Final Review

STUDENT NAME

Search students by name or email...

Q1 Hashtables
23 Points

Q1.1 Bad Hash Function
4 Points

Provide an example of a bad valid hashing function for Objects.

Enter your answer here

Why is it bad?

Enter your answer here

Save Answer

Q1.2 Lots of Collisions
4 Points

If you were to run into a lot of collisions with linear probing, how would you go about redistributing your data without resizing or modifying your original hash function? Justify your answer.

Enter your answer here

Save Answer

Q1.3 Coding Hunger Games

15 Points

Head Gamemaker Sashu's Coding Hunger Games

Head Gamemaker Sashu is hosting her own Hunger Games for 332. She is offering extra office hours to anyone who will sign up for it, but there is a catch. For each requested office hours (with a **maximum of 10 requested office hours per person**), the student's name will be placed in a digital bowl and at the end of the quarter, 12 names will be drafted from that digital bowl to compete in Sashu's Coding Hunger Games. The first person to finish the coding challenge will get a 4.0 for the quarter while the rest of the competitors will lose all of the points to one of the projects.

You are tasked with designing a system in which Sashu can draw the names, specifically implementing the following methods:

- `addSlip(name)`: insert a slip with the given name `name` into the digital bowl (because people can sign up for multiple office hours, duplicates will be allowed)
- `withdrawSlips(name)`: withdraw every slip with the given name from the bowl (e.g. if they are unable to compete, if they have already been selected)
- `pickTribute()`: randomly select a name `name` from the bowl for the Coding Hunger Games and withdraw all slips with name `name` from the bowl. The probability of a specific name being selected is proportional to the number of slips with `name` in the bowl

The average runtime of all of the operations should be similar to that of a hashtable.

Describe the data structure(s) used in your implementation in 1-2 sentences:

Enter your answer here

Describe your implementation for `addSlip(name)` and justify its runtime in 2-3 sentences:

Enter your answer here

Describe your implementation for `withdrawSlips(name)` and justify its runtime in 2-3 sentences:

Enter your answer here

Describe your implementation for `pickTribute()` and justify its runtime in 2-3 sentences:

Enter your answer here

Save Answer

Q2 Sorting
28 Points

Q2.1 List of Lists
6 Points

Suppose we have an unsorted ArrayList of unsorted LinkedLists of integers in Java. We want to sort the LinkedLists in ascending order based on the largest value contained in that LinkedList. If there is a tie, break ties by sorting in ascending order by the smallest value contained in that LinkedList. If there is a tie in terms of both the largest and smallest values in a LinkedList, then break that tie by sorting in ascending order by the size of the LinkedList. After that ties are broken arbitrarily. You may assume that the size of each LinkedList is stored but that the min and max value in each list may NOT be stored.

The algorithm we will use is Quicksort using median-of-three pivot selection and the partitioning algorithm shown in lecture (Hoare partitioning).

Respond to the questions below with n as the size of the ArrayList and with a as the average size of the LinkedLists. **Keep terms involving n and a in your final answer.** Do not use any other variables in your answer.

What is the worst-case runtime of the **pivot selection** part of the algorithm? Give a tight big-O bound and EXPLAIN your answer in a couple of sentences.

Enter your answer here

What is the worst-case runtime of the **partition** part of the algorithm (not including the pivot selection)? Give a tight big-O bound and EXPLAIN your answer in a couple of sentences.

Enter your answer here

What is the **overall worst-case runtime** of the algorithm? Give a tight big-O bound and EXPLAIN your answer in a couple of sentences.

Enter your answer here

Q2.2 Sort 1

6 Points

Here are the first few steps in a sorting algorithm. Choose the most appropriate sorting algorithm and briefly justify your answer.

```
1. [8, 6, 3, 4, 1, 0, 2, 7, 9, 5]
2. [8, 6, 3, 4, 5, 0, 2, 7, 9, 1]
3. [8, 6, 3, 9, 5, 0, 2, 7, 4, 1]
4. [8, 9, 3, 7, 5, 0, 2, 6, 4, 1]
5. [9, 8, 3, 7, 5, 0, 2, 6, 4, 1]
6. [1, 8, 3, 7, 5, 0, 2, 6, 4, 9]
7. [8, 7, 3, 6, 5, 0, 2, 1, 4, 9]
8. [4, 7, 3, 6, 5, 0, 2, 1, 8, 9]
9. [7, 6, 3, 4, 5, 0, 2, 1, 8, 9]
10.[1, 6, 3, 4, 5, 0, 2, 7, 8, 9]
11.[6, 5, 3, 4, 1, 0, 2, 7, 8, 9]
```

- ○ Insertion Sort
- ○ Selection Sort
- ○ Heap Sort
- ○ Merge Sort
- ○ Quick Sort
- ○ Bucket Sort
- ○ Radix Sort

Briefly explain how you determined which algorithm was running. What are the defining features of that sort?

