CS 240 — Lecture 22

Bartosz Antczak *Instructor: Eric Schost* *March 28, 2017*

22.1 Finishing up Compression

22.1.1 Burrows-Wheeler Transform

Achieves the best compression of any algorithm we have seen (at least on English text). BWT proceeds in three steps:

- Place all cyclic shifts of some text S in a list L
- Sort the strings in L lexicographically
- C is the list of trailing characters of each string in L

What will this do?

The idea is, given C, we can generate the *first column* of the array by sorting (i.e., the original encoded string). The decoding algorithm:

- Make an array of A of tuples $(C[i], i)$
- Sort A by the characters, record integer in array N
- Set j to index of \$ in C and S to empty string
- Set $j = N[j]$ and append $C[j]$ to S
- Repeat Step 4 until $C[j] = \$$

22.2 External Memory

The final topic in CS 240 yo. Recall the memory hierarchy:

- Registers (very fast, very small)
- cache L1, L2
- Main memory
- External memory

Our general idea will involve adapting our algorithms to take the memory hierarchy into account, avoiding transfers as much as possible.
We store data in a *dictionary*, usually stored in a BST. A more efficient implementation of trees is a *2-3 Tree*

22.2.1 2-3 Trees

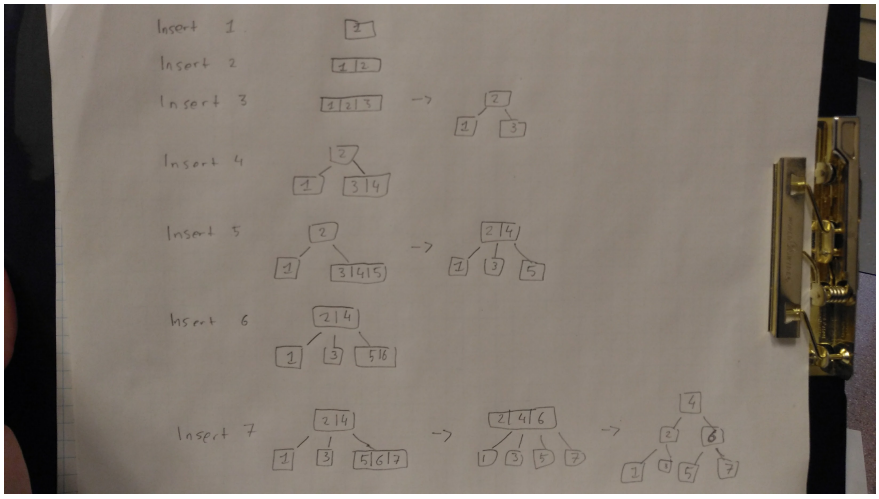
A 2-3 Tree is like a BST with additional structural properties:

- Every internal node either contains one KVP (Key-Value Pair) and two children, or two KVPs and three children (with two KVPs a and b, the children are structured as: $< a$, $a < \& < b$, $> b$)
- The leaves are NIL (do not store keys)
- All the leaves are at the same level

Searching through a 1-node is just like a BST. For a 2-node, we must examine both keys and follow the appropriate path.

Insertion in a 2-3 tree

First, we find the lowest internal node where the new key belongs. If the node has only 1 KVP, just add the new one to make a 2-node. Otherwise, order the three keys as $a < b < c$. Split the node into two 1-nodes, containing a and c, and (recursively) insert b into the parent along with the new link.



22.2.2 Deletion in a 2-3 tree

Please refer to the course slides for how the process occurs.

22.2.3 B-trees

A 2-3 Tree is a specific type of an $(a, b)-$tree. An **(a,b)-tree of order M** is a search tree satisfying:

- Each internal node has at least a children, unless it is the root. The root has at least 2 children
- Each internal node has at most b children
- If a node has k children, then it stores $k - 1$ key-value pairs
- Leaves store no keys and are at the same level