of records for each value = r/1000 = 30000/1000 = 30  
Since Pr = 7 bytes, the number of bytes needed at the level of indirection for each value of DEPARTMENTCODE is 7\*30 = 210 bytes, which fits in one block. Hence, 1000 blocks are needed for the level of indirection.

(iii) the number of first-level index entries and the number of first-level index blocks; (

Solution:

R1 = no of distinct values of DEPARTMENTCODE = 1000 entries  
B1 = ceiling(r1/bfri) = ceiling(1000/34) = 30 blocks

(iv) the number of levels needed if we make it a multi-level index;

Solution:

R2 = b1 = 30 entries  
B2 = ceiling(r2/bfri) = ceiling(30/34) = 1  
Hence, he index has x=2 levels

(v) the total number of blocks required by the multi-level index and the blocks used in the extra level of indirection; and

Solution:

Total no of blocks = bi = b1+b2+b indirection = 30+1+1000 = 1031 blocks

(vi) the approximate number of block accesses needed to search for and retrieve all records in the file having a specific DEPARTMENTCODE value using the index.

Solution:

No of block accesses = x+1 = 2+1 = 3 block accesses   
If we assume that the 30 records are distributed over 30 distinct blocks, we need an additional 30 block accesses to retrieve all 30 records. Hence, total block accesses needed on average to retrieve all the records with a given value for DEPARTMENTCODE = x+1+30 = 33.

(f) Suppose the file is ordered by the non-key field DEPARTMENTCODE and we want to construct a clustering index on DEPARTMENTCODE that uses block anchors (every new value of DEPARTMENTCODE starts at the beginning of a new block). Assume there are 1000 distinct values of DEPARTMENTCODE, and that the EMPLOYEE records are evenly distributed among these values. Calculate

(i) the index blocking factor bfr i;

Solution:

Ri = (Vdepartmentcode + P) = 9+6 = 15 bytes  
bfri = fo = floor(B/Ri) = floor(512/15) = 34 index records per block

(ii) the number of first-level index entries and the number of first-level index blocks;

Solution:

R1 = no of distinct DEPARTMENTCODE values = 1000 entries  
b1 = ceiling(r1/bfri) = ceiling(1000/34) = 30 blocks

(iii) the number of levels needed if we make it a multi-level index;

Solution:

R2 = b1 = 30 entries  
b2 = ceiling(r2/bfri) = ceiling(30/34) = 1  
Hence, index has x=2 levels.

(iv) the total number of blocks required by the multi-level index; and

Solution:

Total no of blocks bi = b1+b2 = 30+1 = 31 blocks

(v) the number of block accesses needed to search for and retrieve all records in the file having a specific DEPARTMENTCODE value using the clustering index (assume that multiple blocks in a cluster are either contiguous or linked by pointers).

Solution:

No of block accesses to search for the first block in the cluster of blocks = x+1=3.   
The 30 records are clustered in ceiling(30/bfr) = ceiling(30/4) = 8 blocks.   
Therefore, total block accesses needed on average to retrieve all the records with a given DEPARTMENTCODE = x+8 = 2+8 = 10 block accesses.

(g) Suppose the file is not ordered by the key field Ssn and we want to construct a B+ tree access structure (index) on SSN. Calculate

(i)the orders p and p leaf of the B+ tree;

Solution:

For a B + -tree of order p, the following inequality must be satisfied for each internal tree node: (p \* P) + ((p - 1) \* V SSN ) < B, or (p \* 6) + ((p - 1) \* 9) < 512, which gives 15p < 521, so p=34. For leaf nodes, assuming that record pointers are included in the leaf nodes, the following inequality must be satisfied: (p leaf \* (V SSN +P R )) + P < B, or (p leaf \* (9+7)) + 6 < 512, which gives 16p leaf < 506, so p leaf =31.

(ii) the number of leaf-level blocks needed if blocks are approximately

69% full (rounded up for convenience);

Solution:

Assuming that nodes are 69% full on the average, the average number of key values in a leaf node is 0.69\*p leaf = 0.69\*31 = 21.39. If we round this up for convenience, we get 22 key values (and 22 record pointers) per leaf node. Since the file has 30000 records and hence 30000 values of SSN, the number of leaf-level nodes (blocks) needed is b 1 = ceiling(30000/22) = 1364 blocks.

(iii) the number of levels needed if internal nodes are also 69% full (rounded up for convenience);

Solution:

We can calculate the number of levels as follows: The average fan-out for the internal nodes (rounded up for convenience) is fo = ceiling(0.69\*p) = ceiling(0.69\*34) = ceiling(23.46) = 24 number of second-level tree blocks b 2 = ceiling(b 1 /fo) = ceiling(1364/24) = 57 blocks number of third-level tree blocks b 3 = ceiling(b 2 /fo) = ceiling(57/24)= 3 number of fourth-level tree blocks b 4 = ceiling(b 3 /fo) = ceiling(3/24) = 1 Since the fourth level has only one block, the tree has x = 4 levels (counting the leaf level).