Enter your answer here

Save Answer

Q2.3 Sort 2

6 Points

Here are the first few steps in a sorting algorithm. Choose the most appropriate sorting algorithm and briefly justify your answer.

```
1. [8, 6, 3, 4, 1, 0, 2, 7, 9, 5]
2. [6, 8, 3, 4, 1, 0, 2, 7, 9, 5]
3. [3, 6, 8, 4, 1, 0, 2, 7, 9, 5]
4. [3, 4, 6, 8, 1, 0, 2, 7, 9, 5]
5. [1, 3, 4, 6, 8, 0, 2, 7, 9, 5]
6. [0, 1, 3, 4, 6, 8, 2, 7, 9, 5]
7. [0, 1, 2, 3, 4, 6, 8, 7, 9, 5]
8. [0, 1, 2, 3, 4, 6, 7, 8, 9, 5]
9. [0, 1, 2, 3, 4, 6, 7, 8, 9, 5]
10. [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

- ○ Insertion Sort
- ○ Selection Sort
- ○ Heap Sort
- ○ Merge Sort
- ○ Quick Sort
- ○ Bucket Sort
- ○ Radix Sort

Briefly explain how you determined which algorithm was running. What are the defining features of that sort?

Enter your answer here

Save Answer

Q2.4 Sort 3

6 Points

Here are the first few steps in a sorting algorithm. Choose the most appropriate sorting algorithm and briefly justify your answer.

```
1. [8, 6, 3, 4, 1, 0, 2, 7, 9, 5]
2. [8, 6, 4, 0, 2, 3, 1, 7, 9, 5]
3. [8, 4, 0, 1, 9, 5, 6, 2, 3, 7]
4. [8, 0, 1, 9, 2, 3, 4, 5, 6, 7]
5. [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

- ○ Insertion Sort
- ○ Selection Sort
- ○ Heap Sort
- ○ Merge Sort
- ○ Quick Sort
- ○ Bucket Sort
- ○ Radix Sort

Briefly explain how you determined which algorithm was running. What are the defining features of that sort?

Enter your answer here

Save Answer

Q2.5 A Place for Horses

4 Points

In your own words define the terms **in-place** and **stable** in terms of **sorting**. For each, describe a real-life sorting application where that property would be useful.

For example, a sorting application for bucket sort would be a job recruiter sorting applicants by GPA. Bucket sort is a good application here because the values are within a limited range and it's fast.

In-place Definition

Enter your answer here

Real-life In-place Application

Enter your answer here

Stable Definition

Enter your answer here

Real-life Stable Application

Enter your answer here

Save Answer

Q3 Parallelism Patterns

15 Points

For problems 3.1-3.5, select the parallelism pattern that would be best suited to tackling each problem.

- reduce
- map operation
- prefix (aka "scan")
- pack (aka "filter")

Select the parallelism pattern ***most appropriate*** for solving the problem and *briefly* **explain why** (a short phrase).

Q3.1 Categorize

3 Points

You are browsing Yelp, looking for a restaurant to order take out from. To make things easier, you would like to categorize each restaurant in a list of restaurants by their cuisine. For simplicity, each restaurant fits into only one cuisine. The output should be a list of pairs, <Restaurant:Cuisine>.

- ○ reduce
- ○ map operation
- ○ prefix (aka "scan")
- ○ pack (aka "filter")

Explanation

Enter your answer here

Save Answer

Q3.2 $$$

3 Points

You've picked a restaurant and now have their menu in front of you. You would like to find the most expensive dish on their menu. If dishes are tied, you can arbitrarily tie break.

- ○ reduce
- ○ map operation
- ○ prefix (aka "scan")
- ○ pack (aka "filter")

Explanation

Enter your answer here

Save Answer

Q3.3 Eat your Veggies!

3 Points

For this meal, you decide to eat vegetarian to get your daily dose of veggies. For convenience, Yelp has flagged all vegetarian dishes with a vegetarian tag. Create a list containing the names of the vegetarian dishes from the menu.

- ○ reduce
- ○ map operation
- ○ prefix (aka "scan")
- ○ pack (aka "filter")

Explanation

Enter your answer here

Save Answer

Q3.4 Just Imagine Waiting...

3 Points

You've ordered your food and start daydreaming about a time when you could go outside and eat at this restaurant. What would that be like? What does a restaurant table feel like anymore? Anyways, you imagine that you are given a list of wait times between each party. How would you find the cumulative wait time for each party?

- ○ reduce
- ○ map operation
- ○ prefix (aka "scan")
- ○ pack (aka "filter")

Explanation

Enter your answer here

Save Answer

Q3.5 Are you gonna eat that dessert?

3 Points

You decide to treat yourself from the dessert menu, but you want to stay under budget. How would you create a list of all the menu items under $10?

