HW #4

(mostly by) Salim Sarımurat

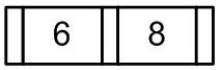
04.12.2009

S. 1.

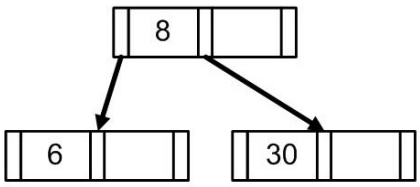
1. a.

1) Insert 6

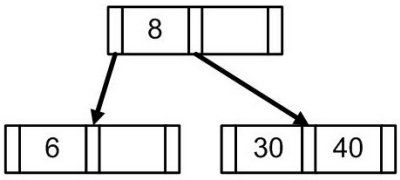
2) Insert 8



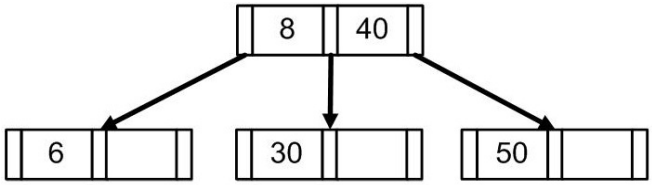
3) Insert 30



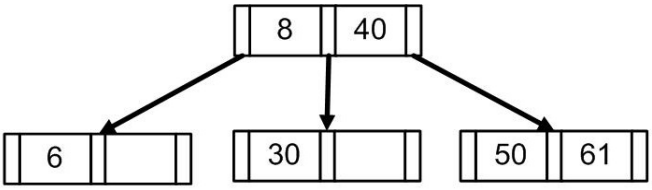
4) Insert 40



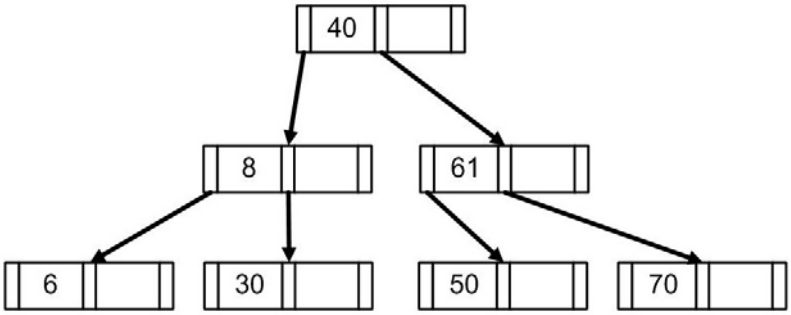
5) Insert 50



6) Insert 61

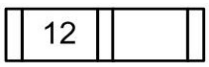


7) Insert 70

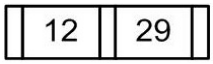


1. b.

