04 Indexing (B/B+ tree)

1. (d)

It is given that key size + Record pointer size = Record size

This makes $\left\lfloor \frac{\text{Blocks size} - \text{Header}}{\text{Key} + \text{Record pointer}} \right\rfloor$ = Index block factor = $\left\lfloor \frac{\text{Blocks size} - \text{Header}}{\text{Record size}} \right\rfloor$

= Database file block factor

2. (12)

Calculating the blocking factor of the relation,

Blocking factor = $\left\lfloor \frac{2400}{200} \right\rfloor = 12$

Number of blocks in file = $\left\lceil \frac{500000}{12} \right\rceil = 41667$

Blocking factor of index file = $\left\lfloor \frac{2400}{20} \right\rfloor = 120$

Number of blocks of 1st level index = $\left\lceil \frac{41667}{120} \right\rceil = 348$

Number of blocks of 2nd level index = $\left\lceil \frac{348}{120} \right\rceil = 3$

Number of blocks of 3rd level index = $\left\lceil \frac{3}{120} \right\rceil = 1$

Cost of query using index = Height of index + 1 block for file = 3 + 1 = 4

Cost of query using binary search

= $\lceil \log_2 41667 \rceil = 16$

Difference = 16 – 4 = 12

3. (a)

Content of index <key, BP > = 6 + 10 = 16

Number of block in database = $\frac{8192}{512} = 256$

In first level entry for each record,

Total size = 256×16

Number of blocks in first level

= $\frac{\text{Size of first level index}}{\text{Block size}} = \frac{4096}{512} = 8$

In second level

Total size = Number of entries × Size of entry

= 8×16

Number of blocks in second level

= $\frac{\text{Size of second level index}}{\text{Block size}}$

= $\frac{64}{512} = 0.125 \approx 1$

4. (2)

If the book were organized as a heap file in a linear search so worst case bound is 500. By using binary search the worst case bound is $\lceil \log_2 500 \rceil = 9$ using the fact that the book is ordered by names with an index for the name of the first entry on each page we get worst case bound is 2 because the index has 500 pages, so entire index files in one page 1 access to the index and 1 to the data.

5. (3)

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

Blocking factor bf = Floor (B/R) = Floor $\left(\frac{512}{115} \right)$

= 4 records per block

Number of blocks needed for file

$= \text{Ceiling}(r/bf) = \text{Ceiling}\left(\frac{30000}{4}\right) = 7500$

Index entry size = (VSSN + P)
= (9 + 6) = 15 bytes

Index blocking factor = floor (B/Index record size) = floor $\left(\frac{512}{15}\right) = 34$

Number of first-level index entries R1 = Number of file blocks b = 7500 entries

Number of first-level index blocks B1 = Ceiling(R1/Index blocking factor)

$= \text{Ceiling}\left(\frac{7500}{34}\right) = 221$ blocks

Number of second-level index entries R2 = Number of first-level blocks B1 = 221 entries

Number of second-level index blocks B2 = Ceiling(R2/Index blocking factor)

$= \text{Ceiling}\left(\frac{221}{34}\right) = 7$ blocks

Number of third-level index entries R3 = Number of second-level index blocks B2 = 7 entries

Number of third-level index blocks B3 = Ceiling(R3/Index blocking factor)

$= \text{Ceiling}\left(\frac{7}{34}\right) = 1$

Since the third level has only one block, it is the top index level.

So 3 levels are required.

6. (d)

The question asks about the average number of blocks over which $m + 1$ records extend (m is for movies and 1 for studio entry), if 10 fit in a block, and they start at a random slot on a block.

Suppose, if $m = 0$, the answer is clearly 1 (only 1 block is required to be accessed as it can hold both studio entry and its movie entry) and if $m = 10$, the answer is clearly 2 (in case of $m = 10$, the total entries are 11 (10 are of movies and 1 is of studio) and since a block can hold only 10 entries, therefore, we need to access 2 blocks in order to get them). As the function must be linear in 'm', we can infer that the average is always '$\frac{m}{10}$'.

