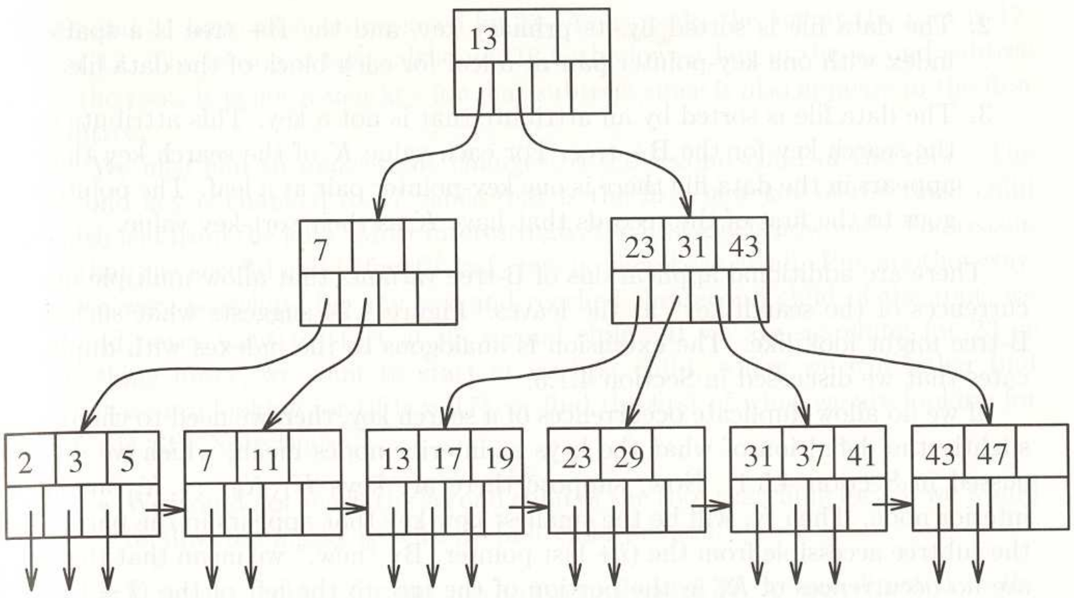
Examination Queries & Retrieval

Dec 6, 2010
9:00 – 10:45 BBL-001

- You may use a cheat sheet (A4)
- Mention your name and student number on every answer sheet
- Write and formulate clearly
- Always explain your answers
- Good luck

[1] B-trees

Below, you see a B+-tree.



1. Describe how the tree looks after deletion of node 7
2. This tree has only leaf level pointers from left to right. What is the advantage of maintaining also right to left pointers on the leaf level?
3. Suppose someone has the following claim. *B-trees should be fat, but according to the current rules, nodes may be only half filled. You could increase performance and decrease space usage by requiring a higher minimal node filling rate, e.g. 2/3 or 3/4.* Discuss this claim. Is it feasible? Does it make sense?

[2] join methods

We distinguish two variants of the hash-join method to process natural joins.
Suppose we have tables R and S with A as join attribute.

Symmetrical
We have buckets that may contain tuples both from R and S. Based on the value of A, all tuples of both R and S are hashed into the right bucket. Finally, we determine for each bucket the valid join-combinations of tuples from R and S and output them. Matching tuples are guaranteed to be in the same bucket.

Asymmetrical
One of the relations, typically the smallest, is, according to the value of A, hashed into buckets. This one is called the building relation. The other one (the probing relation) is scanned. Each tuple is compared to the tuples in the appropriate bucket. In stead of storing it there, the join results for this tuple are immediately determined and written to the output buffer.

[1] Compare these two results (efficiency, space usage). Are there typical circumstances that make one of the methods favorable?

The anti-join operator is defined as follows:

R **aj** S is the bag of tuples t in R such that there is no tuple in S that agrees with t on all attributes common to R and S. In other words: the **aj** selects the tuples of R that will *not* survive the natural join with S.

[2] Describe how the σ algebraically distributes over the **aj.**

[3] Describe how the anti-join could be calculated using the two hash methods mentioned above. Do the same analysis as in question [1].

[4] Describe how the anti-join could be calculated in a way that resembles sort-merge join and index-join.