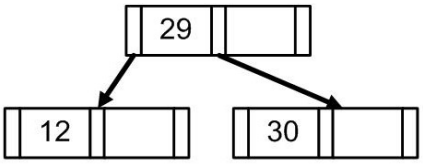
1) Insert 12



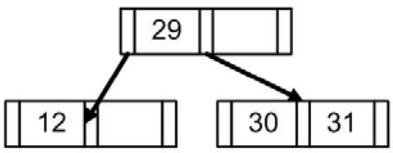
2) Insert 29



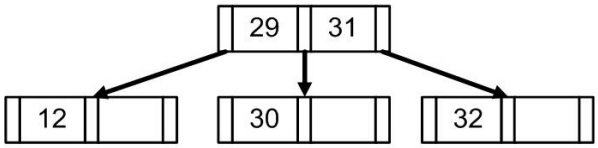
3) Insert 30



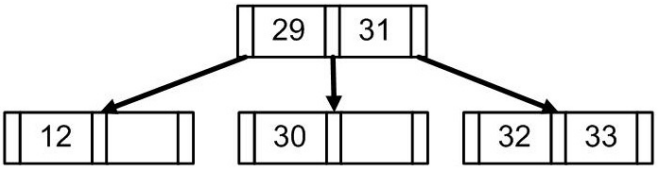
4) Insert 31



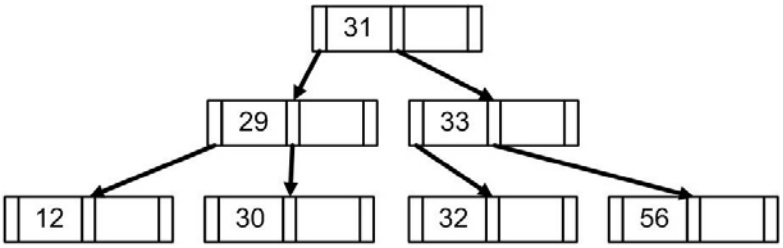
5) Insert 32



6) Insert 33



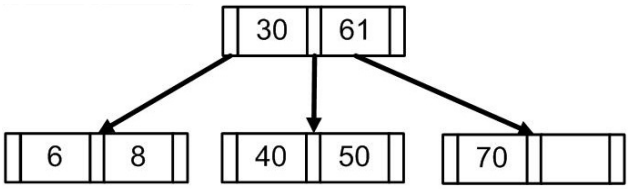
7) Insert 56



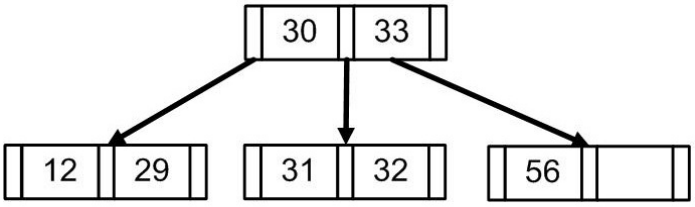
1. c. The distribution of the keys affects the level of depth of the B-tree created. And the level of depth of the B-tree affects the number of nodes that we have in this B-tree. These both – *the level of depth and the number of nodes of the B-tree* – are directly proportional to the retrieval performance of a record.

1. d. We will first need to order the given records. Since the order is given as one, we will group each 2 successive records in the order and make the following record as the root of that record recursively. For the following cases, the **bolded** records will be in the root of its left and right 2 records.

For part a, the order of records is: 6, 8, **30**, 40, 50, **61**, and 70. One possible order of insertion for these records is: 6, 70, **30**, 8, 40, **61**, and 50. And the created B-tree is;



For part b, the order of records is: 12, 29, **30**, 31, 32, **33**, and 56. One possible order of insertion for these records is: 12, 56, **30**, 29, 31, **33**, and 32. And the created B-tree is;



S. 2.

Assume that ***n*** is the number of records in the given file, ***O*** is the order of the B-tree and ***C*** is the maximum number of children a node can have in the B-tree that we will use to insert these records.

The minimum depth of the B-tree is calculated using the formula,

$$\lceil \log_C n \rceil$$

Because there will only be 1, O, O^2, O^3, O^4… in each level. This exponential increase suggests us using this way of calculation.

2. a. The maximum number of children that a B-tree of order 3 can have is 2 * 3 + 1 = 7

So; the minimum depth for a B-tree of order 3 is;

$$\lceil \log_7 100 \rceil = \lceil 2.36 \rceil = 3$$

2. b. The maximum number of children that a B-tree of order 10 can have is 2 * 10 + 1 = 21

So; the minimum depth for a B-tree of order 10 is;

$$\lceil \log_{21} 100 \rceil = \lceil 1.51 \rceil = 2$$

S. 3.

Assume the same definitions that we have introduced in the beginning of the 2nd solution.

The maximum depth of the B-tree is calculated using the formula,

$$\left\lceil \log_{C/2} n \right\rceil$$

The reason is based on explanation above and the fact that each node except the root must have a storage utilization of at least 50% of itself.

