CSCI 2114: Tashfeen's Data Structures

Homework 4

You may download all the code for this homework at: https://tashfeen.org/s/ds/hw4code.zip.

Question 1. Recall from class the runtimes given in table 1.

Data Structure\Operation	Random Insert	Random Access
Array List	$\mathcal{O}(n)$	$\mathcal{O}(1)$
Linked List	$\mathcal{O}(1)$	$\mathcal{O}(n)$

TABLE 1. Asymptotic runtime complexities for array lists and linked lists

In this question for each of the array list and linked list, we'll record times for Java's, List.add(E e) and List.get(int index). Note that both of the Java's `LinkedList` and `ArrayList` implement the above `List` interface.

Finish implementing the Java class `ListExperiment`. Once fully implemented, this class should give you two comma separated files (CSV) with rows corresponding with nanoseconds taken to append the i^{th} element or perform a binary search in a list of size 2^i. Plot each row in these CSV files against their index using your favourite plotting utility. You may use this `plot_list.py` (make sure you set the appropriate variables) if you want. Give the two plots.

Do the plots confirm or overrule what we know about the runtimes linked and array lists? Discuss.

Question 2. Extend `Structure.java` to implement `Stack.java` and `Queue.java`, i. e.,

```
public class Queue<T extends Comparable<T>> extends Structure<T> {...}
```

and

```
public class Stack<T extends Comparable<T>> extends Structure<T> {...}
```

1) Use your stack implementation to figure out that of 1,000,000 randomly generated six character strings consisting of only '(' and ')', e. g., "())())" how many are balanced. I. e., "())())" is not balanced while "()(())" is balanced. State your answer as a ratio in the decimal point notation. Save the class in `Balance.java`.
2) Use your queue implementation to draw the Sierpinksi's triangle.

```
1  public class Sierpinski {
2    public static void main(String[] args) {
3      new Canvas(new Sierpinski());
4    }
5    public void points(int x, int y, int w, int h, Queue<Coordinates> q, int r) {
6      // implement me.
7    }
8  }
```

Question 3. Finish implementing the `BinaryHeap.java`. You need to implement the methods `heapifyUp(Node<T> v)` and `heapifyDown(Node<T> v)`. Run the file and verify your implementation keeps the heap property.

SUBMISSION INSTRUCTIONS

1) Turn in a PDF containing any plots, figures and/or answers from the homework.
2) Turn in your, `ListExperiment.java`, `Stack.java`, `Queue.java`, `Balance.java`, `Sierpinski.java` and `BinaryHeap.java`.