BTree Project Guide

Project workflow hints

- Get the Github repo created for the project and make sure the team has write or admin access and I have write access.
- Study the project description.
- Study the example codes, in particular:
    - Disk IO example: In particular, look at DiskReadWriteExample.java. It shows the implementation of an external binary search tree on disk.
    - Bitwise operators example: In particular, look at BitwiseShiftDemo.java for helpful sequence utility code.
    - SQLite example: A quick starter example on how to set up and use SQLite.
- Create the scrum stories and assign them to team members. Share with your instructor if they ask!
- What are the major stories:
    - Create an interface for *BTree* usage!
    - Develop some basic unit tests for BTree
    - Create a testing plan -- that is, how to get to testing against the provided sample output files as soon as possible.
    - Create the main *BTree* and *BTreeNode* (and supporting) classes
        - Convert pseudocode into code for BTree.java class, BTreeNode.java class (can be inner class). What methods go into those classes?
            - Note, no support for deletion is required!
        - Create a TreeObject class
        - Figure out how to do disk read and write -- this could be done in parallel with the development of the main pseudocode
    - Create the BTree from the given data
        - Write driver code *GeneBankCreateBtree.java* for reading Gene Bank gbk files and insert the data into BTree -- create an interface for BTree so development for this can go in parallel!
    - Common utility classes:
        - *GeneBankCreateBTreeArgs.java* - class to process the command line arguments
        - *SequenceUtils.java* - simple class that contains methods for encoding, decoding DNA sequences and any other supporting methods
        - *ParseArgumentException.java* - given
        - *ParseArgumentUtils* - if needed
    - Searching the BTree
        - Create the driver code *GeneBankSearchBtree.java* with supporting classes as needed.
    - Creating and searching the Database of sequence and frequencies

        - Study the SQLite example to learn how to use it
        - Modify *GeneBankCreateBtree.java* so that it dumps the sequences and frequencies into a SQL database
        - Create and test the driver code GeneBankSearchDatabase.java
    - Adding the Cache from project 1
        - It will be a Cache of BTreeNode objects. Add Cache code from Project 1 to this project.
        - Modify *GeneBankCreateBtree.java* to use the Cache.
        - Test for correctness and performance improvement.
    - Running the major test scripts provided by the instructors and debugging issues as they arise. Note that you may have to adjust the scripts to make them work for your setup.
    - Stretch goal: Try running for the human Y chromosome.

Additional technical hints

1. The BTree is stored in a binary data file on disk.

2. The BTree data file will have an initial metadata section. The metadata section should contain at least the byte offset of the root node. It may also optionally contain the degree of the BTree and the number of nodes. After the metadata, the rest of the file consists of BTreeNodes laid out one after the other. A new node is added to the end of the file.

3. We will read/write one BTreeNode at a time. If the degree t is small, this would be inefficient. In a real-life case we would set the degree t such that a BTreeNode fits one disk block (we are using 4096 bytes for that) as close as possible with some empty padding space at the end (if needed).

4. We can store the byte offset of a node on disk as the child pointers in the BTreeNodes. Note that we never need real child pointers in memory.

5. We will use RandomAccessFile and FileChannel classes to read/write to the BTree data file. This allows us to quickly set the file cursor to anywhere in the file in O(1) time using the position(long pos) method. We will use the ByteBuffer class to read/write to the BTree data file. Please see the example of writing to a random access binary data file shown in DskReadWrite.java in the disk-IO-examples folder in CS321-resources repo.