Section 05: Midterm Review

Section Problems

1. Selecting ADTs and data structures

For each of the following scenarios, choose:

(a) An ADT: `Stack` or `Queue`

(b) A data structure: `array list` or `linked list with front` or `linked list with front and back`

Justify your choice.

(a) You're designing a tool that checks code to verify all opening brackets, braces, parenthesis, etc... have closing counterparts.

(b) Disneyland has hired you to find a way to improve the processing efficiency of their long lines at attractions. There is no way to forecast how long the lines will be.

(c) A sandwich shop wants to serve customers in the order that they arrived, but also wants to look ahead to know what people have ordered and how many times to maximize efficiency in the kitchen.

2. Eyeballing Big-Θ bounds

For each of the following code blocks, what is the worst-case runtime? Give a big-Θ bound. You do not need to justify your answer.

(a)
```
void f1(int n) {
    int i = 1;
    int j;
    while(i < n*n*n*n) {
        j = n;
        while (j > 1) {
            j -= 1;
        }
        i += n;
    }
}
```

(b)
```
int f2(int n) {
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            System.out.println("j = " + j);
        }
        for (int k = 0; k < i; k++) {
            System.out.println("k = " + k);
            for (int m = 0; m < 100000; m++) {
                System.out.println("m = " + m);
            }
        }
    }
}
```

(c)
```
int f3(n) {
    count = 0;
    if (n < 1000) {
        for (int i = 0; i < n; i++) {
            for (int j = 0; j < n; j++) {
                for (int k = 0; k < i; k++) {
                    count++;
                }
            }
        }
    } else {
        for (int i = 0; i < n; i++) {
            count++;
        }
    }
    return count;
}
```

(d)
```java
void f4(int n) {
    // NOTE: This is your data structure from the first project.
    LinkedDeque<Integer> deque = new LinkedDeque<>();
    for (int i = 0; i < n; i++) {
        if (deque.size() > 20) {
            deque.removeFirst();
        }
        deque.addLast(i);
    }
    for (int i = 0; i < deque.size(); i++) {
        System.out.println(deque.get(i));
    }
}
```

3. Best case and worst case runtimes

For the following code snippet give the big-Θ bound on the worst case runtime as well the big-Θ bound on the best case runtime, in terms of n the size of the input array.

```java
void print(int[] input) {
    int i = 0;
    while (i < input.length - 1) {
        if (input[i] > input[i + 1]) {
            for (int j = 0; j < input.length; j++) {
                System.out.println("uh I don't think this is sorted plz help");
            }
        } else {
            System.out.println("input[i] <= input[i + 1] is true");
        }
        i++;
    }
}
```

4. Big-O, Big-Omega True/False Statements

For each of the statements determine if the statement is true or false. You do not need to justify your answer.

(a) $n^3 + 30n^2 + 300n$ is $\mathcal{O}(n^3)$

(b) $nlog(n)$ is $\mathcal{O}(log(n))$

(c) $n^3 - 3n + 3n^2$ is $\mathcal{O}(n^2)$

(d) 1 is $\Omega(n)$

(e) $.5n^3$ is $\Omega(n^3)$

5. Tree Method

Find a summation for the total work of the following expressions using the Tree Method.

Hint: Just as a reminder, here are the steps you should go through for **any** Tree Method Problem:

i. Draw the recurrence tree.

ii. What is the size of the **input** to each node at level i? As in class, we call the root level $i = 0$. This means that at $i = 0$, your expression for the input should equal n.

iii. What is the amount of **work** done by each node at the i-th *recursive* level?

iv. What is the total number of nodes at level i? As in class, we call the root level $i = 0$. This means that at $i = 0$, your expression for the total number of nodes should equal 1.

v. What is the total work done across the i-th *recursive* level?

vi. What value of i does the last level of the tree occur at?

vii. What is the total work done across the base case level of the tree (i.e. the last level)?

viii. Combine your answers from previous parts to get an expression for the total work.

