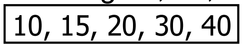
Implementation of DBMS
Exercise Sheet 8, Solutions
Klingemann, WS 2024 / 2025

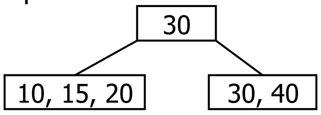
1) Insert the keys 20, 40, 10, 30, 15, 35, 7, 26, 18, 22, 5, 42, 13, 46, 27, 8, 32, 38, 24, 45, 25 in this order into an initially empty B+-tree of order 4.

Solution:

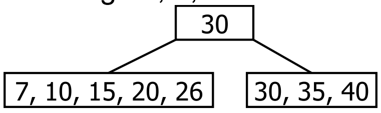
Inserting 20, 40, 10, 30, 15 → overflow:



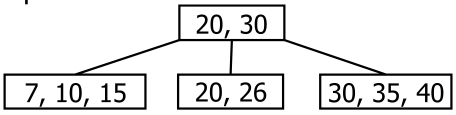
Split:



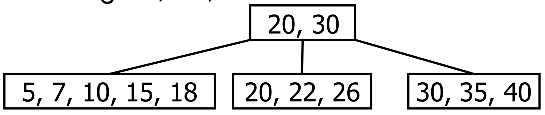
Inserting 35, 7, 26 → overflow:



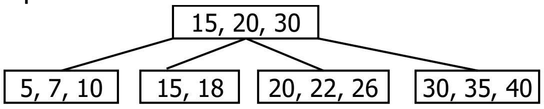
Split:



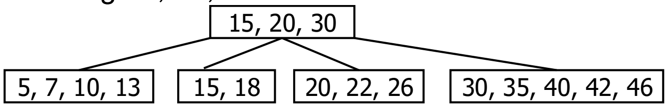
Inserting 18, 22, 5 → overflow:



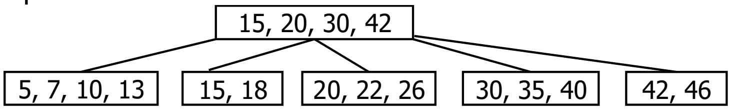
Split:



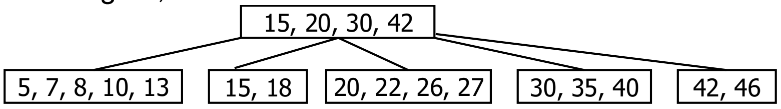
Inserting 42, 13, 46 → overflow:



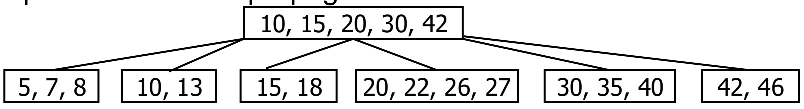
Split:



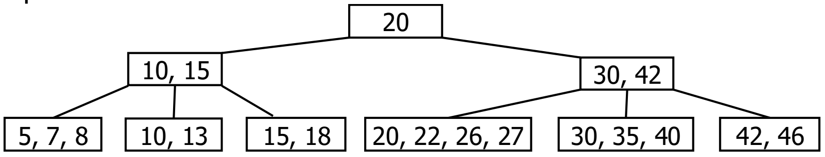
Inserting 27, 8 → overflow:



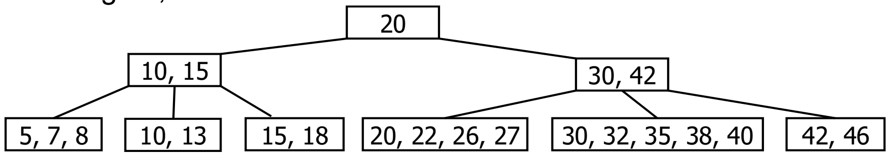
Split 1 → overflow propagates to the next level:



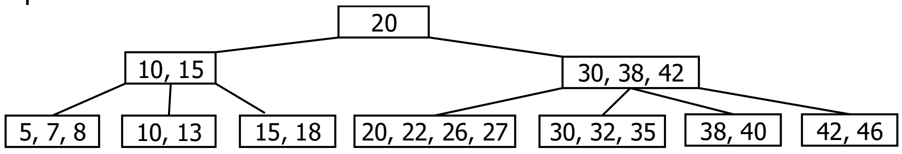
Split 2:



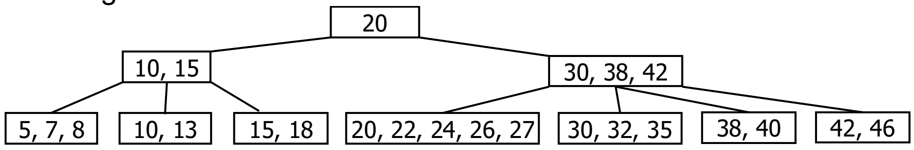
Inserting 32, 38 → overflow:



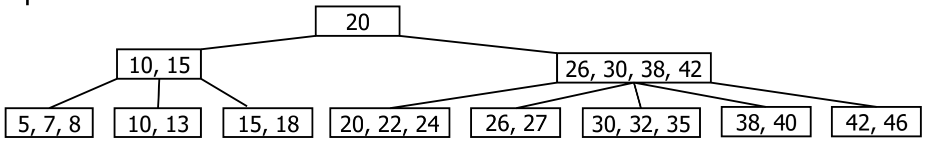
Split:



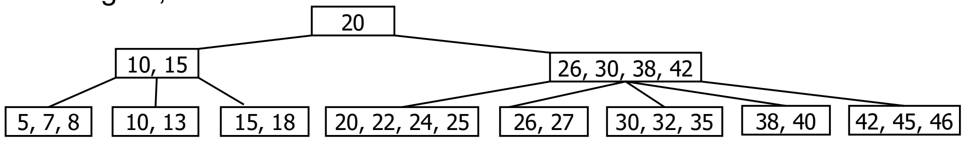
Inserting 24 → overflow:



Split:



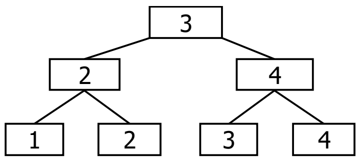
Inserting 45, 25:



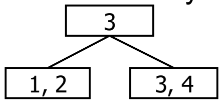
2) Consider B+-trees of order 2. Give an example of a B+-tree with three levels whose set of keys could alternatively be represented in a B+-tree with two levels. Your example should consist of two trees, one with three levels and another one with two levels but the same set of keys.

Solution:

In a B+-tree of order 2 each node can have either one or two keys. This allows different examples to be constructed. A simple approach to solve this task is to create a tree with three levels and the minimum number of keys. Such a tree could look like this:



These four keys can also fit in a tree with just two levels. Such a tree could look like this:



3) Suppose we have a B+-tree of order 3. We continuously insert the keys 1, 2, 3, ... into an initially empty tree. At the insertion of what key will the B+-tree first reach four levels?

Solution:

First of all, we can note that if we have just reached the fourth level, the last step is a split. When we split a leaf of a B+-tree of order 3, each of the resulting nodes gets 2 keys. Due to the order in which we insert, the left node will remain unchanged. When we split a non-leaf, the left node gets 2 keys and the right node gets 1 key. Again, the left node will not be affected by subsequent insertions. As a result, after a split we have a tree in which all leaves have exactly 2 keys. On each non-leaf level, the rightmost node has one key and all other ones have 2 keys. This allows us to derive the structure of the tree:

The 4th level consists just of the root node with a single key and 2 children.

Of the 2 nodes on the 3rd level the left one has 3 children and the right one has 2 children.

Of the 5 nodes on the 2nd level the first 4 ones have 3 children each and the rightmost one has 2 children.

This means that we have 14 leaf-nodes. As each of them has 2 keys, we have inserted 28 keys.