7. (d)

I. This is false. A non clustered index is stored at one place and table data is stored in another place. This is similar to a textbook where the book content is located in one place and index is located in another. Therefore non-clustered index improve the performance of the queries not covered by the clustered index.

II. *B*-tree is never used for sequential access of records.

8. (89)

Since, index block factor = $\frac{512}{(10+5)} = 34$

The index built is secondary; therefore, there will be an index entry for each record. So, at first level index blocks will be

$\left\lceil\frac{300}{34}\right\rceil = \lceil 88.23\rceil = 89$

9. (c)

We are given that a block can hold 6 records when n denotes the total number of records, therefore, the total number of blocks in the data files are $\frac{n}{6}$. Since a block can have 15 key pointer pairs, therefore, the total number of blocks in the index will be $\frac{n}{15}$. Thus, the number of blocks needed to hold the data file in the dense index will be given by the sum of the total blocks in the data file and in the index file,

i.e., $\frac{n}{6}+\frac{n}{15}=\frac{7n}{30}$.

10. (c)

Option (a) and option (b) are not correct because index can be created using any column or combination of columns, which need not be unique. Basically, indexed column is used to sort the rows of table. Whose data record of file is sorted using index so the correct option is option (c).

11. (a)

In the indexed scheme of blocks to a file, the maximum possible size of the file depends on the number of blocks used for index and the size of index.

12. (c)

Clustering index is created on data files whose files records are physically ordered on a non-key field which doesn't have a distinct value for each record.

13. (3374)

Level 0	Level 1	Level 2
↓	↓	↓
14 +	15 × 14 +	15 × 15 × 14
= 3374		

14. (4)

Sparse index and maximum (minimum filling in nodes)

Number of blocks in file = $\frac{1250}{3}$ = 417 blocks

Minimum number of index blocks at 1st level

$$= \frac{417}{5} = \lfloor 83.4 \rfloor = 83$$

Minimum number of index blocks at 2nd level

$$= \left\lceil \frac{83}{6} \right\rceil = 13$$

Minimum number of index blocks at 3rd level

$$= \left\lceil \frac{13}{6} \right\rceil = 2$$

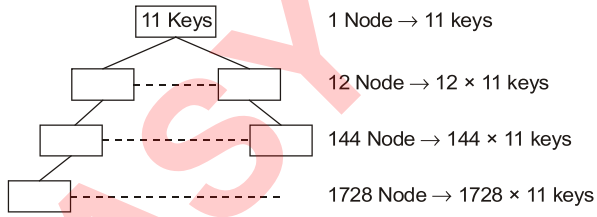
Minimum number of index blocks at 4th level = 1

So, maximum number of level are 4.

15. (0)

Number of keys = 7000

***B*-Tree:** Order P = 12 [Max possible child pointer in node]



Hence, number of levels required are 4.

***B*+ Tree:** Leaf node order = 11 (key R_P) + Block pointers

Internal node order = 11 keys + 12 child pointers

1. 5 child pointer — 1 node
2. 54 child pointers — $\left\lceil \frac{54}{12} \right\rceil$ = 5 nodes
3. 637 child pointers — $\left\lceil \frac{637}{12} \right\rceil$ = 54 nodes
4. 7000 keys — $\left\lceil \frac{7000}{11} \right\rceil$ = 637 nodes

Level = 4

Difference = 4 – 4 = 0

16. (9000)

In level 2 there will be 10 nodes as order is 10.

In level 3 there will be 10 × 10 = 100 nodes

In level 4 there will be 10 × 100 = 1000 nodes

At level 4 each node will have 9 record pointers.

So, therefore the maximum number of records that be indexed are 1000 × 9 = 9000.

