CSE 373: Homework 3

Dictionaries and Testing

Due: April 19th, 11:59 PM to Canvas

Introduction

In this assignment, you will implement the Dictionary ADT (represented by the Dictionary.java interface). You will do this by creating five separate implementations that use the interface: unsorted linked list, unsorted array, sorted linked list, sorted array and binary search tree. Instead of creating a dictionary suite that tests functionality, you will be asked to create a report which shows the runtimes off the three dictionary functions across the five different implementations.

You may only import Java.util.Random for this assignment

Importing into Eclipse

Since there are no .class files, importing into eclipse will be very easy. Download the HW3Eclipse.zip file from the website. In eclipse, select File → Import → Existing Projects into Workspace. There, choose Select Archive File and select the .zip file. Make sure HW3 is selected and click finish.

If you choose not to use eclipse, the relevant java files are available in HW3StarterCode.zip.

1 Implementing Dictionaries

In this part of the assignment, you will implement five different Dictionaries with int keys and String values. We have given you an interface, Dictionary.java and some starter code for your implementations. *Do not modify the Dictionary interface.* For simplicity, you may assume that keys inserted into the heap are unique, but the Dictionaries should support duplicate values. You may also assume that the values are non-null. You may create helper functions as needed. These functions will be tested by our test suites. Write your own test code as you find helpful.

There are three functions that we expect you to implement in your dictionaries:

- `void insert(int key, String value)`: This function inserts a key, value pair into the dictionary
- `String find(int key)`: This function returns the value associated with the given key. If the key does not exist in the dictionary, it should return null.
- `boolean delete(int key)`: This function should remove the key, value pair associated with the key given. It should return true if successful and false if the key was not in the dictionary.

The five implementations of the dictionary are given as follows. *For the array implementations, your array should automatically resize*:

- `Unsorted linked list`: indicated by the ULLDict.java file
- `Unsorted array`: indicated by the UADict.java file
- `Sorted linked list`: indicated by the SLLDict.java file
- `Sorted array`: indicated by the SADict.java file
- `Binary search tree`: indicated by the BSTDict.java file.

2 Writeup

You will need to create your own test suite to determine the runtimes of the three functions across all five implementations. You do not have to submit your test suite. Only submit the graphs and data from your results. Your writeup should have the following three graphs at a minimum. Each of these graphs should have at least 5 points along the x-axis corresponding to 5 different input sizes. These values should be between 10 and 10,000.

- A comparison of time for insert functions for random data across the five implementations. The x axis should be the number of insertions performed and the y-axis should be the system time it took the implementations to perform the insertions.
- A comparison of the find function across the five implementations. To test the runtime of find. First, fill the dictionary with some number of elements, this will be your x-axis. Then, perform some constant number of find operations (I suggest at least 100, or machine errors might accumulate). The y-axis will be the time it took to complete the constant number of finds. *Do not vary the number of finds.*

- A comparison of the delete function across the five implementations. Here the x-axis will be the number of elements deleted and the y-axis will be the time that those deletes took. To maintain consistency, the deletes should be all of the elements from the dictionary. For example, for an x-axis point of 100, you should do 100 random inserts and then measure the time it takes to remove all 100 elements.

Use these graphs to answer the following questions.

1. Which of the implementations seems to be the best overall? Explain why?
2. Are there any surprising findings? Do all O(n) functions perform at the same speed? Why or why not?
3. Create inputs for the BSTDict to exhibit best case and worst case scenarios. Show the timing difference for the find function for large values of n.

Deliverables

For this assignment, there will be two submissions on Canvas.

- Part 1 consists of a zip of your five dictionary implementations. Only submit the 5 java files.
- Part 2 is the pdf of the writeup. Include all of the graphs mentioned in the writeup and answer all of the quesstions