Programming Assignment 1

Amortized Dictionary

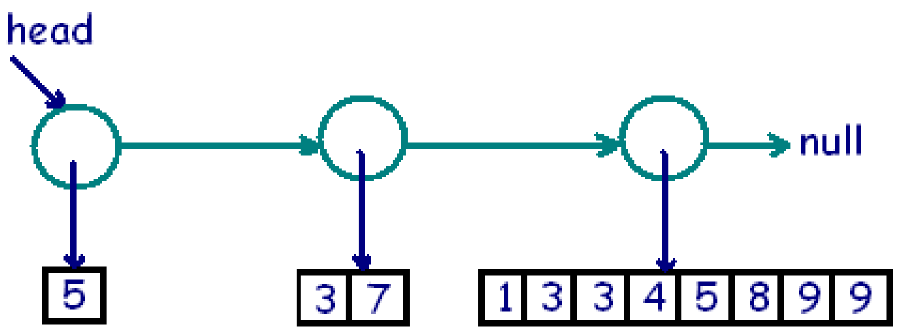
Due date: Feb. 01 by 11:00pm

Overview

One of the most important structures in computer science are the *dictionary* data structures that support fast insert and lookup operations. Dozens of different data structures have been proposed for implementing dictionaries including hash tables, skip lists, search trees and others.

In this lab assignment you will implement a dictionary based on linked lists and sorted arrays. This structure combines a fast lookup on sorted arrays with ease of linked list-insertion. Note that a sorted array is good for lookups (think of a binary search) but bad for inserts. A linked list is good for inserts but bad for lookups (they can take linear time).

The idea of this data structure is as follows. We will have a linked list of arrays, where array k has size 2^k, and each array is in sorted order. However, there will be no relationship between the items in different arrays. The issue of which arrays to be used is based on the binary representation of the number of items we are storing. For example, if we have 11 items to store, then the arrays of sizes 1, 2, and 8 will be used, since 11 = 1 + 2 + 8, and our dictionary might look like this:



This data structure has interesting amortized and average-case performance, hence the name "Amortized Array-Based Dictionary" (AAD), which we will be using thought the writeup.

Objectives

- Refresh your mind about programming in Java

- Analyze a complex data structure
- Gain an understanding of the complexity of algorithms

Starter Code

Download the starter code for assignment (zip file). Once you finish implementation, submit java files to Autolab.

What to Submit

You submit the following java file:

- Dictionary.java

Do not submit anything else like zip files, folders, desktops, class files. Make sure you do not use packages in your java files as well. Failure to do so will result in a penalty.

Instructions

You are going to implement the array-based dictionary data structure

```
public class Dictionary<E extends Comparable<E>> implements DictionaryInterface<E>
{
   private int size;
   private Node head;
   ...
}
```

using a linked list of sorted arrays. Specifically, you'll maintain a linked list of the following `Nodes`:

```
private static class Node
{
   private Comparable[] array;
   private Node next;
   ...
}
```

These nodes are organized in the ascending order of array sizes - there are no two nodes containing arrays of the same size.

You have been given some starter code to help you out. Your main goal is to implement the Dictionary interface:

add(AnyType e):

You create an array of size one, containing a specified element e, and insert the array into the linked list. Next, you must traverse the linked list and merge sorted arrays of the same size until at most one array remains of each size.

contains(AnyType e):

Returns true if the dictionary contains a specified element e, otherwise - false. Since each array in the list is in sorted order, you run Java's binary search on each of them. .

If you get stuck, or don't understand something, re-read the notes. Ask for help if you need it!

Coding Style:

Your first programming assignment won't be graded for the coding style. But it will be graded on correctness as well as its ability to deal with so-called edge cases, like empty lists, lists with only one element, and so on.

Testing

Testing should be straightforward. Given a specific sequence of operations on an AAD, the resulting AAD is unique, so you can test by simply seeing if it is correct (the already-implemented toString() method should be really helpful for doing this).

That being said, here is a TestingDriver.java a class to help test your implementation. It will run some simple tests on your code and provide feedback.

Victor S. Adamchik, CMU, 2016