17. (5461)

Maximum number of node at 1st level = 1
Maximum number of node at 2nd level = 4 (because order is 4 i.e., 4 block pointers)
Maximum number of node at 3rd level = 4 × 4 = 16
Maximum number of node at 4th level = 16 × 4 = 64
⋮
Maximum number of node at 7th level = 1024 × 4 = 4096
So total number of nodes = 1 + 4 + 16 + 64 + 256 + ... + 4096 = 5461

18. (4)

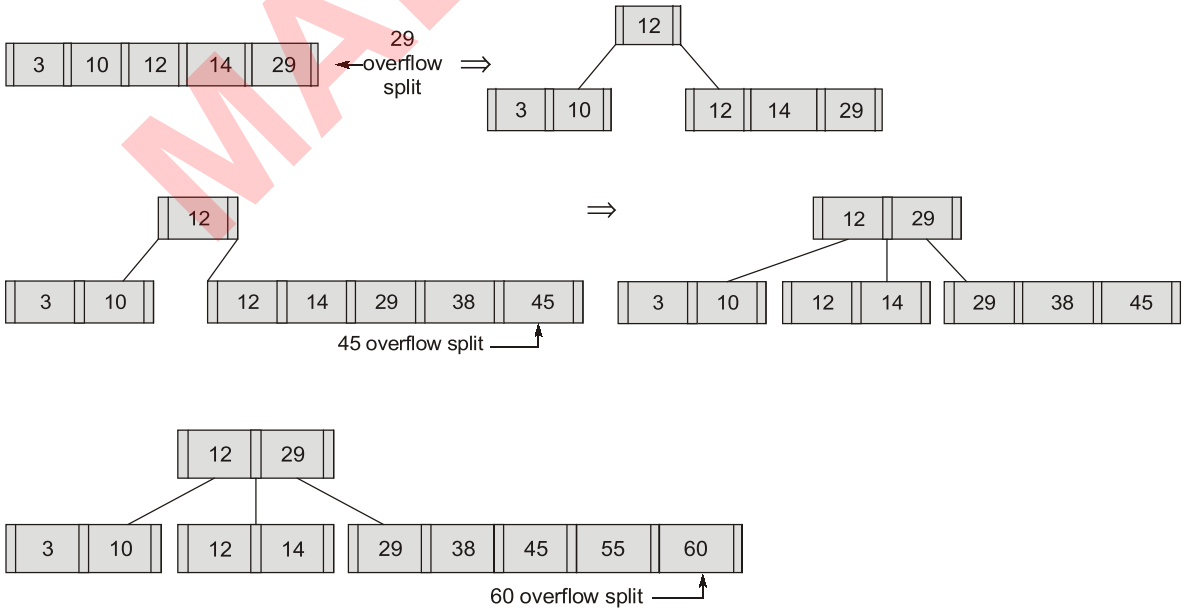
Leaf node order = 9 (keys R_P) + Block pointers
Internal node order = 9 keys + 10 child pointers

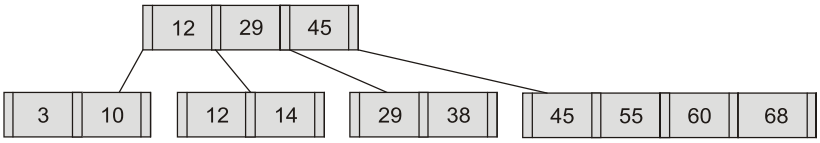
1. 5000 keys $= \left\lceil \frac{5000}{9} \right\rceil$ = 556 nodes
2. 556 child pointers $= \left\lceil \frac{556}{10} \right\rceil$ = 56 nodes
3. 56 child pointers $= \left\lceil \frac{56}{10} \right\rceil$ = 6 nodes
4. 6 child pointers = 1 node

So, 4 level of indexes are required.

19. (3)

Number of key in internal node = 4 i.e, [5 – 1 = 4]
Number of key in leaf node = 4 i.e., [order = # Key]





So the maximum number of splits of leaf node is 3.

