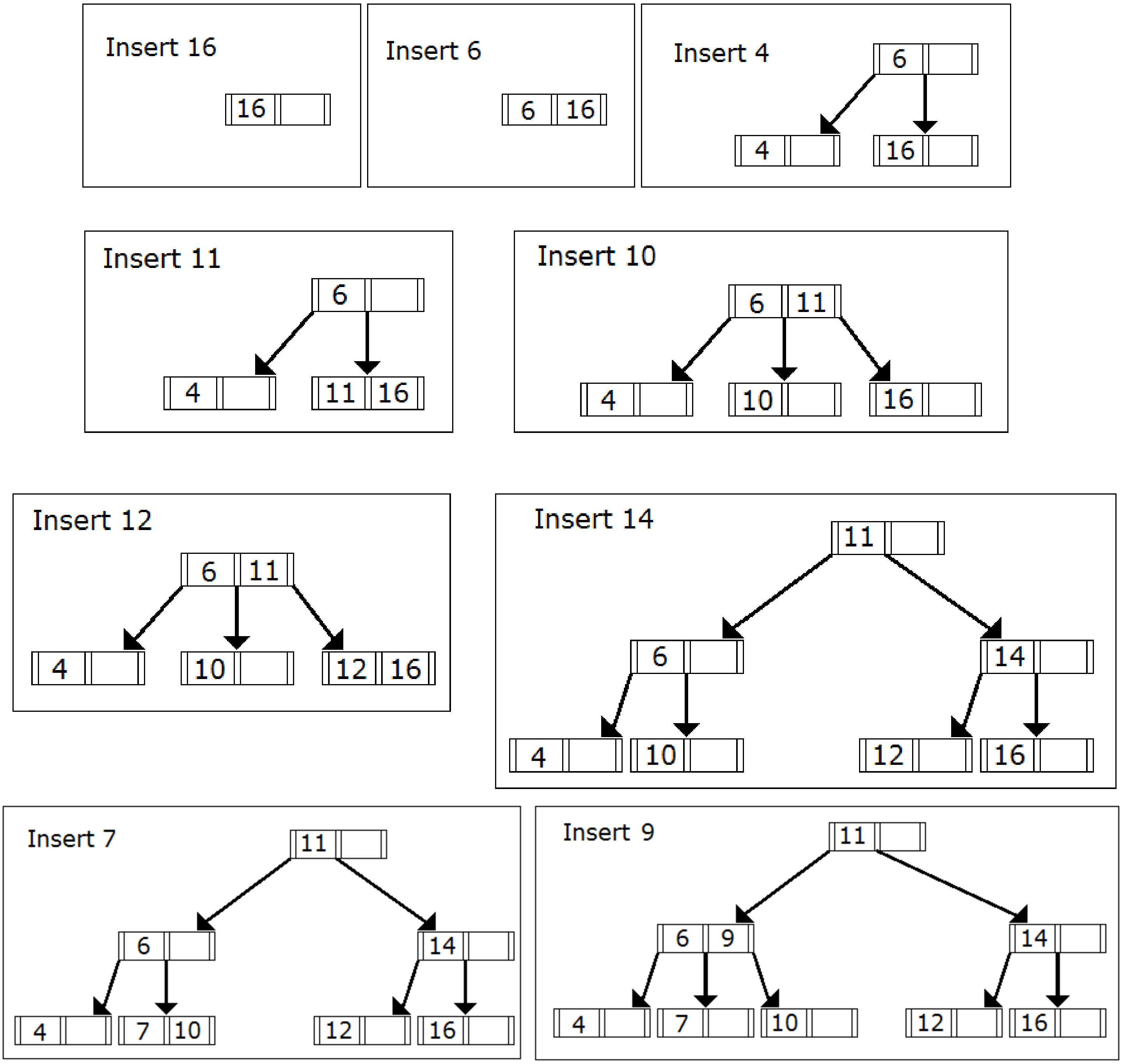
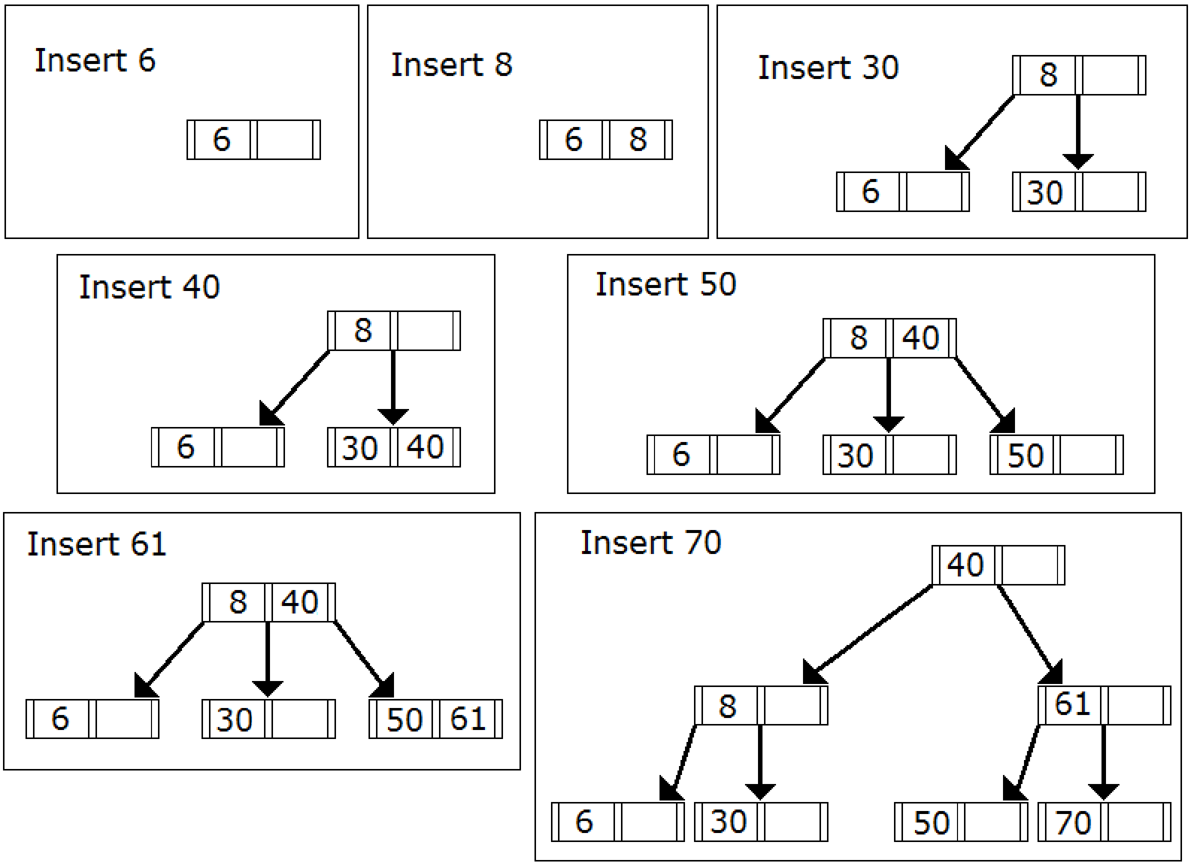
1) Degree d= 1, therefore max. no. of records per node is 2. (Insert 16, 6, 4, 11, 10, 12, 14, 7, 9.)

