CS/ENGRI 172, Fall 2003: Computation, Information, and Intelligence
10/8/03: Information and Intelligence

From Computation To Information

> "In an alternative view of machine-intelligence development, personal computers are already so powerful that they are not the bottleneck problem at all. This is my view. ...
>
> Our human dependence on [knowledge] is very far-reaching. It comes into play with spoken and written language (as when we try to decipher someone's handwriting) and in our actions (e.g., when driving a car and deciding whether to brake or accelerate or swerve to avoid something). Before we let robotic chauffeurs drive around our streets, I'd want the automated driver to have general [knowledge] about the value of a cat versus a child versus a car bumper, ... about death being a very undesirable thing, and so on. That "and so on" obscures a massive amount of general knowledge of the everyday world without which no human or machine driver should be on the road, at least not near me [or] in any populated area."

— Douglas B. Lenat, "From *2001* to 2001: Common Sense and the Mind of Hal", in David G. Stork, ed., *HAL's Legacy:* 2001*'s Computer as Dream and Reality*, 1997.

Information Retrieval (IR)
The standard setting for information retrieval assumes as given:

- a *corpus*, or document collection, D consisting of n documents $d_1, d_2, \ldots, d_n$
- a *vocabulary* V of m distinct words $w_1, w_2, \ldots, w_m$

and assumes that the user provides a *query* expressing the user's information needs.

An information retrieval system outputs documents related to the query. How it *finds* and *presents* those documents, efficiently and accurately, are the primary problems with which IR is concerned.

Corpus indexing
Given the standard IR setting, we can use a *linear index* that contains all the vocabulary items in sorted order and that indicates, for each word w_i in V, at the least the following information:

- Those documents d_j that contain w_i,
- The location(s) of w_i in each such d_j.

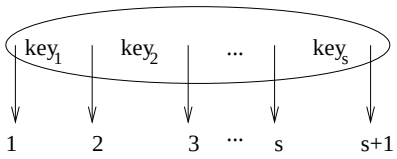
Indexing Example
d_1: Bill Gates of Microsoft spoke at yesterday's convention. We were kind of surprised at some of the predictions he made, but later on some other presentations clarified the situation. After all, the industry's followed these trends so far.

d_2: My friend Bill says weird versions of common proverbs. Just the other day, he said "Gates make for good neighbors." I also heard him say, "Microsoft wasn't built in a day", which is true, you have to admit.

B-trees

B-trees (*balanced multiway search trees*) help make the term lookup process more efficient. Conceptually, you can think of a B-tree as sitting "on top" of an index, with each leaf corresponding to the information for some single term in the index. (Strictly speaking, when the leaves are data items we have a B^+*-tree* rather than a B-tree, but we won't make this distinction.)

Every B-tree has some *order* (or *minimization factor*) t such that each internal node contains between t and $2t$ *keys* in sorted order, and the root has between 1 and $2t$ keys. If a node has $t \leq s \leq 2t$ keys, is has $s+1$ children. For example:



The keys give information about the leaves of the $s+1$ *subtrees*. In general, the ith child "covers" terms w such that $\text{key}_{i-1} \preceq w \prec \text{key}_i$. The 1st child "covers" terms w such that $w \prec \text{key}_1$, and the $s+1$st child "covers" terms w such that $\text{key}_s \preceq w$.

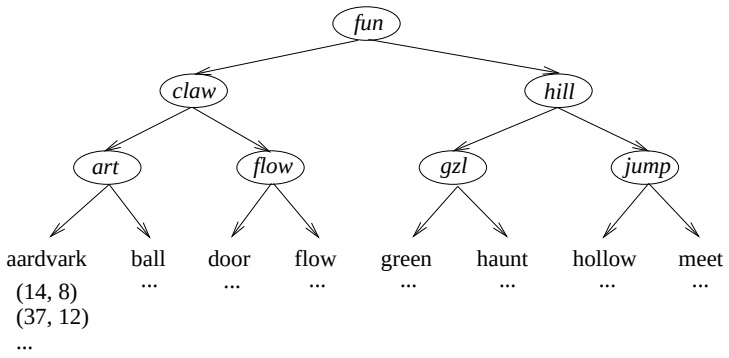
Finally, we require that every leaf of the B-tree have the same depth. A crucial fact about B-trees for our indices is that the depth of such B-trees can be at most roughly $\log_t(m)$.

B-Trees Examples

The following two B-trees organize the same index of terms. The B-trees can be thought of as overlayed on top of the index, with leaves storing the terms with the list of documents they appear in (and the term location in each document).

Example 1: Order 1

All leaves are at depth 3.



Example 2: Order 2

All leaves are at depth 2.