20. (a)

Order P:

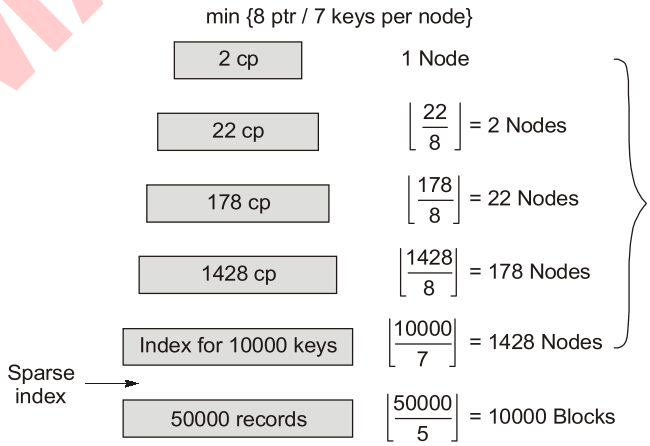
$$P \times \text{Pointer} + (P-1) \times \text{Key} \le \text{Block}$$
$$P \times 8 + (P-1)\,12 \le 1000$$
$$20\,P \le 1012$$
$$P = \left\lfloor \frac{1012}{20} \right\rfloor = 50$$

Level	Min nodes	Min B_P	Min keys
1	1	2	1
2	2	2 × 25	24
3	50	–	50 × 24
			1200

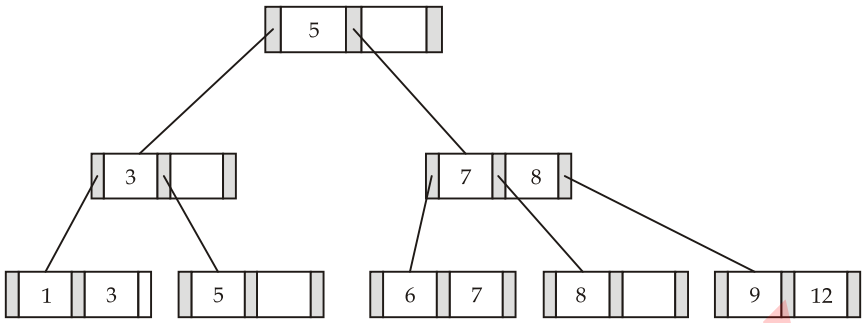
21. (b)

Order $P \Rightarrow$ $P \times B_P + (P-1)\,\text{Key} \le \text{Block}$
$$P \times 12 + (P-1)\,20 \le 512$$
$$32\,P \le 532$$
$$P = \left\lfloor \frac{532}{32} \right\rfloor = 16$$

Maximum index nodes in index mean min fill factor



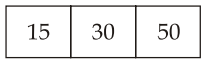
22. (18)



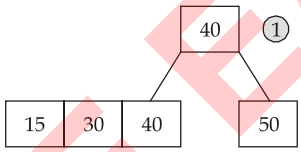
3 + 7 + 8 = 18 at height 1.

23. (8)

