File systems and Indexing

1) Given a B-tree where each node can contain at most 5 keys, determine the order of the B-tree.

Solution:

- The maximum number of keys per node is 5.
- For a B-tree of order mmm, each node can contain at most m-1 keys.
- Therefore, m−1=5 implies m=6.

The order of the B-tree is **6**.

2) Calculate the minimum height of a B-tree of order 4 (a 4-ary B-tree) that contains 1000 keys.

Solution:

- For a B-tree of order m, each node (except the root) has at least $\lceil m/2 \rceil$ children.
- Each node contains $k - 1$ keys where k is the number of children.
- In the worst case, each node has the minimum number of children, which is $\lceil m/2 \rceil = \lceil 4/2 \rceil = 2$.

At each level, the number of keys grows by a factor of the minimum number of children:

- Level 0 (root): 1 node, 0 keys
- Level 1: 2 nodes, 2 keys
- Level 2: $2^2 = 4$ nodes, $4 \times (2 - 1) = 4$ keys
- Level 3: $2^3 = 8$ nodes, $8 \times (2 - 1) = 8$ keys

The height h can be found by solving:

$$\sum_{i=0}^{h-1} 2^i = \frac{2^h - 1}{2 - 1} = 1000$$

$$2^h - 1 = 1000 \Rightarrow 2^h = 1001 \Rightarrow h = \lceil \log_2(1001) \rceil = 10$$

The minimum height of the B-tree is **10**.

3) What is the maximum number of nodes in a B-tree of order 3 with height 3?

Solution:

- For a B-tree of order m, each node can have at most m children.
- Height $h = 3$.

At each level, the number of nodes grows by a factor of m:

- Level 0 (root): 1 node
- Level 1: 3 nodes
- Level 2: $3^2 = 9$ nodes
- Level 3: $3^3 = 27$ nodes

Total number of nodes:

$$1 + 3 + 9 + 27 = 40$$

The maximum number of nodes in the B-tree is **40**.

4) You are given a B-tree in which each node (except the root) has at least 3 keys and at most 6 keys. Determine the order mmm of the B-tree.

Solution:

- The minimum number of keys in a node (except the root) is 3.
- The maximum number of keys in a node is 6.

For a B-tree of order m:

- The minimum number of keys in a node is $\lceil \frac{m}{2} \rceil - 1$.
- The maximum number of keys in a node is $m - 1$.

Given that the minimum number of keys is 3:
$\lceil \frac{m}{2} \rceil - 1 = 3$
$\lceil \frac{m}{2} \rceil = 4$

Since $\lceil \frac{m}{2} \rceil$ is an integer, $\frac{m}{2}$ must be either 4 or 4.5.
$m = 8$

Given that the maximum number of keys is 6:
$m - 1 = 6$
$m = 7$

Since we must satisfy both conditions, we have an inconsistency with $m = 8$ not fitting. Thus, re-evaluating, if the constraints don't match, likely misinterpreted keys per node rules.

Hence, the order of the B-tree is **7**.

5) In a B-tree, suppose the search key is 8 bytes long, the disk block size is 1024 bytes, the record pointer is 8 bytes, and the block pointer is 8 bytes. Calculate the order of the B-tree node.

Solution:

The total space required for $m-1$ keys and m pointers is:

$$(m-1) \times (\text{key size} + \text{record pointer size}) + m \times (\text{block pointer size})$$

Given:

- Key size = 8 bytes
- Record pointer size = 8 bytes
- Block pointer size = 8 bytes
- Disk block size = 1024 bytes

$$(m-1) \times (8+8) + m \times 8 \leq 1024$$
$$(m-1) \times 16 + m \times 8 \leq 1024$$
$$16m - 16 + 8m \leq 1024$$
$$24m - 16 \leq 1024$$
$$24m \leq 1040$$
$$m \leq \frac{1040}{24}$$
$$m \leq 43.33$$

Since m must be an integer, the maximum order m is 43.

Therefore, the order of the B-tree node is **43**.

6) In a B tress suppose search key is 9 bytes long, disk block size is 512 B,
record pointer is 7 B, block pointer is 6 B, then calculate the order of the B-tree node.

Solution:. Let maximum number of children to be placed = n
n ∗ BlockP ointer + (n − 1)(k + pr) ≤ BlockSize
Solve the above equation by entering the values:
BlockP ointer = 6
pr = 7
k = 9
Block Size = 512
n(6) + (n − 1)(9 + 7) ≤ 512
6n + 16n − 16 ≤ 512
22n ≤ 528
n ≤ 24
∴ Order(M ax.NumberofChildren) = 24

7) Consider a B-tree with key size 10 bytes, Block size 512 B, data pointer is 8 B and block pointer is 5 B. What is the order of B-tree?