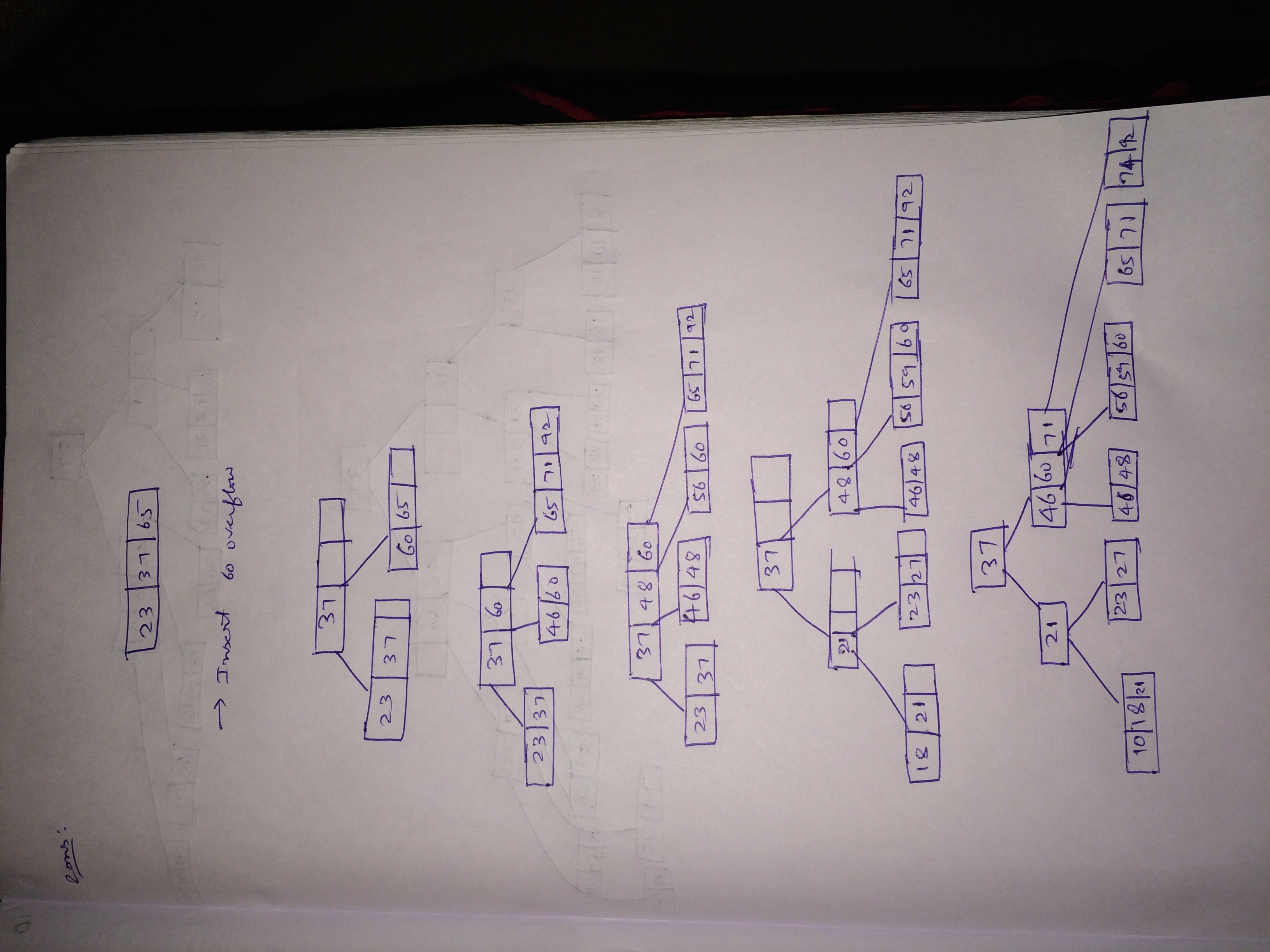
(iv) the total number of blocks required by the B+ tree; and (v) the number of block accesses needed to search for and retrieve a record from the file --given its SSN value-- using the B+ tree.