3. a. Using the values in the part **a** of the 2nd solution;

$$\left\lceil \log_{7/2} 100 \right\rceil = \left\lceil \log 100 / \log 3.5 \right\rceil = \lceil 3.67 \rceil = 4$$

3. b. Using the values in the part **a** of the 2nd solution;

$$\left\lceil \log_{21/2} 100 \right\rceil = \left\lceil \log 100 / \log 10.5 \right\rceil = \lceil 1.96 \rceil = 2$$

S. 4.

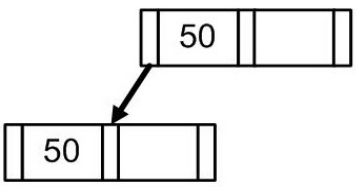
The maximum number of children that a B-tree of order 100 can have is 2 * 100 + 1 = 201

So; the minimum depth for a B-tree of order 100 is;

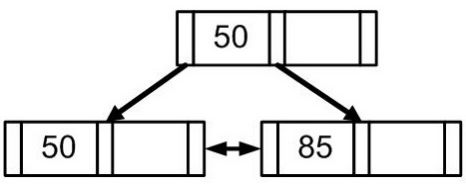
$$\lceil \log_{201} 45000 \rceil = \lceil 2.02 \rceil = 3$$

S. 5.

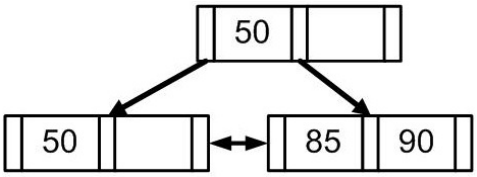
1) Insert 50



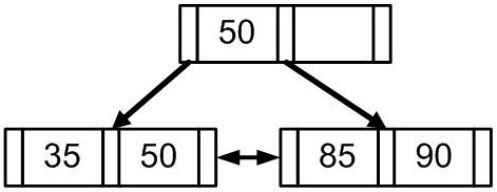
2) Insert 85



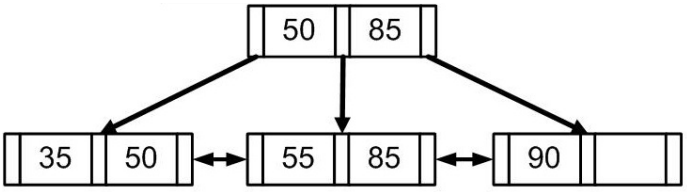
3) Insert 90



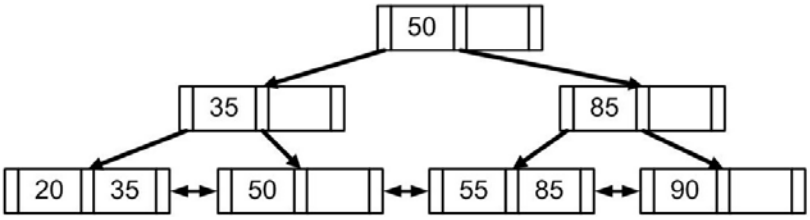
4) Insert 35



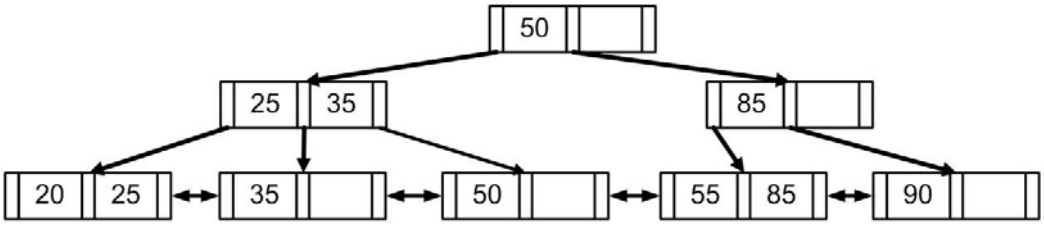
5) Insert 55



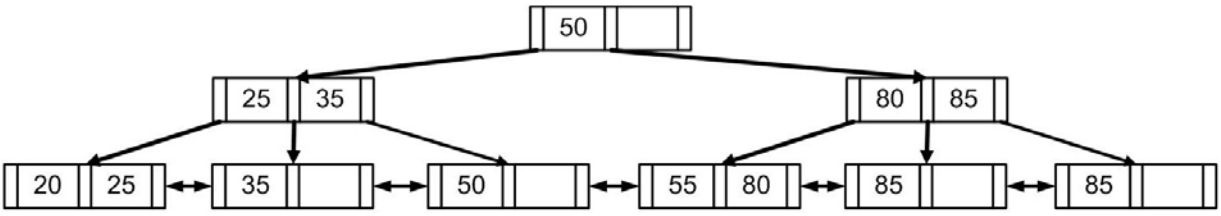
6) Insert 20



7) Insert 25



8) Insert 80



And the index set is; **25, 35, 50, 80, 85**

S. 6.

As we have index of order 7, we can have at most 7 * 2 + 1 = 15 links from each index node. So, we are going to have 1 root node, 15 index nodes in the following level, 15 * 15 = 225 nodes in the following level etc. This exponential increase – as in S #2 & S #3 – suggest us calculating the depth of the B-tree using the logarithm function. However, here we also consider that data nodes can hold up to 6 records. So, basically we will calculate the number of records that can be hold in the data nodes after calculating the number of levels in the deepest index nodes. If it is less than the given number of records, we will add one more index level until the total number index links in the deepest index nodes are equal to the total number of data nodes.

6. a. Minimum depth.

$$\left\lceil \log_{15}\left(100/6\right)\right\rceil = \left\lceil \frac{\log 100/6}{\log 15} \right\rceil = \lceil 1.038 \rceil = 2$$

Eventually, adding the data level 2 + 1 = **3** is the minimum depth of the B-tree.