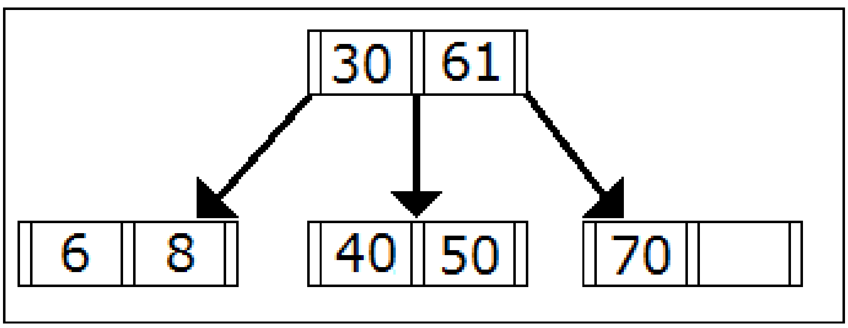


2) Degree (d) is 1.



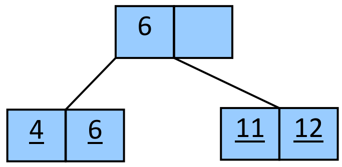
Number of nodes = 4

Minimum number of nodes can be obtained in a tree which has its every node %100 full (as much as possible). So; if we insert keys with the sequence 8, 30, 40, 6, 61, 70, 50; we obtain the tree as the following:

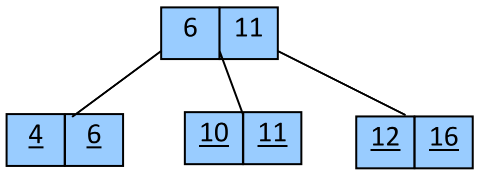


3) (6, 12, 4, 11, 10, 16, 14, 7, 9, 5, 15, 3, 1)

Note that data nodes are connected to each other (not shown in the following figure). After inserting the first four records we have the following.