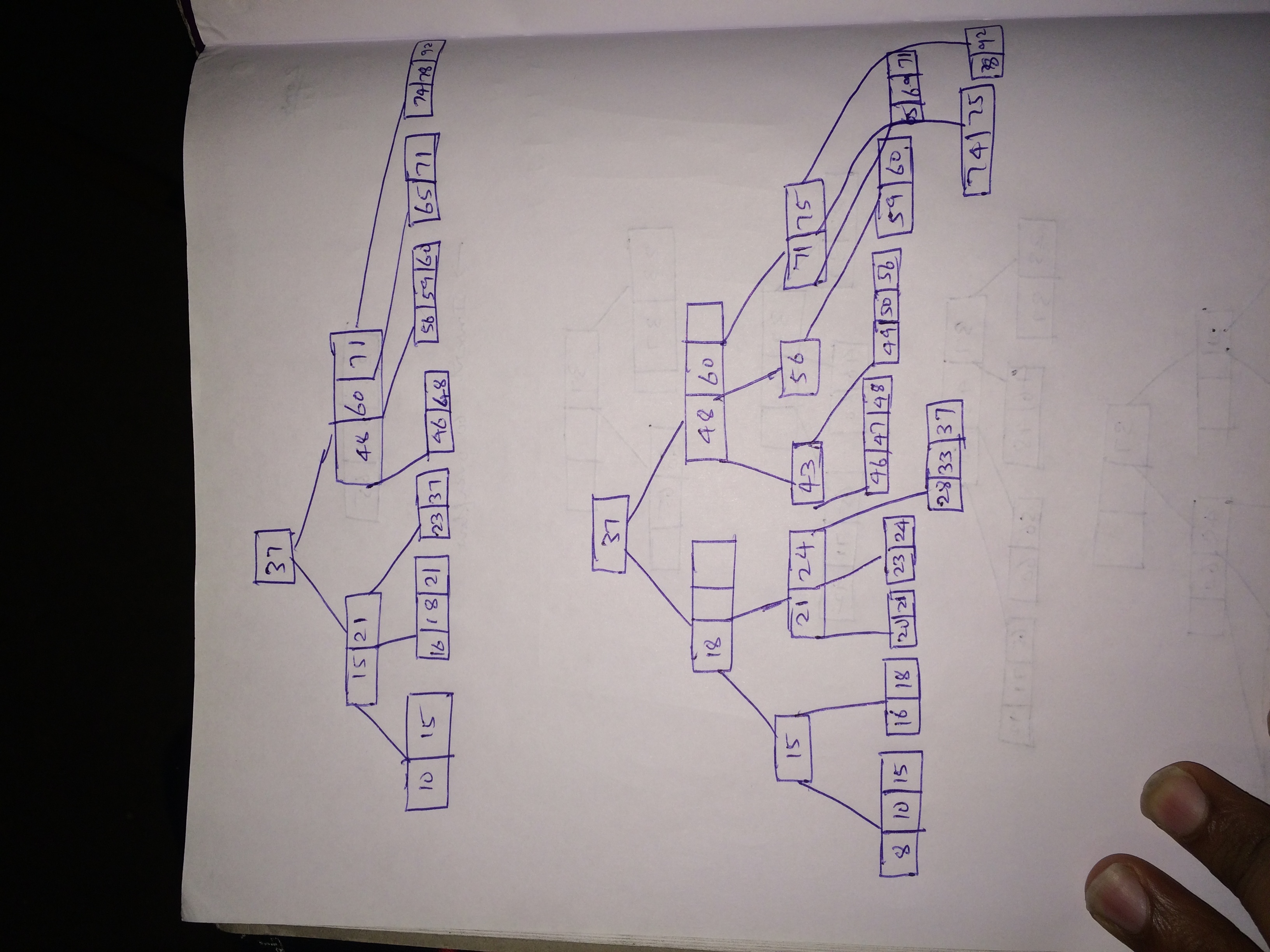
Solution:

Total no of blocks for the tree bi = b1+b2+b3+b4 = 1364+57+3+1 = 1425 blocks

**2.** A PARTS file with Part# as key field includes records with the following Part# values: 23, 65, 37, 60, 46, 92, 48, 71, 56, 59, 18, 21, 10, 74, 78, 15, 16, 20, 24, 28, 39, 43, 47, 50, 69, 75, 8, 49, 33, 38. Suppose the search field values are inserted in the given order in a B+ tree of order p=4 and p leaf =3; show how the tree will expand (only after every split) and what the final tree looks like. **(30 points)**

Solution:





**3.** Optimize the following SQL query on the Company Database to find names of employees earning over $80,000 per year, names of projects in which they work more than 30 hours, where the project is located in Chicago and the manager of the project's controlling department started after January 1, 2009.

Select Lname, Fname, Pname, Hours

From Project P, Employee E, Department D, Works\_on W

Where E.Ssn = W.Essn

and P.Dnum = D.Dnumber

and W.Pno = P.Pnumber

and Plocation = 'Chicago'

and Hours > 30

and Salary > 80000

and Mgr\_start\_date >= '1/1/2009'

Show the final query tree.  **(20 points)**

Solution:

