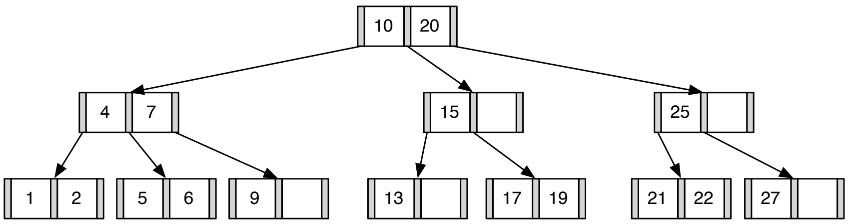
TU München, Fakultät für Informatik
Lehrstuhl III: Datenbanksysteme
Prof. Alfons Kemper, Ph.D.

Exercise for *Database System Concepts for Non-Computer Scientist* im WiSe 19/20

Alexander van Renen (renen@in.tum.de)
http://db.in.tum.de/teaching/ws1920/DBSandere/?lang=en

Sheet 12

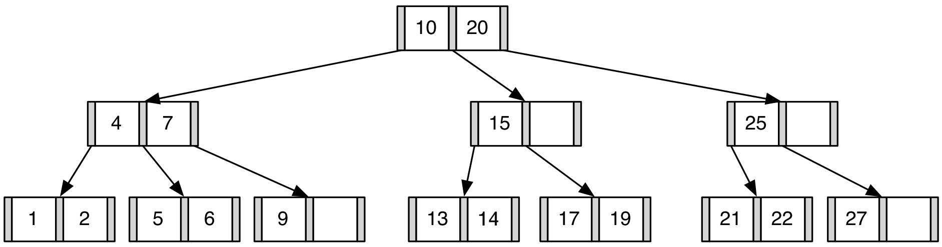
Exercise 1



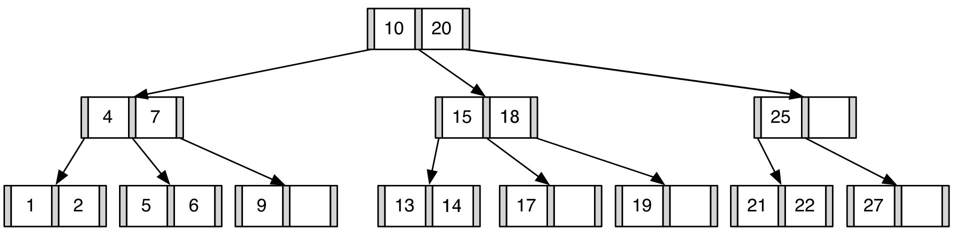
Insert 14, 18 and then 3 into the depicted B-Tree (degree $i = 1$).

Solution:

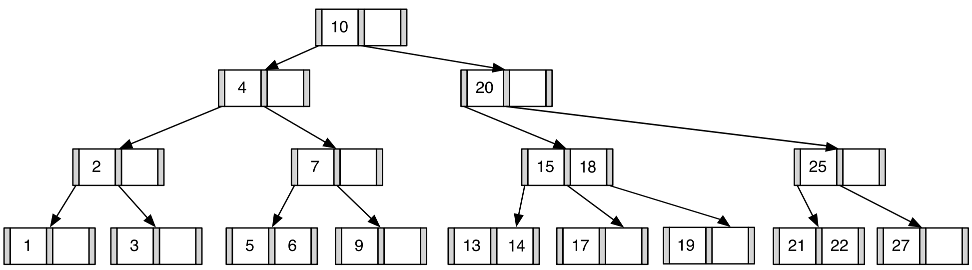
After inserting 14:



After inserting 18:



After inserting 3:



Exercise 2

Give a permutation of the numbers 1 to 24, such that when inserted into an empty B-Tree (degree $i = 2$) the height of the tree (number of layers) of the B-Tree is minimal. Draw the resulting tree.

Solution:

To be of minimal height. The resulting root of the tree must contain 5,10,15 and 20.
On possible option is the following:

- 1,2,5,6,7: a new root containing 5 is created
- 10,11,12: 5 and 10 are in the root node now
- 15,16,17: 1,10 and 15 are in the root node now
- 20,21,22: 1,10,15,20 are in the root node now
- Now, we can insert the remaining keys in an arbitrary order

Exercise 3

Calculate the optimal degree i and the number of required levels (also known as the "height" of the tree) for a B-Tree with the following properties:

- The B-Tree should store all humans currently living on earth (assume an even 10 billion).
- For each human we store the name, country and a unique identifier (100 Byte per human). The unique identifier will be used as the key an requires 8 Byte to store.
- The degree i of inner and leaf nodes may be different.
- Each node has to fit on a 16KB (16000 Byte) page.
- The page ids in the inner nodes require 8 Byte.
- This time (unlike in the lecture), we want to be precise: an inner node with n tuples requires $n + 1$ page ids to identify its children (in the lecture we simplifies this and assumed that a node with n tuples has n page ids).

Solution:

For leaf nodes, we simply have to store the tuples themselves and we can calculate the number of tuples fitting on a single leaf node as follows: leaf_size ÷ tuple_size = 16KB ÷ 100B = 160. The degree of a node is half of that: 80. Using this, we can calculate the number of leaf nodes required to store 10 billion human tuples: number_of_humans ÷ tuples_per_leaf = 10e9 ÷ 160 = 62500000.

<table>
<tr><td rowspan="2">V_0</td><td>S_1</td><td rowspan="2">V_1</td><td>S_2</td><td rowspan="2">V_2</td><td rowspan="2">...</td></tr>
<tr><td>D_1</td><td>D_2</td></tr>
</table>

Figure 1: Struktur eines B-Baum Knotens

Next we calculate how many separator keys (x) can fit on an inner node. From this we can derive the fan-out of an inner node (how many pages can be addresses by an inner node). Using the structure of an inner node (Figure 1), we can create the following formula:

$$
\begin{aligned}
x * (\text{key\_size} + \text{tuple\_size}) + (x+1) * \text{page\_id\_size} &\leq 16\text{KB} \\
x * (8\text{B} + 100\text{B}) + (x+1) * 8\text{B} &\leq 16\text{KB} \\
x * 108\text{B} + x * 8\text{B} + 8\text{B} &\leq 16\text{KB} \\
x * 116\text{B} &\leq 16\text{KB} - 8\text{B} \\
x &\leq (16\text{KB} - 8\text{B}) \div 116\text{B} \approx 137.86
\end{aligned}
$$

Hence, we can store 137 tuples in an inner node (because we need to round up, because we only store "complete" tuples) and can therefore address $137 + 1 = 138$ child pages. Therefore, a total of $62500000 \div 138 = 456205$ inner pages are required to store each page on the leaf level. But these 426205 pages need to be addresses as well ...

To figure out the height of the tree (number of layers, not counting the root), we can either continuously divide until there is only one page left: $62500000 \div 138 \div 138 \div 138 \div 138 = 0.17$ and see that there is one leaf level and four layers of inner nodes. Or, we can use a logarithm: $\log_{138}(62500000) = 3.64$ to derive the number of inner layers (rounded up: 4). In both cases we end up with 5 layers (4 inner, 1 leaf). Therefore, the tree has a height of 4, because the root does not count.