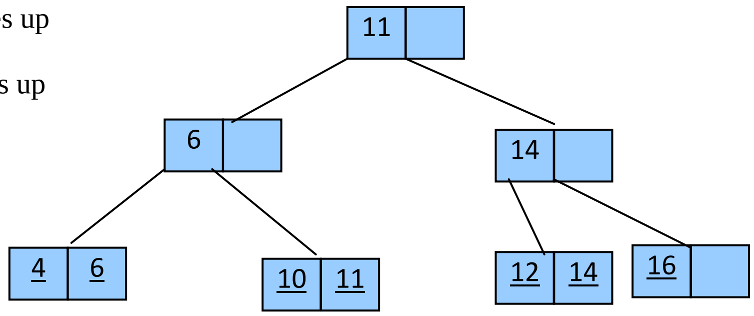


insert 10, 16=>(10, 11, 12) 11 goes up

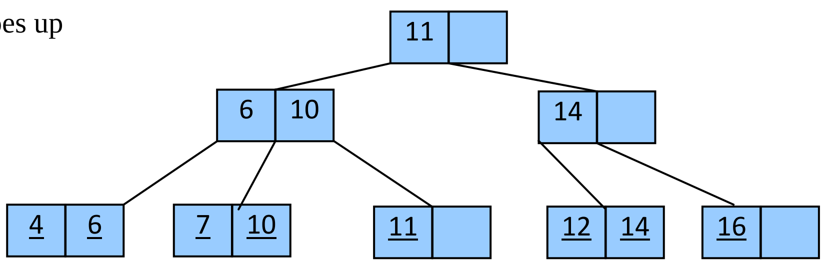


insert 14 => (12, 14, 16) 14 goes up

(6, 11, 14) 11 goes up

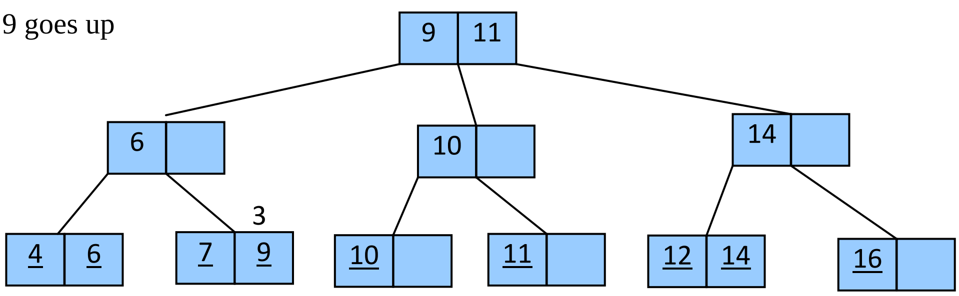


insert 7 => (7, 10 ,11) 10 goes up

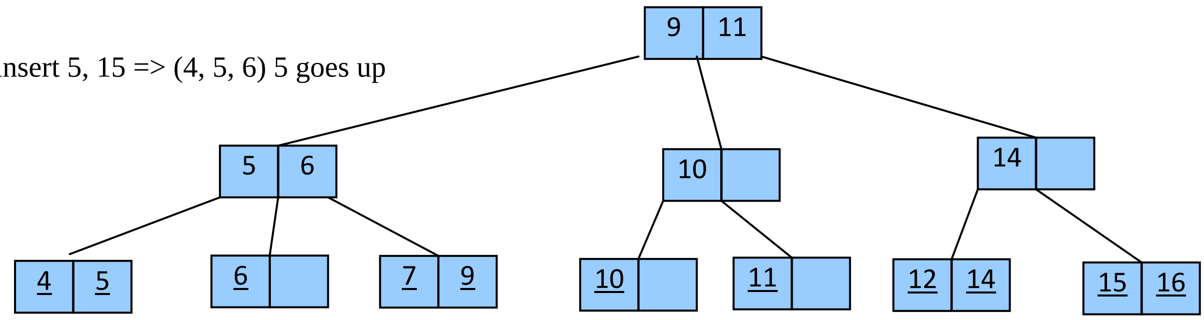


insert 9 => (7, 9, 10) 9 goes up

(6, 9 ,10) 9 goes up



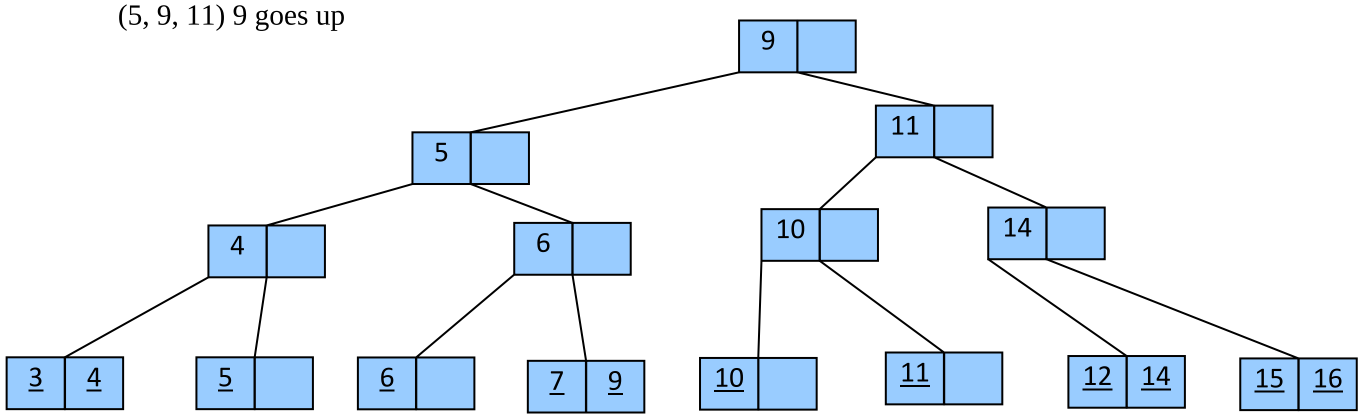
insert 5, 15 => (4, 5, 6) 5 goes up



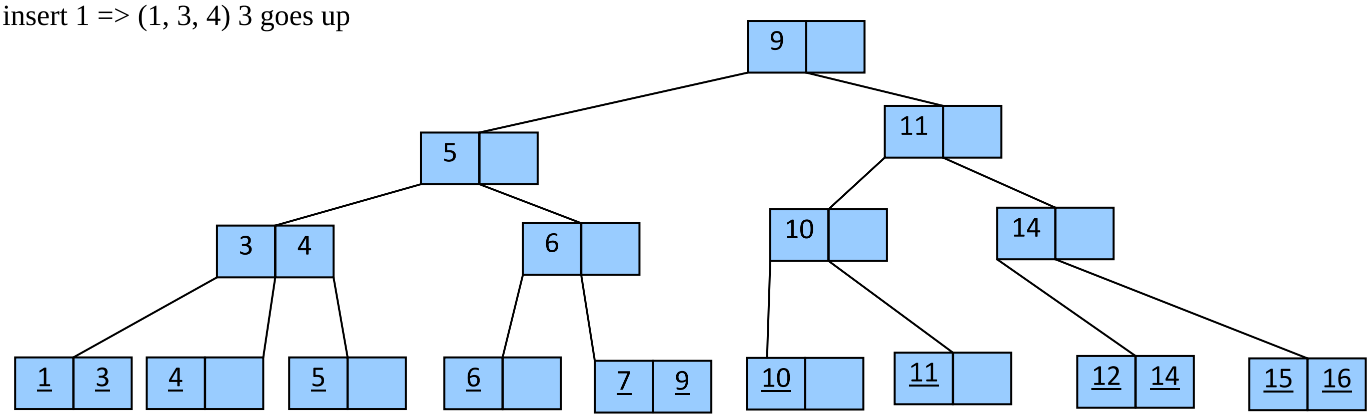
insert 3 => (3, 4, 5) 4 goes up

(4, 5, 6) 5 goes up

(5, 9, 11) 9 goes up



insert 1 => (1, 3, 4) 3 goes up



4) Minimum:

We would have 300 / 10 = 30 data nodes. We need 30 pointers to point to these data blocks. Hence the number of index nodes above the data nodes is Ceiling (30 / (2d+1)) = Ceiling (30 / 11) = 3

We also need to have a root node.

Thus the depth is 2 excluding the data level.

Maximum:

We would have 300 / 5 = 60 data nodes. We need 60 pointers to point to data blocks. Hence the number of index nodes above the data nodes Floor (60 / (d+1)) = Floor (60 / 6) = 10

We also need to have a root node.

Thus the depth is 2 excluding the data level.

5) Minimum number of data nodes to be accessed is 2, since 100 of these records can be stored in one data node and 2 of them can be stored in the adjacent data node. Maximum number of disk access can be 3 since the records to be accessed can be at the most in 3 different data nodes.

6)

a- B+ tree is more efficient when we need to process a range query or sequential access by its nature.

B+ tree index structure is dynamic and changes according to the needs.

b- Index structure of a ISAM is static and therefore we may have overflow records.

c- B+ tree gives you to get the range queries in easy and faster way with the connections between data nodes.