Practical Solutions: For having the minimum depth all data nodes must be full as much as possible. Each data node can contain at the most 6 records. 100/6 ➔ implies 16 data nodes with 6 records and 1 data node with 4 records (6 x 16 + 1 x 4= 100). So we have 17 data nodes. Each index node can contain at the most 15 pointers (order of index nodes is 7 so they can contain 2x7= 14 key values and 15 pointers). To access 17 data nodes we need 2 index nodes. Having 2 index nodes means we need to have a root node pointing to these 2 index nodes. Therefore we have total of 2 index node layers + data node layer and hence total of 3 levels.

6. b. Maximum depth.

At least the 50% of the nodes in the B-tree must be filled with records. So each level will have ceiling of 15 * 50% – which is 8 – links to the level below itself.

$$\left\lceil \log_8 \left(100/3 \right) \right\rceil = \left\lceil \frac{\log 100/3}{\log 8} \right\rceil = \lceil 1.68 \rceil = 2$$

Eventually, adding the data level 2 + 1 = **3** is the maximum depth of the B-tree.

Practical Solution: For having the maximum depth each data node must be as empty as possible. However, data nodes must contain at least 3 records (they have to be 50% full). 100/3➔ 33 data nodes (33 x 3= 99, we have one extra record and it will be stored in one of these 33 data nodes, it cannot be stored in a data node by itself since remember that data nodes must be at least 50% full). Each index node contains at least 7 key values and therefore at least 8 pointers. 33/8 ➔ 4 index nodes (4x8= 32, we need to have one more pointer for the 33rd data node and that pointer must be stored in one of the 4 index nodes. To point to these 4 index nodes we need a separate root node (remember that the root does not need to be 50% full). So we have two layers of index nodes and one layer of data nodes and all together 3 levels.

S. 7.

Using the same methodology in S #7;

7. a.

$$\left\lceil \log_{15} \left(400/6 \right) \right\rceil = \left\lceil \frac{\log 400/6}{\log 15} \right\rceil = \lceil 1.55 \rceil = 2$$

Eventually, adding the data level 2 + 1 = **3** is the minimum depth of the B-tree.

&. b.

$$\left\lceil \log_8 \left(400/3 \right) \right\rceil = \left\lceil \frac{\log 400/3}{\log 8} \right\rceil = \lceil 2.019 \rceil = 3$$

Eventually, adding the data level 3 + 1 = **4** is the maximum depth of the B-tree.