MATH 8650 Advanced Data Structures Fall 2018
Midterm Exam

October 19–21, 2018

- This take-home exam is due by 8:00am, October 21, 2018.Sign, scan, and return this cover sheet with your exam. Scan handwritten responses or type responses in Word or LaTeX, and turn in a PDF file with the answers neatly written and in order. For the programming section, turn in a .py file.
- There are a total of 80 points, 10 for each conceptual problem and 20 for the programming part.
- You may consult the published course notes, your own class notes, or the instructor, but no other sources.
- Mark your answers clearly. *Show your work.* Unsupported correct answers receive partial credit.
- Start each numbered question on a new sheet. Be sure to include your name on *each* page.
- Good luck!

Name: VIVEK KOODLI UDUPA

I certify that I have not received any unauthorized assistance in completing this examination.

Signature: [illegible]

Date: Oct 21, 2018

Programming

Implement a Python class `HashTable` with the following specifications:

- For a `HashTable` object `H`,
  - `H = HashTable(n)` creates a hash table with `n` buckets as a Python list with `n` empty entries.
  - `H.insert(k,v)` inserts an object with key `k` and value `v`.
  - `H.delete(k)` removes the object with key `k` if it exists or throws a `KeyError` if not.
  - `H.find(k)` returns the value of the object with key `k` if it exists or throws a `KeyError` if not.
  - `H.list()` lists the key-value pairs in arbitrary order.
- Keys are 10-digit phone numbers and the hash function simply treats the number as an integer and computes its remainder mod `n`.
- There is no collision resolution. If `insert()` produces a hash index that is already occupied, replace the key and value of the existing entry with the new key and value.
- The `list()` method should run in time $O(p)$, where p is the number of live entries in your table. So you need an auxiliary data structure to keep track of which locations contain live entries. Insertions and deletions should take constant time, so your auxiliary data structure should be updatable in constant time. You can use an array to hold copies of keys and indexes into the hash table, and store the index into that array together with the hash-table entry. Append new entries to the end of your array. Delete entries by copying the last entry over the deleted one and popping the last entry.

Conceptual Problems

1. Consider a singly linked list of n nodes, numbered 0, 1, 2, $\ldots$, $n-1$ starting at the head. Write a recursive function to delete all the even-numbered nodes.
2. Let $f(n) = n^{3/2} \log n$ and $g(n) = 3n^2$. Prove rigorously using the definition that $f(n)$ is $O(g(n))$ but $f(n)$ is not $\Theta(g(n))$
3. Consider a binary heap with 18 elements stored in an array. Suppose all elements have distinct priorities. We know the element with the smallest key appears in position 0 of the array.
   (a) In what locations could the element with the second smallest key appear?
   (b) In what locations could the element with the largest key appear?

4. Construct an algorithm to reverse the contents of a stack S of n integers stored in an array, so that the top element in the modified stack is the bottom element of the original stack. You may use an auxiliary stack T of the same size and one auxiliary integer variable v and you must return the original stack S with its elements reversed, i.e., you may not return T instead. You may write pseudocode or python. What is the complexity of your algorithm?

5. You are given an array A of nonnegative integers and you need to determine whether there are two indices i and j such that $A[i] = 3 \times A[j]$. For example, if $A = (3, 24, 10, 19, 72)$ the your algorithm should print *yes* because $72 = 3 \times 24$, but if $A = 6, 25, 72, 19, 20)$, your algorithm should print *no*.

   (a) Describe an algorithm (in pseudocode or 2–3 English sentences) for this problem with worst-case running time $O(n^2)$.

   (b) Describe another algorithm with worst-case running time better than $O(n^2)$. Analyze the complexity of your algorithm.

6. The following is the preorder traversal of a binary expression tree.

$$+ \ \times \ 16 \ - \ 2\ 3\ 9$$

   Draw the expression tree and compute the value of the expression.