

Trees Assignment Part #2

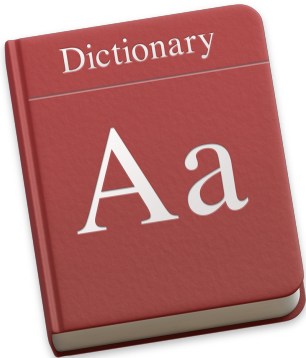
Re-submit Assignment

Due May 16, 2017 by 11:59pm **Points** 100 **Submitting** a file upload **File Types** zip

Overview

For the second part of this two-part assignment we will be building a dictionary based on a balanced binary search tree. We will then unit test our dictionary and present it to a user on the Java Console.

Note: This is a continuation of the following assignment: [Trees Assignment Part #1](#)



Dictionary API

Build a dictionary class that stores word-definition pairs in an instance of your symbol table from part #1 (i.e. you should not be using any other map classes as part of your solution here). The Dictionary class should have the following structure

```
public class Dictionary
{
    private BSTSymbolTable<String, String> words;

    //stuff goes here...
}
```

The dictionary class follows the API described below:

public Dictionary()	A default constructor that creates a dictionary containing zero words.
public Dictionary(String[] words, String[] definitions)	<p>Creates a new dictionary with the word/definition pairs given. The tree should be perfectly balanced using the algorithm described in the "Balancing Our Tree" section below.</p> <p>This method throws an IllegalArgumentException if the size of the both input arrays differ.</p> <p>This method throws an IllegalStateException if the elements of the words array are not in sorted order.</p>
public boolean updateDictionary(String word, String definition)	Adds a word/definition pair to the dictionary if the input word is not presently in the dictionary. Otherwise, the method will update the definition for an already existing word.
public boolean hasWord(String word)	Returns true if the input word is in the dictionary, otherwise false.
public String define(String word)	Returns the definition of the input word from the dictionary.

	This method throws a NoSuchElementException if the word is not found in the dictionary.
public List<String> words()	Returns an ordered list of words from the dictionary. There should be no duplicates in the result list.
public List<String> definitions()	Returns an unordered list of definitions from the dictionary.
public int size()	Returns the number of words in the dictionary.
public boolean isEmpty()	Returns true if the dictionary has no words, otherwise false.
public void clear()	Removes all words from the dictionary.

Note: Any students submitting solutions with word/definition pairs stored in any other data structure besides your BSTSymbolTable class from part #1 will be forced to resubmit your assignments.

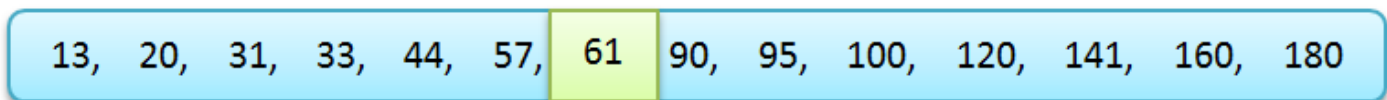
Balancing Our Tree

Your second constructor in the Dictionary class should generate a perfectly balanced tree by carefully choosing an insert order for the underlying symbol table BST. We will use the following algorithm to ensure these results:

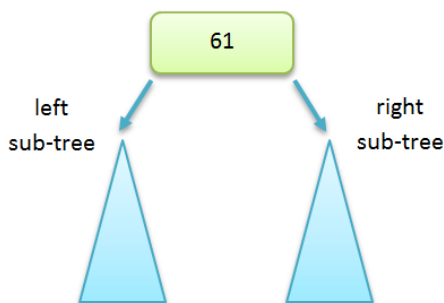
- Step #1: Sort your word/definition pairs by the lexicographic order of each word
- Step #2: Choose the middle element in your input array and add it to the symbol table
 - *Note: this ensures that half the dictionary elements will be located in the left sub-tree and half in the right sub-tree*
- Step #3: recursively choose a middle element of the left sub-tree and the middle element of the right sub-tree.
 - Continue with Step #2 for each sub-tree.
 - You will need to keep track of a low & high pointer that identify what portion of your input array is being considered at a moment.
 - Your base case will be when low & high are out of order (i.e. when to stop recursing).

