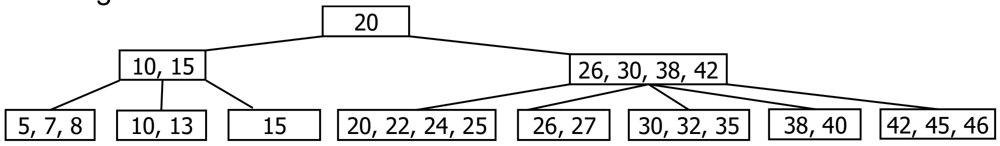
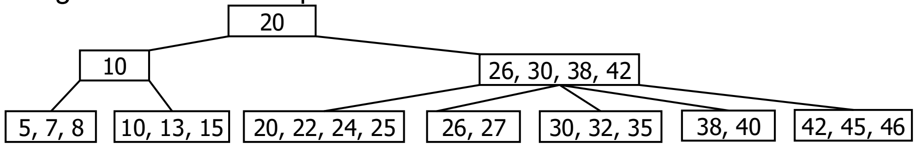
Implementation of DBMS
Exercise Sheet 9, Solutions
Klingemann, WS 2022 / 2023

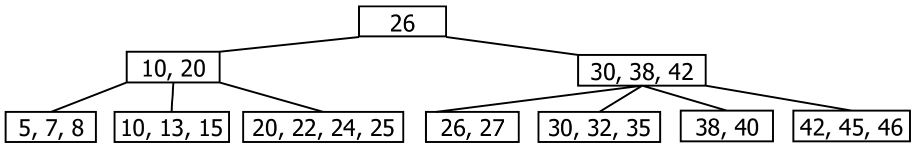
1) Delete from the B+-tree of order 4 you produced in Sheet 7, task 3 the key 18.
Solution:
Deleting 18 → underflow:



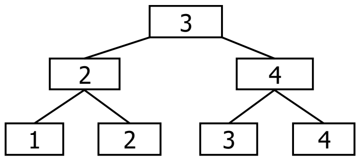
Merge → underflow in parent:



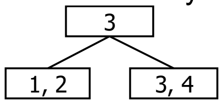
Transfer:



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 the equivalent one with two levels.
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:



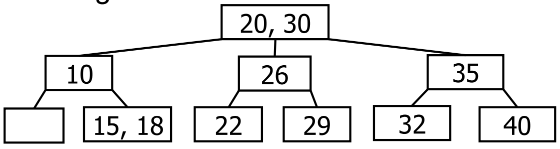
These four keys can also fit in a tree with just two levels. Such a tree could look like this:



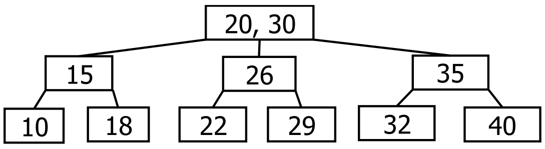
3) Delete from the B-tree of order 2 you produced in Sheet 8, task 3 the keys 7, 30, 10 in this order.

Solution:

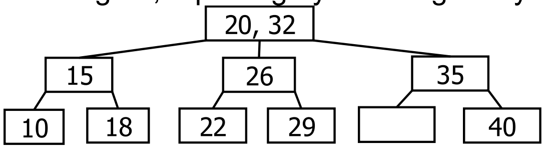
Deleting 7 → underflow:



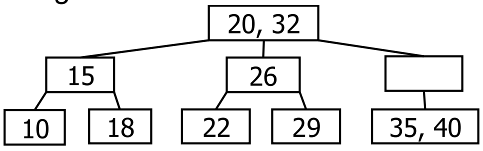
Transfer:



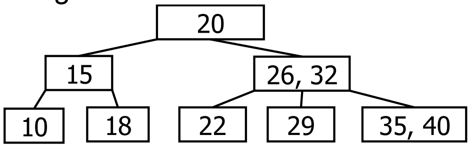
Deleting 30, replacing by next larger key in tree → underflow:



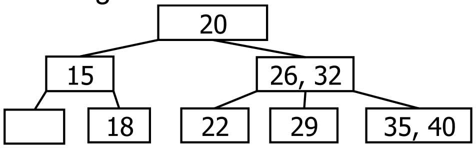
Merge 1:



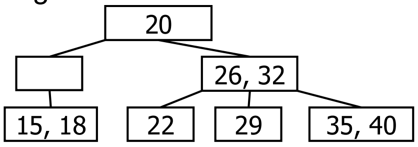
Merge 2:



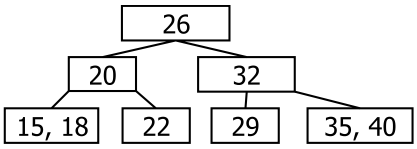
Deleting 10 → underflow:



Merge:



Transfer:



4) Suppose we have a file of 1,000,000 records that we want to hash into a table with 1000 buckets. 100 records will fit in a block, and we wish to keep blocks as full as possible, but not allow two buckets to share a block. Empty buckets do not consume a block. What is the minimum and maximum number of blocks that we could need to store this hash table? buckets vs. blocks

Solution:

In the best case, each bucket contains a number of records which is a multiple of hundred. In this case each block is completely full. For example, each bucket could contain 1000 records and therefore requires $\lceil$1000 records / 100 records per block$\rceil$ = 10 blocks. In total, we therefore need 10 blocks per bucket * 1000 buckets = 10000 blocks.

In the worst case, we have in each bucket a block which contains just a single record. For example, we can have 999 buckets with one record, each. Every bucket of this kind consumes one block. The last bucket contains the remaining records and needs $\lceil$999001 records / 100 records per block$\rceil$ = 9991 blocks. In total, we therefore need 999 blocks + 9991 blocks = 10990 blocks.

5) In an extensible hash table with n records per block, what is the probability that an overflow will have to be handled recursively, i.e., all members of the block will go into the same one of the two blocks created in the split?

Solution:

We split a block when we have n+1 records for this block. When we split a block, we consider an additional bit to decide to which of the two blocks a record will be assigned. When all n+1 records go into the same block, they have the same value for this bit. The value of this bit can be for all records either 0 (with a probability $(½)^{n+1}$) or 1 (also with a probability $(½)^{n+1}$). In total, the probability is $(½)^{n}$.