d- Sequential file provides a way to keep the data in less memory compared to B+tree since there is no index structure. It is suitable for applications that require entire file access without paying attention to ordering of records (e.g., for finding averages).

e- In B+ tree structure we can access records by following the links between the consecutive data nodes. In B tree we need to use inorder traversal which requires more disk accesses.

7)

We have Leaf node size = 2400 bytes   Available memory = 5 MB

R (record size) = 200 bytes   average fo = 50

bk: no. of data buckets

5×10^6 / 2400 = 2083 index blocks can be kept in the main memory

$2083 = bk / (fo)^2 = bk / 50^2 \Rightarrow bk = 5{,}207{,}500$

2400/200 = 12 Bkfr;   n(number of records) = bk x 0.7 x 12

So; we can have approximately n = 43,743,000 records.

In this calculation Salzberg (p. 151) assumes that we have three layers above the parent of leaf level.

8) Using binary search.

No. of records	Binary Search Pass No.
1,000,000	1
500,000	2
250,000	3
125,000	4
62,500	5
31,250	6
15,625	7
7812	8
3906	9
1953	10
976	11
488	12
244	13
122	14
61	15

After the 15^{th} pass, we do not need to make any disk access since the record is now in the desired bucket. So the time for fetching a record is as follows:

T_F = 15 * 10 ms = 150 ms (a)

Number of records	Pass no
10,000	1
5,000	2
2,500	3
1,250	4
625	5
312	6
156	7
78	8

In this part, we know that we search the record in a particular 10,000 records through the index structure. Therefore, we start to apply binary search in these 10,000 records. After 8^{th} pass, we do not need to make any disk access since the record is now in the desired bucket. So the time for fetching a record is as follows:

T_F = 8 * 10 ms = 80 ms (b)

Some solutions are from Anıl Yaman and Eren Gölge