(a) $T(n) = \begin{cases} T(n-1) + n^2 & \text{if } n > 19 \\ 57 & \text{otherwise} \end{cases}$

(b) $T(n) = \begin{cases} T(n/2) + n^2 & \text{if } n \geq 4 \\ 5 & \text{otherwise} \end{cases}$

(c) $T(n) = \begin{cases} 2T(n/3) + 5n & \text{if } n > 1 \\ 9 & \text{otherwise} \end{cases}$

6. Modeling

Consider the following method. Let n be the integer value of the `n` parameter, and let m be the size of the `LinkedDeque`.

```java
public int mystery(int n, LinkedDeque<Integer> deque) {
    if (n < 7) {
        System.out.println("???");
        int out = 0;
        for (int i = 0; i < n; i++) {
            out += i;
        }
        return out;
    } else {
        System.out.println("???");
        System.out.println("???");
        out = 0;
        // NOTE: Assume LinkedDeque has working, efficient iterator.
        for (int i : deque) {
            out += 1;
            for (int j = 0; j < deque.size(); j++) {
                System.out.println(deque.get(j));
            }
        }
        return out + 2 * mystery(n - 4, deque) + 3 * mystery(n / 2, deque);
    }
}
```

Give a recurrence formula for the **worst-case** running time of this code. It's OK to provide a $\mathcal{O}$ for non-recursive terms (for example if the running time is $A(n) = 4A(n/3) + 25n$, you need to get the 4 and the 3 right but you don't have to worry about getting the 25 right). Just show us how you got there.

7. Challenge: Design

Imagine a database containing information about all trains leaving the Washington Union station on Monday. Each train is assigned a departure time, a destination, and a unique 8-digit train ID number.

What data structures you would use to solve each of the following scenarios. Depending on scenario, you may need to either (a) use multiple data structures or (b) modify the implementation of some data structure.

Justify your choice.

(a) Suppose the schedule contains 200 trains with 52 destinations. You want to easily list out the trains by destination.

(b) In the question above, trains were listed by destination. Now, trains with the same destination should further be sorted by departure time.

(c) A train station wants to create a digital kiosk. The kiosk should be able to efficiently and frequently complete look-ups by train ID number so visitors can purchase tickets or track the location of a train. The kiosk should also be able to list out all the train IDs in ascending order, for visitors who do not know their train ID.

Note that the database of trains is not updated often, so the removal and additions of new trains happen infrequently (aside from when first populating your chosen DS with trains).

8. Hash tables

(a) Consider the following key-value pairs.

(6, a), (29, b), (41, d). (34, e), (10, f), (64, g), (50, h)

Suppose each key has a hash function $h(k) = 2k$. So, the key 6 would have a hash code of 12. Insert each key-value pair into the following hash tables and draw what their internal state looks like:

(i) A hash table that uses separate chaining. The table has an internal capacity of 10. Assume each bucket is a linked list, where new pairs are appended to the end. Do not worry about resizing.

(ii) A hash table that uses linear probing, with internal capacity 10. Do not worry about resizing.

(iii) A hash table that uses quadratic probing, with internal capacity 10. Do not worry about resizing.

9. Analyzing dictionaries

(a) What are the constraints on the data types you can store in an AVL tree?

(b) When is using an AVL tree preferred over a hash table?

(c) When is using a BST preferred over an AVL tree?

(d) Consider an AVL tree with n nodes and a height of h. Now, consider a single call to `get(...)`. What's the maximum possible number of nodes `get(...)` ends up visiting? The minimum possible?

(e) **Challenge Problem:** Consider an AVL tree with n nodes and a height of h. Now, consider a single call to `insert(...)`. What's the maximum possible of nodes `insert(...)` ends up visiting? The minimum possible? Don't count the new node you create or the nodes visited during rotation(s).

10. Big-$\mathcal{O}$

Write down a tight big-$\mathcal{O}$ for each of the following. Unless otherwise noted, give a bound in the worst case.

(a) Insert and find in a BST.

(b) Insert and find in an AVL tree.

(c) Finding the minimum value in an AVL tree containing n elements.

(d) Finding the k-th largest item in an AVL tree containing n elements.

(e) Listing elements of an AVL tree in sorted order

11. Memory, Caches, and B-Trees

(a) What is a cache and what is it used for?

(b) Define the two types of memory locality and give an example of when we might see each type of locality in code.

(c) Does spatial locality benefit arrays or LinkedLists more when we are iterating through each data structure? Why?

(d) Give an example of a situation that would be most appropriate for the use of a B-Tree as a data structure. Are there any constraints on the pieces of data that a B-Tree can store?