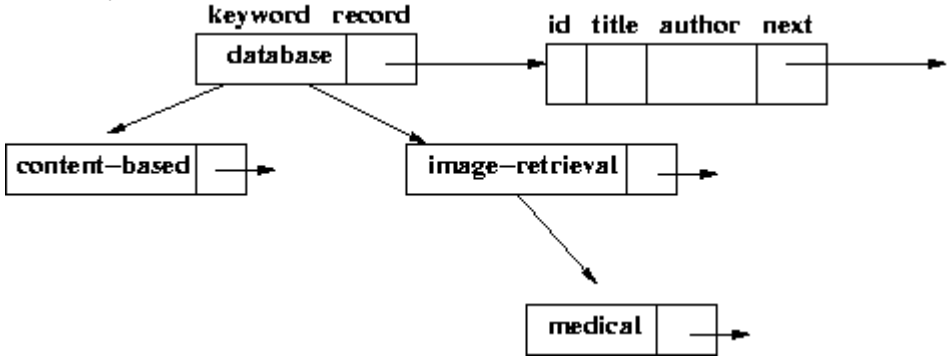
CSE 373 Spring 2013 HW3: AVL Trees

Background

Information retrieval systems allow users to enter keywords and retrieve articles that have those keywords associated with them. For example, once a student named Yi Li wrote a paper called, "Object Class Recognition using Images of Abstract Regions," and included the following keywords: 'object recognition', 'abstract regions', 'mixture models', and 'EM algorithm'. If someone does a search for all articles about the EM algorithm, this paper (and many others) will be retrieved.

Assignment

You are to implement an AVL tree and use it to store and retrieve articles. The tree will be sorted by `keyword`, and each node will contain an unordered linked list of `Record` objects which contain information about each article which corresponds to that `keyword`. This image shows the idea:



Getting started: The necessary files are available at `http://www.cs.washington.edu/education/courses/cse373/13sp/homework/hw03/hw3files.zip`. Included in this archive are:

- A **Data file** which contains records to be read into the data structure.
- Three **Java skeleton files**, which will, upon your completing them, convert the Data File into an AVL tree. See the section below which describes these files in detail.
- Five **Query files** which contain queries to be run on your data structure.

Required Functionality:

The required functions will appear as stubs with specific arguments in the code we provide. Note that some functions have extra requirements, stated below:

print() - a recursive function which outputs all keywords in alphabetical order along with the titles of articles for each keyword. The output should begin as follows:

```
blobs
    Region-Based Image Querying
buildings
    Consistent Line Clusters for Building Recognition
causal-relationships
    Mining Observational Databases for Causal Relationships
classification-rules
    A Theory of Learning Classification Rules
    Learning Classification Rules Using Neural Networks
clustering
    An Efficient Clustering Algorithm for Large Databases
    An Algorithm for Discovering Clusters in Large Spatial Databases
    Scaling Clustering Algorithms to Large Databases
    Consistent Line Clusters for Building Recognition
```

... and continue with the rest of the keywords and articles in the correct order. The keywords are not indented. Each title gets a separate line which begins with a "\t" (tab) character.

tree_stats() - this function will output the following stats about your data structure:

- the number of insertions that required no rebalancing
- the number of insertions that required only a single rotation
- the number of insertions that required a double rotation

get_records() - outputs the list of articles associated with a given keyword. The output must be formatted as in this example:

```
KEYWORD medical

47550
Knowledge-Based Image Retrieval with Spatial and Temporal Constraints
Wesley Chu

83528
Query by Example: the CANDID Approach
Paul Kelly

46359
A Content-Based Retrieval System for Medical Images
John Anderson
```

```
KEYWORD abstraction

NO TITLES FOUND
```

Note that "KEYWORD" is capitalized and the output of the records has the id, title, and author on separate lines. Please maintain this format so as to facilitate grading. If a keyword is not found, please print "NO TITLES FOUND" in all caps on its own line.

get_stats() - this function will read a file of keywords and compute the following stats about their retrieval:

- minimum number of nodes accessed over all keywords in the list
- maximum number of nodes accessed over all keywords in the list
- average number of nodes accessed over all keywords in the list

Note that the `get_stats()` and `get_records()` functions are located in the `test.java` file.

Skeleton file details:

The java skeleton files which you must edit are `test.java` and `avl.java`.
`Record.java` contains the data structure for storing the particulars about a single article.
The main script is run as `java test datafile.txt <command [query file]>` See the usage details with `java test -h`

Turn in:

Please provide the completed `avl.java` and `test.java` files. Also, please provide separate files containing the following:

- the output of `java test datafile.txt print` in a file called print.txt
- the output of `java test datafile.txt tree_stats` in a file called tree_stats.txt
- the output of `java test datafile.txt get_records gr_query.txt` in a file called get_records.txt
- the results of `java test datafile.txt get_stats query-N.txt` where N $\in \{1, 2, 3, 4\}$ (the provided files). Please put the results of each call in get_stats_N.txt for the respective N.

Write Up:

Please complete the following exercises and turn them in along with your the other files:

1. You have an empty AVL tree which sorts animals by size, and the following animals are input in order: ant; mouse; orca whale; raccoon; horse. Please submit a description of where each element is inserted and what, if any, rotations are triggered at each insertion. You must include drawings of the tree at every stages to help illustrate the process.

2. You have a binary tree whose nodes have an integer `data` field. Implement in psuedo-code a recursive function which would return the sum of these data fields.

3. Using at least a half-dozen complete English sentences, answer the following: How did you feel about this assignment? What parts were difficult for you? What was the most fun/satisfying part? Also, describe one bug you encountered during your implementing this data-structure and how you diagnosed it (If you programmed this perfectly on the first try, please describe your general bug-fixing procedure).

4. **Extra Credit:** Write a function which prints the tree in the following "lisp" format:

```
(key_of_root (<left subtree>)
             (<right subtree))

e.g.
(5 (3 (1 null null)(2 null null))
   (8 (6 null null)(10 null null)))
```