Solution:. Let maximum number of children to be placed = n
$n * BlockPointer + (n - 1)(k + pr) \leq BlockSize$
Solve the above equation by entering the values:
BlockPointer = 5
pr = 8
k = 10
Block Size = 512
$n(5) + (n - 1)(10 + 8) \leq 512$
$5n + 18n - 18 \leq 512$
$23n \leq 530$
$n \leq 23.04$
$\therefore$ Order(Max.NumberofChildren) = 23

8) Search key field is given as 9 bytes, block size is 512 B, record pointer pr is 7 B, block pointer p is 6 B, What is the order of internal nodes and leaf node of B+-tree.

Solution:. Let order of internal node = o
$o(p) + (o - 1)(k) \leq BlockSize$ Solve the above equation by entering the values:

BlockPointer = 6B
pr = 7B
k = 9
Block Size = 512
$o(6) + (o - 1)(9) \leq 512$
$15o \leq 512$
$n \leq 34$
$\therefore$ Order (Max. Number of Children) of Internal Node = 34
In B
+-tree all leaves are interconnected so it makes range queries easier.
$Oleaf(K + Pr) + P \leq B$
$Oleaf(7 + 9) + 6 \leq 512$
$16 * Oleaf \leq 512$
$Oleaf \leq 31.6$
$\therefore$ Order of Leaf Node = 31

9) A data file consisting of 1,50,000 student-records is stored on a hard disk with block size of 4096 bytes. The data file is sorted on the primary key RollNo. The size of a record pointer
for this disk is 7 bytes. Each student-record has a candidate key attribute called ANum of size 12
bytes. Suppose an index file with records consisting of two fields, ANum value and the record pointer
the corresponding student record, is built and stored on the same disk. Assume that the

records of
data file and index file are not split across disk blocks. The number of blocks in the index file is

Solution:

Record size in Index = 12+7 = 19 and Block size = 4096 Number of Index records in 1 Block = floor(4096/19) = 215 records in 1 block. Number of Blocks in the Index File = Total Number of records/Records per block = ceil(1,50,000/215) = 698

10) Consider a database implemented using B+ tree for file indexing and installed on a disk drive with block size of 4 KB. The size of search key is 12 bytes and the size of tree/disk pointer
is 8 bytes. Assume that the database has one million records. Also assume that no node of the B+
tree and no records are present initially in main memory. Consider that each record fits into one
disk block. The minimum number of disk accesses required to retrieve any record in the database is

Solution:

Given,

1. Search Key: 12 bytes
2. Tree Pointer: 8 bytes
3. Block Size: 4096 bytes
4. Number of database records: 10^6

Since it's a B^+ tree, an internal node only has search key values and tree pointers. Let p be the order of an internal node. Hence,

$$p(8) + (p-1)(12) \leq 4096$$

which gives $p \leq 205.4$.

Therefore $p = 205$

Now,

Level	Nodes	Keys	Pointers
1	1	204	205
2	205	204×205	205^2
3	205^2	204×205^2	205^3

Level 3 alone has approximately 8.5×10^6 entries. So we can be sure that a 3-level $B+$ tree is sufficient to index 10^6 records.

So to access any record (in the worst case), we need 3 block access to search for the record in the index along with 1 more access to actually access the record.

Hence, 4 accesses are required.

11) In a file which contains 1 million records and the order of the tree is 100, then what is the maximum number of nodes to be accessed if B+ tree index is used?

Solution:
We have to find the maximum no. of nodes to be accessed in B+ tree so we have to consider

the minimum fill factor.
Here,
number of record = 1 million = 10^6 (Given)
order of b+tree= number of pointers per node = p = 100 (Given)
Minimum pointers per node =⌈ p/ 2⌉ = ⌈ 100/ 2⌉ = 50
number of nodes in last level of tree =10^6 / 50 = 2 * 10^4
Number of nodes in Second last level of tree = 2*10^4 / 50 = 400
Number of nodes in Third last level of tree = 400/50 =8
number of node in Fourth last level of tree = 8/50 =1
The maximum no. of nodes to be accessed = number pf B+ tree levels = 4

12) In a B+ tree, if the search -key value is 8 bytes long, the block size is 512 bytes and the block pointer size is 2 bytes, then maximum order of the tree is

Solution:
Order of a B+ tree node is maximum number of children
in an internal node

Let the order be x. Number of keys in a node is equal to
number children minus 1.
So a full node has (x-1) keys and x children.

(x-1)*(search key) + x * block ptr <= block size

(x-1)*8 + x*2 <= 512
10x <= 520
x <= 52

13) The order of leaf node in B+ - tree is the maximum number of (value, data record pointer) pairs it can hold. Given that the block size is 1K bytes, data record pointer is 7 bytes long,
the value field is 9 bytes long and a block pointer is 6 bytes long, what is the order of the leaf node?