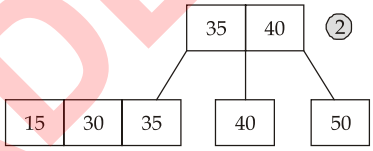
1. On insertion of 50, 15, 30



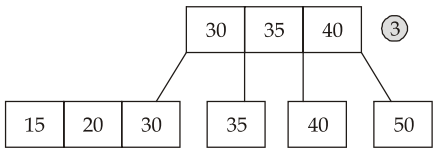
2. On insertion 40



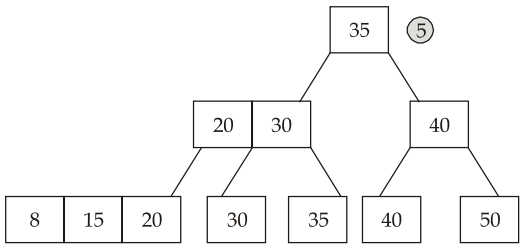
3. On insertion 35



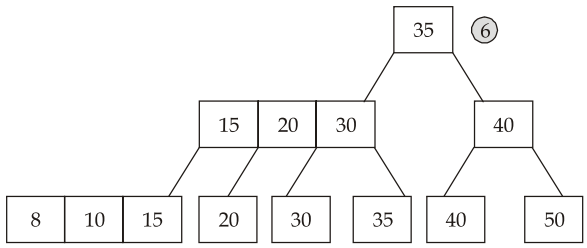
4. On insertion 20



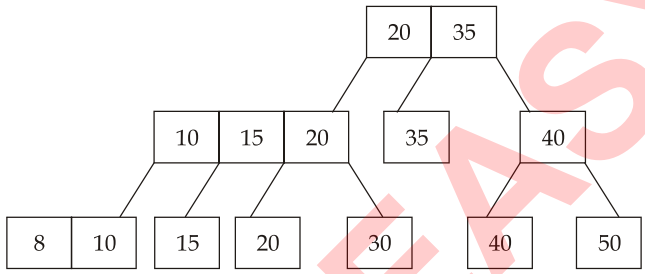
5. On insertion 8



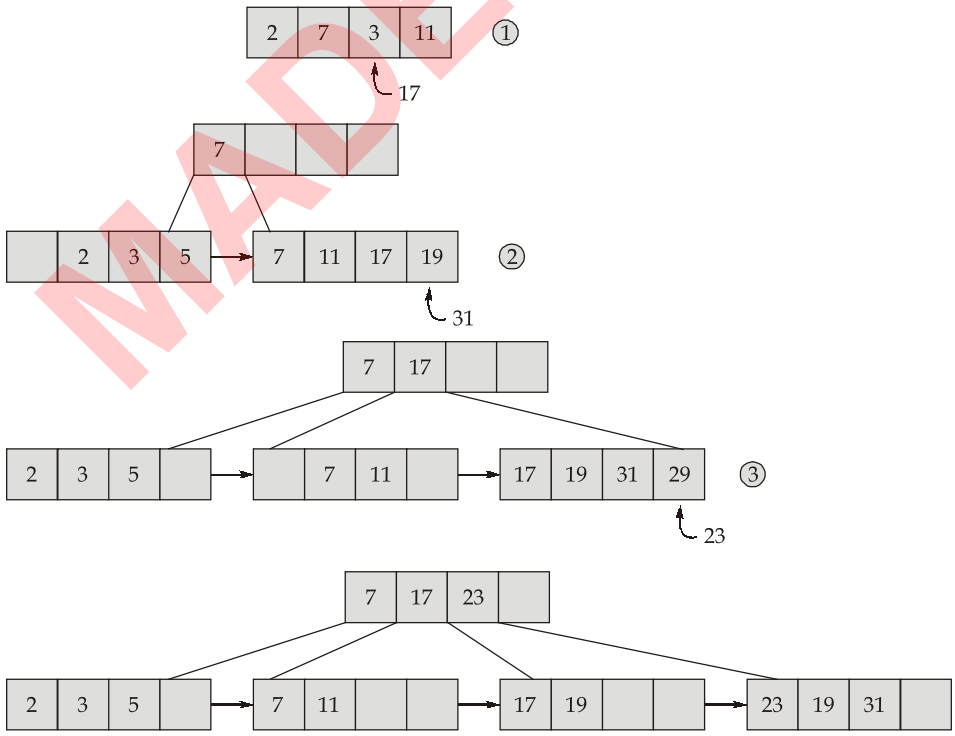
6. On insertion 10



7. On insertion 5
Two more times nodes going to split.
So 6 + 2 = 8 times



24. (71)