Visualizing the Algorithm

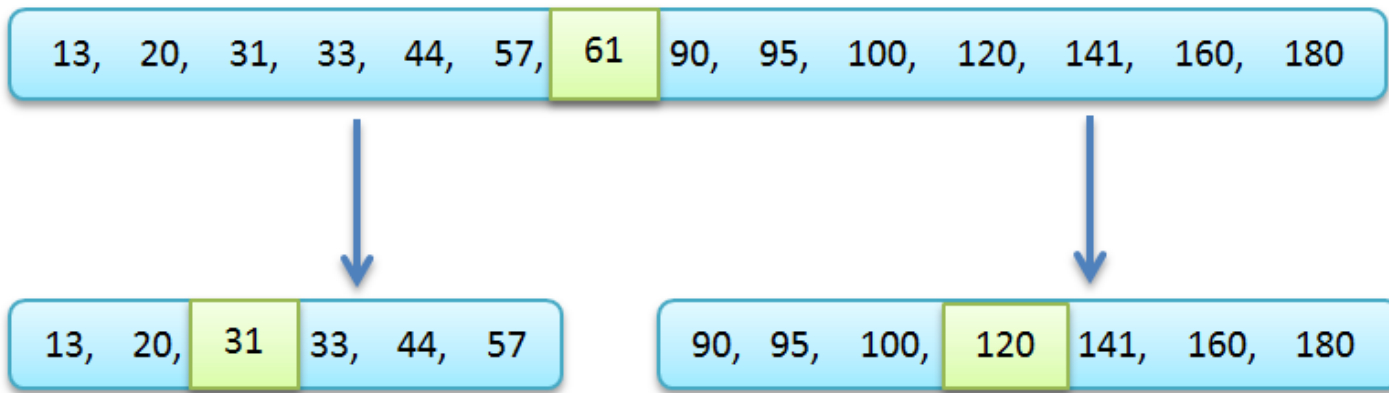
First select your middle element:



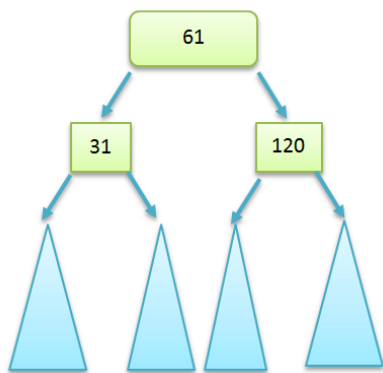
Adding this to the BST creates a tree rooted at 61:



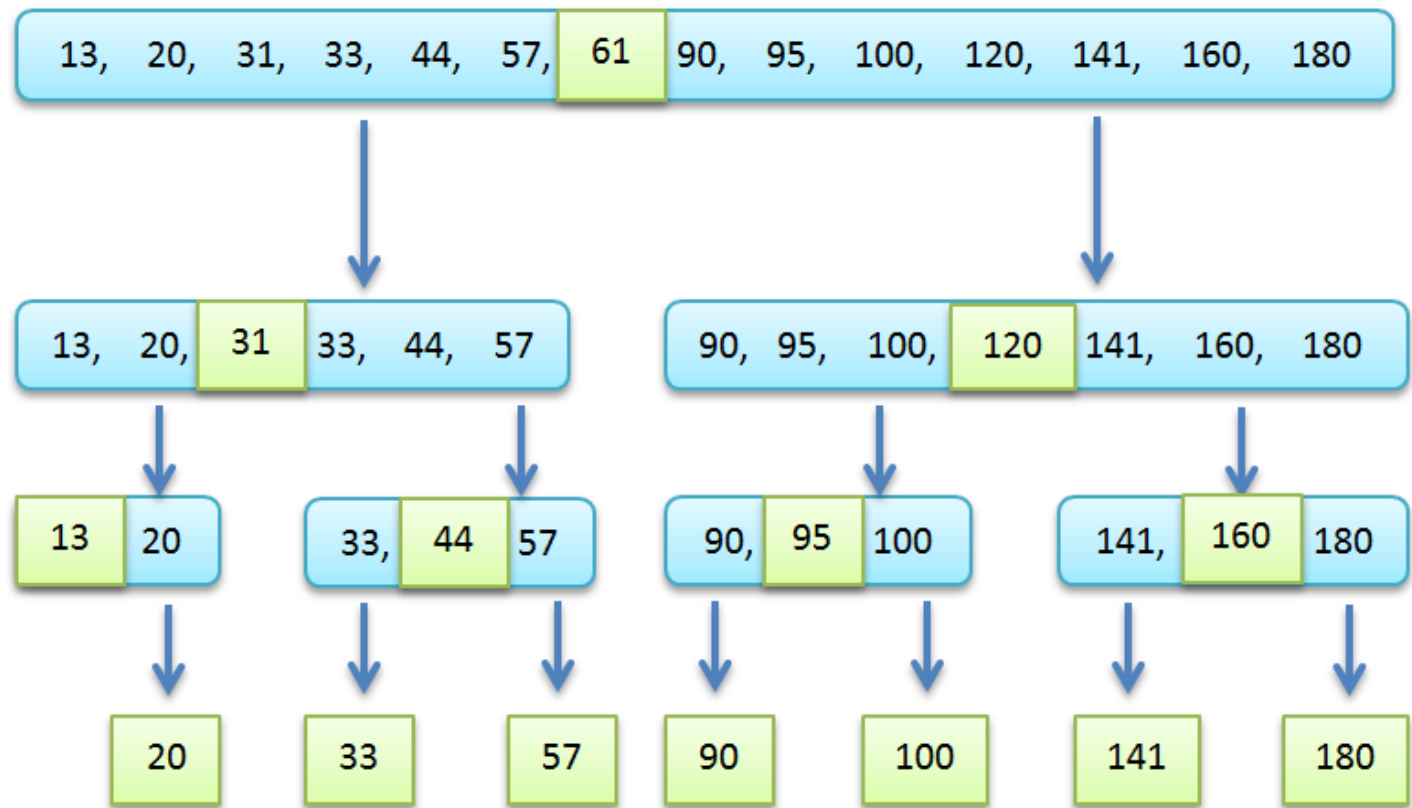
You should then recursively select and add the middle element in the left & right sub-tree:



Afterward your tree will look like the following:



This recursion should continue until you reach your base cases:




You are required to submit a set of unit tests for part #2 of this assignment. Your unit tests should be comprehensive and include all the methods of the Dictionary API:

- public Dictionary()
- public Dictionary(String[] words, String[] definitions)
- public boolean updateDictionary(String word, String definition)
- public boolean hasWord(String word)
- public String define(String word)
- public List<String> words()
- public List<String> definitions()
- public int size()
- public boolean isEmpty()
- public void clear()

Note: Be careful to write your tests against the method descriptions above.

Note: You should be testing different types of inputs, return values and exceptions that should be thrown.

Console Program

The last step of this assignment is to provide the user with a console program that allows them to interact with your dictionary. Add the following text file to your Eclipse project: [dictionary.txt](#) . This file contains 80000+ word definition pairs. Familiarize yourself with the file format.

You should first provide the user with the following menu:

1. Load dictionary
2. Lookup word
3. Exit

Load Dictionary

When a user selects this option, your program should read in all word/definition pairs from dictionary.txt into two String[] arrays (one for words, one for definitions).

You should then instantiate a new Dictionary object using your second constructor. This will create a new balanced tree with all 80000+ elements in the dictionary.

Lookup Word

This choice will first prompt the user to enter a word:

Please enter a word: apple

The console program should then query the dictionary object to see if a definition is present. If a definition is present it should be printed to the console:

Definition: Any tree genus Pyrus which has the stalk sunken into the base of the fruit;
an apple tree.

If no definition is present the program should ask the user for a definition and update the dictionary with a new word/definition pair.

Please enter a definition for "apple": A red or green tasty fruit!

Exit

After each menu choice the user should be presented with the main menu again. If the user chooses the "Exit" option, the program should terminate.

Thanks for viewing my dictionary!

Submission

Submit an eclipse project with:

- Your Dictionary, BSTSymbolTable & BinarySearchTree classes
- Your unit tests of the Dictionary class
- Your console program

Trees Assignment Part #2 Rubric

Criteria	Ratings		Pts
The updateDictionary() method adds or updates a word/definition pair in the dictionary	5.0 pts Full Marks	0.0 pts No Marks	5.0 pts
The hasWord() and define() methods allow you to search for entries in the dictionary according to the API descriptions	5.0 pts Full Marks	0.0 pts No Marks	5.0 pts
The words() and definitions() methods return an ordered list of words or unordered list of definitions.	5.0 pts Full Marks	0.0 pts No Marks	5.0 pts
The size(), isEmpty() and clear() methods correctly track/remove elements in the dictionary.	5.0 pts Full Marks	0.0 pts No Marks	5.0 pts
The second constructor in the dictionary API verifies that it has correct inputs and then recursively balances the tree of dictionary entries.	15.0 pts Full Marks	0.0 pts No Marks	15.0 pts
Unit tests are included for the updateDictionary() method and correctly test the API description.	5.0 pts Full Marks	0.0 pts No Marks	5.0 pts
Unit tests are included for the hasWord() and define() methods and correctly test the API description.	5.0 pts Full Marks	0.0 pts No Marks	5.0 pts
Unit tests are included for the words() and definitions() methods and correctly test the API description.	5.0 pts Full Marks	0.0 pts No Marks	5.0 pts
Unit tests are included for the size(), isEmpty() and clear() methods and correctly test the API description.	5.0 pts Full Marks	0.0 pts No Marks	5.0 pts
Unit tests are included for both constructors in the Dictionary class and correctly test the API description.	5.0 pts Full Marks	0.0 pts No Marks	5.0 pts
Your console program correctly loads the input file into your Dictionary class.	5.0 pts Full Marks	0.0 pts No Marks	5.0 pts
Your console program gives the user access to the menus and dictionary as described above.	5.0 pts Full Marks	0.0 pts No Marks	5.0 pts
Styles: naming conventions, brace placement, spacing, commented code, redundancy, packages and magic numbers.	20.0 pts Full Marks	0.0 pts No Marks	20.0 pts
Formal documentation: full Javadocs & comment block header.	10.0 pts Full Marks	0.0 pts No Marks	10.0 pts
Total Points: 100.0			