R·I·T	Rochester Institute of Technology Golisano College of Computing and Information Sciences School of Information

Lab 3 (4 points)
Binary Tree Indexing and Wildcard Search

Overview

This lab consists of two major tasks:

- Building a binary tree to index the term dictionary and using it to answer single term queries and conjunctive queries
- Using a binary search tree to perform wildcard queries.

Resources

- You should have read Chapters 2 and 3 of Introduction to Information Retrieval.
- Carefully read the lecture examples of weeks 5 and 6 and understand the technical details.
- Go over the lecture notes of weeks 5 and 6.
- The Queue.java file is provided in case it is needed (for the wildcard search).

Note: Make JavaDoc comments in your Java programs including Course #, Lab #, Your name, and main functional description of each method with @param & @return if applicable at the minimum.

Ref. http://www.oracle.com/technetwork/articles/java/index-137868.html

Task 1: Binary Tree Index Construction (2 points)

In this task, you need to construct the binary tree index and use it to answer basic queries:

1. Complete the following two methods in BinaryTree.java, using which you can insert nodes into a tree and search a term from the tree index.

```
/**
 * insert a node to a subtree
 * @param node root node of a subtree
 * @param iNode the node to be inserted into the subtree
 */
public void add(BTNode node, BTNode iNode)
{
        //TO BE COMPLETED
}
/**
 * Search a term in a subtree
 * @param n root node of a subtree
```

```
     * @param key a query term
     * @return tree nodes with term that match the query term or null if
    no match
     */
    public BTNode search(BTNode n, String key)
    {
            //TO BE COMPLETED
    }
```

2. Complete the constructor of BTreeIndex.java:

```
    /**
     * Construct binary search tree to store the term dictionary
     * @param docs List of input strings
     *
     */
    public BTreeIndex(String[] docs)
    {
        //TO BE COMPLETED
    }
```

3. Add test cases into the main method of BTreeIndex.java to test of using binary tree index to answer single term and conjunctive queries.

Task 2: Wildcard Query Processing (2 points)

In this task, you need to implement wildcard search using the binary tree index and design test cases to test the algorithm.

1. Complete the wildCardSearch method in BinaryTree.java

```
    /**
     * Do a wildcard search in a subtree
     * @param n the root node of a subtree
     * @param key a wild card term, e.g., war (terms like warehouse will
    be returned)
     * @return tree nodes that match the wild card
     */
    public ArrayList<BTNode> wildCardSearch(BTNode n, String key)
    {
            //TO BE COMPLETED
    }
```

2. Complete the wildCardSearch method in BTreeIndex.java

```
/**
 * @param wildcard the wildcard query, e.g., war (so that warehouse
 can be located)
 * @return a list of ids of documents that contain terms matching the
 wild card
 */
public ArrayList<Integer> wildCardSearch(String wildcard)
{
        //TO BE COMPLETED
}
```

3. Add test cases into the main method of BTreeIndex.java to test of using binary tree index to answer wildcard queries.

Submit your programs to a lab drop box in MyCourses by 11:59PM April 11th.