Exam I Practice Problem Set

CSE 373 Summer 2020

The purpose of this problem set is to help you prepare for Exam I by giving you examples of the types of skills you may be asked to use. Please note that it is not intended to be representative of the length of the exam nor the exact types of questions you will be asked. For a more thorough list of the topics that may be included, refer to the topics list and learning objectives list published on the exams page on the course website.

1. Design Decisions

For each of the scenarios described below, select one of the following ADTs that would best suit the situation. Then, justify your answer by briefly describing how you would use the ADT. Your justification should mention a specific operation on the ADT, and *why* that operation would be important in this particular situation.

You can choose from the following ADTs: **List, Stack, Queue,** and **Map**. Each ADT should be used exactly once.

1.1 NFL Draft

You are writing a portion of a program to manage the NFL draft. In the draft, teams rotate through 7 rounds of picking in which the team with the worst record for the previous year picks first, followed then by the second and so on until the team that won the super bowl picks last. The order of picking is the same in each round. Which ADT would you choose to manage the order of the teams as they rotate through their turn each round?

1.2 Music Festival

You are writing a program to manage the arrangement of songs at a music festival. Which ADT would you choose to store the order of songs as you build out the show? The process of designing the show will be iterative and you will need the ability to move, add and remove songs before you get to your final design.

1.3 Inbox

You are writing a program to manage unread emails in an inbox. When a new mail arrives, it should appear at the very top of the inbox and as the mails are read, they should be removed from the unread view. Which ADT would you choose to manage the ordering of the mails?

1.4 Online Store

You are writing a program to return search results for an online store. When a user types in a query you should return all the products that include the search term in their title. Which ADT would you choose to manage the products and their titles?

2. Case and Asymptotic Analysis

Scenario: Joyce is a very nervous boi™ trying to land her first SWE internship at a company called Schmoogle™. On the HackerRank challenge they sent her, she was faced with the following problem:

Given an array of integers, return **indices** of the two numbers such that they add up to a specific target. You may assume that each input would have ***exactly*** one solution, and you may not use the same element twice.

Example:

```
Given nums = [2, 7, 11, 15], target = 9,

Because nums[0] + nums[1] = 2 + 7 = 9,
return [0, 1].
```

The company states that she should code her solution as if it were to handle large inputs (~1,000,000 entries) on a computer with a huge amount of memory space. She comes up with the following 3 solutions to the problem, but isn't quite sure which one to go with. And so, she turns to you, hoping that you could help her analyze her code (wow she's cheating on a coding challenge??? Already fired lol).

NOTE: assume that Joyce is utilizing Java's implementation of HashMap within these solutions. You may assume that both map.get() and map.containsKey() consistently run in O(1) time.

2.1 Programming Solution A

```
public int[] twoSum(int[] nums, int target) {
    for (int i = 0; i < nums.length; i++) {
        for (int j = i + 1; j < nums.length; j++) {
            if (nums[j] == target - nums[i]) {
                return new int[] { i, j };
            }
        }
    }
    throw new IllegalArgumentException("No two sum solution");
}
```

a) What is the Big-Theta runtime for this method in the worst case?
b) What is the Big-Theta runtime for this method in the best case?
c) Describe the best and worst case for this method, using example input arrays/target ints to illustrate your response (if applicable). If there is no difference between the two cases, note that instead. (~1-2 sentences)

2.2 Programming Solution B

```
public int[] twoSum(int[] nums, int target) {
    Map<Integer, Integer> map = new HashMap<>();
    for (int i = 0; i < nums.length; i++) {
        int complement = target - nums[i];
        if (map.containsKey(complement)) {
            return new int[] { map.get(complement), i };
        }
        map.put(nums[i], i);
    }
    throw new IllegalArgumentException("No two sum solution");
}
```

a) What is the Big-Theta runtime for this method in the worst case?
b) What is the Big-Theta runtime for this method in the best case?
c) Describe the best and worst case for this method, using example input arrays/target ints to illustrate your response (if applicable). If there is no difference between the two cases, note that instead. (~1-2 sentences)

2.3 Programming Solution C

```
public int[] twoSum(int[] nums, int target) {
    Map<Integer, Integer> map = new HashMap<>();
    for (int i = 0; i < nums.length; i++) {
        map.put(nums[i], i);
    }
    for (int i = 0; i < nums.length; i++) {
        int complement = target - nums[i];
        if (map.containsKey(complement) && map.get(complement) != i) {
            return new int[] { i, map.get(complement) };
        }
    }
    throw new IllegalArgumentException("No two sum solution");
}
```

a) What is the Big-Theta runtime for this method in the worst case?
b) What is the Big-Theta runtime for this method in the best case?
c) Describe the best and worst case for this method, using example input arrays/target ints to illustrate your response (if applicable). If there is no difference between the two cases, note that instead. (~1-2 sentences)

2.4 Comparing Solutions

Out of solutions A, B and C, which solution seems to be the best or “most optimal”? Defend your selection in 2-3 sentences.

3. Hashing

In the following problems, refer to the following `Point` class.

```
public class Point {
      public final int x, y;
      public Point(int x, int y) {
            this.x = x;
            this.y = y;
      }
      public int hashCode() {
            return this.x + this.y;
      }
}
```

3.1 Collisions

a) Which of the following points will collide with `Point(1, 2)` in a hash table with 2 buckets?

```
Point(1, 1)
Point(3, 1)
Point(1, 4)
Point(2, 1)
Point(1, 3)
Point(1, 5)
```

b) Does there exist a point that is guaranteed to collide with Point(1, 2) for any number of buckets?

( ) Yes
( ) No
( ) More Information is Needed

If yes, mark all of the points that are guaranteed to collide.

```
Point(1, 1)
Point(3, 1)
Point(1, 4)
Point(2, 1)
Point(1, 3)
Point(1, 5)
```

3.2 Properties of Hash Maps

One of the important properties for a hash function is for it to be **deterministic**: giving the function the same input should produce the same output.

a) Describe an implementation of the hashCode function for the Point class that is NOT deterministic. You do not need to write code (describe at a high level), and you may assume you have access to any external libraries you want to use.

b) Suppose we have a Hash Map of Point objects with a non-deterministic hashCode method. Give an example series of method calls on that map that could give incorrect behavior. You do not need to justify your answer in this part.

c) Explain *why* a non-deterministic hash function could break the Hash Map. What role does a hash function play in the implementation of a Hash Map, and why would inconsistent results prevent that behavior?

4. Recursive Code Analysis

4.1 Recursive Code Modeling

Consider the following recurrence relation, where c is some arbitrary constant.

$$T(n) = \begin{cases} c & \text{if } n=0 \\ 5T(n/3) + n^2 & \text{otherwise} \end{cases}$$

Complete the following Java code so that the above recurrence models its **runtime** in terms of n.

```
public void fun(int n) {
      if (                        ) {



      } else {









      }
}
```

5. AVL Trees

5.1 AVL Insertions

Give an ordering of the keys: 1, 2, 3, 4, 5, 6, 7 such that inserting those keys into an empty AVL tree will cause no rotations.

5.2 Properties of AVL Trees

For each of the following statements:

- State whether the statement is true or false
- If the statement is false, briefly justify your answer (~1-2 sentences).

a) If you're inserting a **new** key into an AVL tree, the best-case runtime is O(1).

b) The **maximum** number of rotations on a single insert in an AVL tree is Θ(1).

c) It is possible to insert a value into an AVL tree such that no rotations can fix the AVL invariant.