Computation, Information, and Intelligence (COMS/ENGRI/INFO/COGST 172), Fall 2005
9/26/05: **Lecture 14 aid – Information retrieval and indexing basics**

Agenda: information retrieval (IR) using Boolean keyword retrieval; we may get to B-trees. Note: the next few lectures are drawn from material in Frakes and Baeza-Yates's *Information Retrieval: Data Structures and Algorithms*, which is on reserve at the library in Carpenter Hall.

Follow-ups: PLA self-check: if you apply the PLA to our sequence of examples that showed that the NPLA can oscillate forever, you should find that $\overrightarrow{w}^{(3)} = \overrightarrow{w}^{(4)} = \left(1 - \frac{\sqrt{2}}{2}, 1 + \frac{\sqrt{2}}{2}\right)$.

I. Standard IR setting

We have a *corpus* D consisting of n documents $d_1, d_2, \ldots, d_n$ and a *vocabulary* V consisting of m distinct terms $w_1, w_2, \ldots, w_m$. The user expresses their information need via a *query* q.
We can build an *index* that contains all the vocabulary items in sorted order and that indicates, for each term w_i in V, at least the following information:

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

II. Two example documents

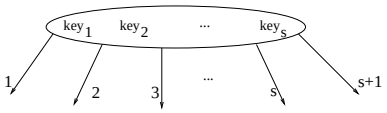
d_1: Bill Gates of Microsoft spoke at yesterday's convention. We were kind of surprised at some of the predictions he made, but later on some other presentations clarified the situation. After all, the industry's followed these trends so far.

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.

(OVER)

III. B-trees (a.k.a. *balanced multiway search trees*) 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.[1]

Every B-tree has some *order* (or *minimization factor*) t such that except for the root, each internal node contains between t and $2t$ (inclusive) *keys* in sorted order and has one more child than keys; schematically, this looks like the following:

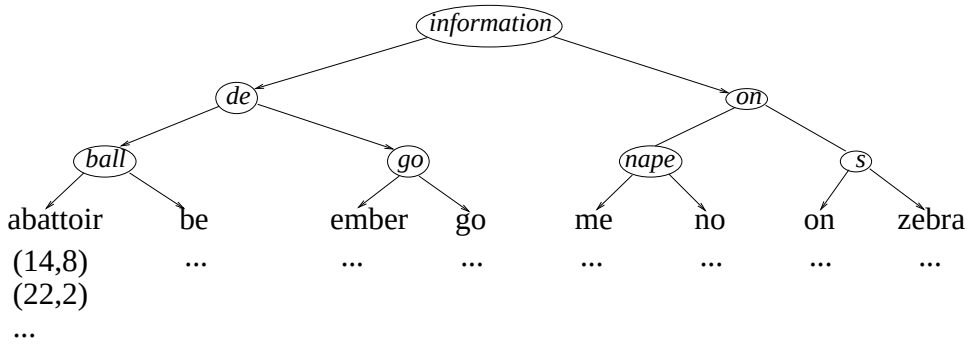


where s is some number such that $t \leq s \leq 2t$. (The root itself is an exception; it can have between 1 and $2t$ keys.) Different internal nodes need not contain the same number of keys. Every leaf is required to have the same depth.

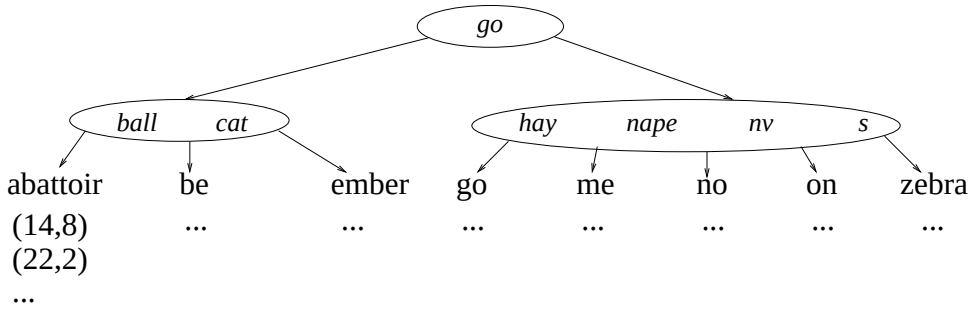
The keys give information about the leaves at the bottom of the $s+1$ *subtrees*. The i^{th} child "covers" terms w such that $\text{key}_{i-1} \preceq w \prec \text{key}_i$. The exceptions are the first child, which "covers" terms w such that $w \prec \text{key}_1$, and the $s+1$th child, which "covers" terms w such that $\text{key}_s \preceq w$.

The depth of an order-t B-tree is at most roughly $\log_t(m)$.

IV. Example B-trees Here is a B-tree of order 1. How long does it take to find the term "go"? How about "volition"?



This B-tree, of order 2, is for the same index as the previous B-tree.



[1]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.