Sum of keys which make split of leaf nodes are 17 + 31 + 23 = 71

25. (75)

Assume order of leaf node is P

Format of B$^+$ tree leaf node is 1

$$B_p + P \times (\text{Key size}) + (P)\, R_p \leq \text{Block size}$$

$$P \times (15) + (P)\, 12 + 10 \leq 2048$$

$$27\, P \leq 2038$$

$$P \leq \lfloor 75.48 \rfloor$$

$$P \leq 75$$

26. (b)

If disk block allocated for B^+ tree index and same size disk block allocated for B tree index then number of index blocks and I/O cost of B^+ tree index less than or equal to B tree index for given distinct keys.

27. (3)

If we insert keys 45,48,55 in same order, then on insertion of key 55, root will be overflow and new level will be created

28. (c)

Since a block can hold 10 records, therefore, there are 100,000 data blocks

$$\left(\frac{1000000}{10} = 100000\right)$$

If there are an average of 70 pointers per block in the bottom-levels nodes of the B-tree, then there are $\frac{1000000}{70} = 14286$ B-tree blocks at that level. The next level of the B-tree requires $\frac{1}{70^{th}}$ of that, or 204 blocks, and the third level has $\frac{1}{70^{th}}$ of that, or B blocks. The fourth level has only the root block. The number of blocks that needed is therefore

100,000 + 14,286 + 204 + 3 + 1 = 114,494 blocks

29. (b)

Indexing nodes is more in B^+ trees because it holds key value both in internal nodes as well as in leaf node while B tree only has one entry for one key. Since, in B^+ trees search is repeated but this is not the case with B tree, therefore the number of levels in B^+ trees is more than that of B tree.

30. (63)

Since, order = 4

Therefore, minimum keys at root, internal node or leaf = 1

and, node pointers = 2

for height = 5, the number of levels will be = 6

So, number of keys at level 0 is 1 // root node

number of keys at level 1 is 2.

at level 2 is 4

at level 3 is 8

at level 4 is 16

at level 5 is 32

Therefore,

Total keys = 32 + 16 + 8 + 4 + 2 + 1 = 63

31. (57)

$$= 5 \times K + (K-1)(7+6) \leq 1024$$

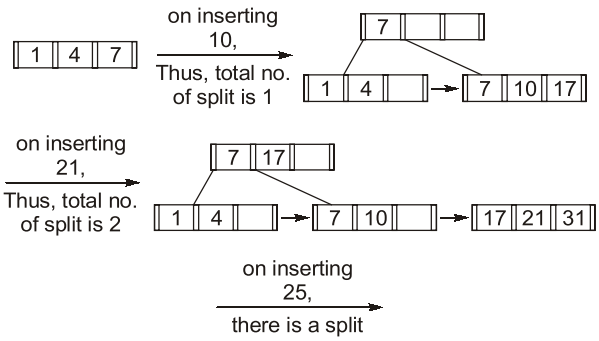
$$= 5K + (K-1)13 \leq 1024$$

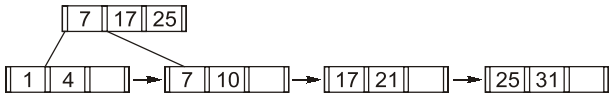
$$= 5K + 13K - 13 \leq 1024$$

$$= 18\, K \leq 1024 + 13$$

$$K = \text{floor}\left(\frac{1037}{18}\right) = 57$$

32. (3)





Thus, there are total 3 splits.

33. (c)

Since we are always adding at the right extreme of the *B* tree, once split, the first of the pair never changes. Thus, the leaf blocks will each have two keys only (1, 2) (3, 4) (5, 6) and so on. At the second level, we divide pointer 3 to the left and 2 to the right, so each except the rightmost node at that level has three pointers. The same holds for any other level blocks with 3, 3, 3, 3 and 2 pointers. These 14 pointers go to leaf blocks with two pointers each, for a total of 40 pointers to records.

