Introduction to Algorithms: 6.006
Massachusetts Institute of Technology — Fall 2008
Professors Ronald L. Rivest and Sivan Toledo — Handout 7

Quiz 1 Review

Contents

1 Python

Though we are not aiming to test your python knowledge, you should now be familiar with the running times of operations on lists, sets, and dictionaries. Also, Understand how methods such as `__contains__` are used in python.

1. Lists
   How long does it take to ... (you may assume lists are of length n)
   (a) ... access an element:
   (b) ... concatenate (L1 + L2):
   (c) ... append (L1.append(elt)):
   (d) ... slice a list:
   (e) ... len:
2. Sets
   How long does it take to ... (you may assume sets have n elements)
   (a) ... intersect two sets:
3. Dictionaries
   How long does it take to ...
   (a) ... access an element:
   (b) ... contains (x in d):

2 Analysis Methods

2.1 Recurrences

A good site for reviewing recurrences: `http://www.cs.duke.edu/ ola/ap/recurrence.html` Especially the “Recurrence Relations to Remember” section.

How do we obtain recurrence equations from code?

Describe the following recurrence solution methods

- substitution (guess and check)
- iteration (plug and chug)
- recursion tree
- master method

2.2 Asymptotic Growth

Define

1. $f(n) = O(g(n))$
2. $f(x) = \Omega(g(x))$
3. $f(x) = \Theta(g(x))$

2.3 Runtime Analysis

Be able to analyze code / an algorithm and determine its runtime.

2.4 Amortized Analysis

What is the idea behind amortized analysis?
Know how to create, bound and solve cost equations. (See the dynamically resizing hash tables or the textbook for a examples of amortized analysis problems)

3 Data Structures

What is an Abstract Data Type?

3.1 Arrays

(aka Python lists) What is the runtime of insertion, deletion, and search in (i) a sorted array (ii) an unsorted array.

3.2 Trees

3.2.1 Binary Search Trees

1. Describe at a high level the data structure

2. Describe how `find` works

3. Describe how `insert` works

4. Describe how `delete` works

5. Describe how `find-min` works

6. Describe how `next-larger` works

7. Describe best and worst case performance of the above BST operations

3.2.2 AVL Trees

1. What BST problem are AVLs trying to solve?

2. Describe the augmentation and augmentation invariant.

3. Describe a left/right rotation

4. Explain how/when an AVL uses rotations

5. Describe best and worst case performance of the above BST operations on an AVL tree.

3.3 Heaps

1. Describe at a high level the data structure

2. Describe the heap invariant

3. Describe / write psuedocode for the following operations. And give their runtimes.

   (a) `extract-max`

   (b) `heapify`

   (c) `build`

   (d) `increase-key`

   (e) `insert`

3.4 Hash

1. Describe what a hash function is.

2. How do hash tables work? What assumptions do we make about the hash functions used?

3. What are collisions, and how do we handle them?

4. Describe how a probing hash table with linear probing works.

5. Describe how a probing hash table with double hashing works.

6. What is a rolling hash?

3.5 Augmented Data Structures

4 Sorting

4.1 definitions

What does it mean for a sort to be ...

1. ... in place:

2. ... stable:

3. ... a selection sort:

4.2 Insertion Sort

1. Describe at a high level the sorting algorithm

2. Write out the algorithm in psuedo-code

3. Runtime (best case? worst case?)

4. What benefits (pitfalls) does this algorithm have?

5. Is this sort stable? in place?

4.3 Merge Sort

1. Describe at a high level the sorting algorithm

2. Write out the algorithm in psuedo-code

3. Runtime (best case? worst case?)

4. What benefits (pitfalls) does this algorithm have?

5. Is this sort stable? in place?

4.4 Heap Sort

1. Describe at a high level the sorting algorithm

2. Write out the algorithm in psuedo-code

3. Runtime (best case? worst case?)

4. What benefits (pitfalls) does this algorithm have?

5. Is this sort stable? in place?

4.5 Counting Sort

1. Describe at a high level the sorting algorithm

2. Write out the algorithm in psuedo-code

3. Runtime (best case? worst case?)

4. What benefits (pitfalls) does this algorithm have?

5. Is this sort stable? in place?

5 Other

5.1 Merging lists

1. Describe how to merge two sorted lists.

2. What is the runtime of the merge?

3. What is the space requirement of the merge?

5.2 Rabin Karp String Matching

1. How does Rabin-Karp string matching work?

2. What is the runtime of the algorithm?

3. What is the space requirement of algorithm?

5.3 Binary Search

Know the difference between a Binary Search, and a Binary Tree. Do you need a binary tree in order to perform a binary search?

1. How does a Binary Search work?

2. What is the runtime of the algorithm?

5.4 Dynamic resizing

1. What is the high level idea of dynamically resizing a data structure?

2. What sort of analysis would you use to determine the cost of a dynamically resizing data structure?

3. Describe how to resize hash tables and analyze the new runtimes of hash table operations.

5.5 Divide and Conquer

What are the stages of Divide and Conquer?

1.

2.

3.

Priority Queues

1. What is an abstract data type?

2. What are the operations of a Priority Queue?

3. What is a good implementation of a Priority Queue?