Solution:
Disk Block size = 1024 bytes

Data Record Pointer size, r = 7 bytes
Value size, V = 9 bytes
Disk Block ptr, P = 6 bytes
Let order of leaf be m. A leaf node in B+ tree contains at most m record pointers, at most m values, and one disk block pointer. r*m + V*m + p <= 1024 so, 16m <= 1018 or m =< 63.

14) Given a block can hold either 3 records or 10 key pointers. A database contains n records, then how many blocks do we need to hold the data file and the dense index

Solution:
Number of blocks needed to store data file with n records = n/3 Number of blocks needed to store dense file index = n/10 Total blocks required = n/3 + n/10 = 13n/30 blocks.

15) Consider a B+ tree in which the search key is 12 bytes long, block size is 1024 bytes, record pointer is 10 bytes long and block pointer is 8 bytes long. The maximum number of keys that
can be accommodated in each non-leaf node of the tree is .

Solution:
Let m be the order of B+ tree

m(8)+(m-1)12 <= 1024
[Note that record pointer is not needed in non-leaf nodes]

m <= 51

Since maximum order is 51, maximum number of keys is 50.

16) Calculate the order of leaf (Pleaf) and non leaf (P) nodes of B+ tree based on the information given below:
search key field: 12 field
Record Pointer: 10 bytes
Block Pointer: 8 bytes
Block Size: 1KB

A) Pleaf : 51 and P: 46
B) Pleaf : 47 and P: 52
C) Pleaf : 46 and P: 51
D) Pleaf : 52 and P: 47

Solution:

i) leaf node:
let the order of leaf be \'n\'
size of search key field * n + record pointer * n + block pointer <= 1024
12 * n + 10 * n + 8 <= 1024
22n <= 1016
n = 46

ii) for non-leaf node:
size of search key field * n + block pointer * (n+1) <= 1024
12 * n + 8 * n + 8 = 1024
n = 50.8
order of non-leaf node (p) = 50

17) Consider a file of 16384 records. Each record is 32 bytes long and its key field is
of size 6 bytes. The file is ordered on a non-key field, and the file organization is unspanned. The
file is stored in a file system with block size 1024 bytes, and the size of a block pointer is 10 bytes.
If the secondary index is built on the key field of the file, and a multi-level index scheme is used to

store the secondary index, the number of first-level and second-level blocks in the multi-level index
are respectively
A) 8 and 0
B) 128 and 6
C) 256 and 4
D) 512 and 5

Solution:

Number of records in file = 16384

Record size = 32 bytes
Key Size = 6 bytes
Block Size on file system = 1024 bytes
Size of Block Pointer = 10 bytes

Size of a record or index Entry = 10 + 6 = 16

Number of blocks in first level = (Number of records in file)/
(Disk Block Size)
= (16384 * 16)/(1024)
= 16 * 16
= 256

In second level, there will be 256 * 16 entries.
Number of blocks in second level = (Number of entries) /
(Block Size)
= (256 * 16) / 1024
= 4

© is correct.

18) In a database file structure, the search key field is 9 bytes long, the block size is 512 bytes, a record pointer is 7 bytes and a block pointer is 6 bytes. The largest possible order of a
non-leaf node in a B+ tree implementing this file structure is

Solution:
For B+ tree with order n, index pointer p, and block size = B
For a non leaf node it can be given that
n*p + (n-1)*(k) <= B
n*6 + (n-1)*9 <= 512
n <= 34.77 ie 34.

19) Consider a table T in a relational database with a key field K. A B-tree of order p is used as an access structure on K, where p denotes the maximum number of tree pointers in a B-tree
index node. Assume that K is 10 bytes long; disk block size is 512 bytes; each data pointer

PD is 8
bytes long and each block pointer PB is 5 bytes long. In order for each B-tree node to fit in a single
disk block, the maximum value of p is

Solution:

key field size = 10 bytes data pointer size = 8 bytes block pointer size = 5 bytes
(p – 1)(key field size + data pointer size) + p * block pointer size <= 512
23p – 18 <= 512
p <= 23

20) A B+-tree index is to be built on the Name attribute of the relation STUDENT.
Assume that all student names are of length 8 bytes, disk blocks are of size 512 bytes, and index
pointers are of size 4 bytes. Given this scenario, what would be the best choice of the degree (i.e. the
number of pointers per node) of the B+-tree

Solution:

Size of 1 record = 8 + 4 = 12

Let the order be N.

No. of index values per block = N - 1

(N - 1) 12 + 4 = 512

12N - 12 + 4 = 512

12N - 8 = 512

12N = 520

N = 520/12

N = 43.333 = 43.