Thus, when the record with the key 40 is inserted, 4^{th} level created.

34. (40)

Level	Maximum nodes	Max block pointers
0	1	3
1	3	3 × 3 = 9
2	9	9 × 3 = 27
3	27	27 × 3 = 81
Total nodes	40	

Thus, maximum number of nodes will be 40.

35. (6250)

If filled to the minimum five pointers, the first five levels (total levels will be 6 including the root level) give us 2 × 5 × 5 × 5 × 5 = 1250 leaf nodes. (Root node will have only one key in minimum therefore, only 2 block pointers in the first level)

If leaf nodes are half full of record pointers, they contain 5 records each.

So, 1250 × 5 = 6250 record pointers.

36. (d)

The order given = 30 and the *B*-tree node is 65% full. Therefore, order of *B*-tree, according to the utilization will be 0.65 × 30 = 19.5 = 20

So, at level 0 (root) : 19 keys

At level 1 : 20 × 19 keys = 380 keys

At level 2 : 20 × 20 × 19 keys = 7600 keys

37. (4)

Given that the total number of records = 1250 and each block contain 3 records.

Therefore, the total number of blocks required to allocate all the records are = $\frac{1250}{3}$ = 416.6

= 417 blocks.

Since, it is mentioned that indexing is sparse, therefore; the number of blocks required in index

= $\frac{417}{10}$ = 42.

So, B^+ tree will be having 42 blocks at ground level.

Next level will have $\frac{46}{6}$ = 7 blocks

In the next level, it will have $\frac{7}{6}$ = 2 blocks

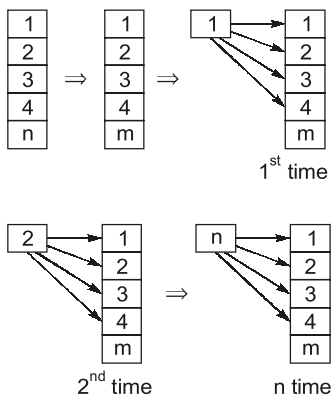
In the next level, it will have $\frac{2}{6}$ = 1 block

So, in total we will have 4 levels.

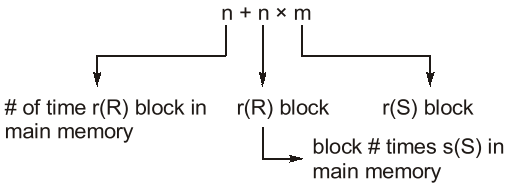
38. (b)

39. (a)

Block nested loop join.



i.e. If r(R) is outer



40. (15020)

Nested Loop algorithm will involve $n_r \times b_s + b_r$ block transfers.

n_r = # records in relation r,
b_r = # blocks in relation r
n_s = # records in relation s,
b_s = # blocks in relation s

Either T_1 can be R or T_2.

If R is T_1 then total number of block access is $4000 \times 20 + 50 = 80050$

If R is T_2 then total number of block access is $300 \times 50 + 20 = 15020$

Better one is the second case, total number of block accesses = (15020).

41. (c)

Number of tuples in R = Nr = 40000
Number of tuples in S = Ns = 60000
Block size = 150 tuples/block

$$\text{Number of blocks in } R = \text{Br} = \left\lceil \frac{40000}{150} \right\rceil = 267$$

$$\text{Number of blocks in } S = \text{Bs} = \left\lceil \frac{60000}{150} \right\rceil = 400$$

Optimality is achieved when the outer relation is smaller. Therefore, R is the outer relation.

Number of block transfers in NLJ

$= B_{(\text{outer})} + N_{(\text{outer})} \times B_{(\text{inner})}$
$= 267 + 40000 \times 400$
$= 16000267$

42. (a)

Except hash, we do not use any formula to determine the physical location of records.

■■■■