- ○ reduce
- ○ map operation
- ○ prefix (aka "scan")
- ○ pack (aka "filter")

Explanation

Enter your answer here

Save Answer

Q4 Parallel Suffix

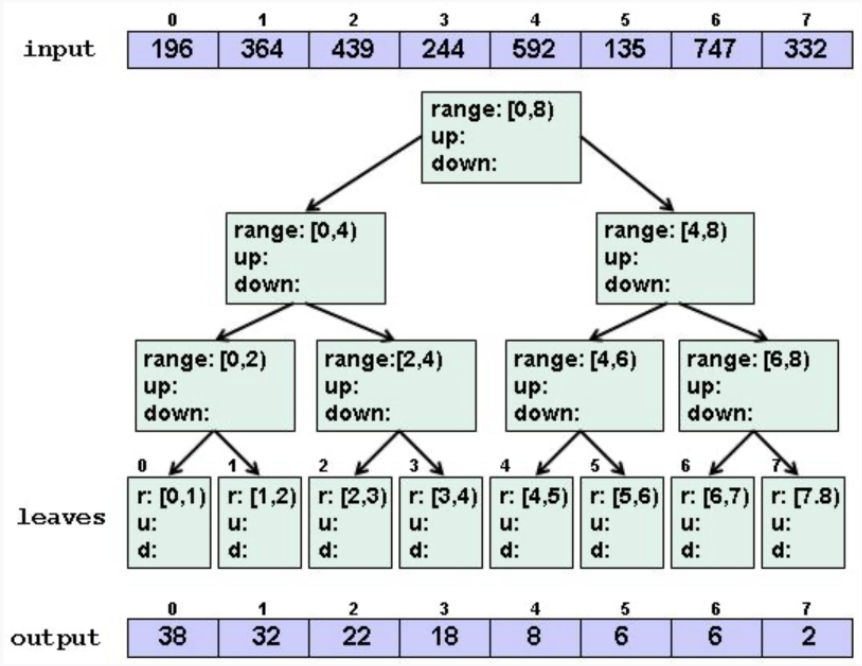
6 Points

For this problem, our `input` is an **array of integers** that are only **3-digit numbers**. We want our `output` array to represent the **sum** of all **even digits** found in the 3-digit numbers from our index to the **right** in the `input` array. In other words, `output[i]` should contain the sum of all even digits in the 3-digit numbers from `input[i]` to `input[input.length - 1]`. Note: that we are **summing** the **digits** that are even, not the values that are even. `input` and `output` are zero-indexed.

This is *similar* to the **parallel prefix** computation we did in class, but from the right instead of from the left. Below is a diagram representing the tree structure. Each node has four fields:

`up` - the field used for the up pass
`down` - the field used for the down pass
`left` - reference to its left child
`right` - reference to its right child
`leaves` - the leaves can be referenced using array indexing, but the leaves are part of the tree, they are not the `output` array.

Fill in each blank with the appropriate calculation. In your calculations, you can refer to things like `input[i]`, `node.down`, `node.left.up`, `leaves[i].down`, etc.



Examples:

```
output[0] = 38
...
output[5] = 4 + 2 + 0 = 6
output[6] = 4 + 2 = 6
output[7] = 2
```

Q4.1 leaves[i].up

1 Point

How would we calculate: (You can describe this algorithm in English if you like)

`leaves[i].up =`

Enter your answer here

Save Answer

Q4.2 node.up

1 Point

How would we calculate: (use pseudocode)

`node.up =`

Enter your answer here

Save Answer

Q4.3 root.down

1 Point

What is the value of: (use pseudocode)

`root.down =`

Enter your answer here

Save Answer

Q4.4 node.left.down

1 Point

How would we calculate: (use pseudocode)

`node.left.down =`

Enter your answer here

Save Answer

Q4.5 node.right.down

1 Point

How would we calculate: (use pseudocode)

`node.right.down =`

Enter your answer here

Save Answer

Q4.6 output[i]

1 Point

How would we calculate: (use pseudocode)

`output[i] =`

Enter your answer here

Save Answer

Q5 Grade Our Assignments Faster

16 Points

The teaching staff is considering a new approach to grading assignments, hoping to leverage concurrency as well as grading assignments as they come in to get grades out as fast as possible (we will not be doing this for this quiz however). For the assignment in question, all 9 TAs will be grading. We have written the following program to help assign student submissions to TAs:

You may want to also look at the explanation for the code, appearing after the code.

```
public class GradingAssigner {
1   private int[] tas = {-1, -1, -1, -1, -1, -1, -1, -1, -1};
2   private int[] submissions = new int[100];
3   private int nextSubmission = 0;
4   private int numUngraded = 0;
5   private ReentrantLock submissionsLock, numUngradedLock, taLock;6
6
7   public void newSubmission(int studentId) {
8      // Add a new submission and ensure each TA has something to grade
9     numUngradedLock.acquire();
10    if (howManyUngraded() > submissions.length) {
11      numUngradedLock.release();
12      throw new UhhSystemIsBrokenException();
13    } else if (howManyUngraded() == submissions.length) {
14      System.out.println("Sorry, please try submitting again later :(");
15    } else {
16      submissionsLock.acquire(); // Add this submission
17      submissions[(nextSubmission + numUngraded) % submissions.length] = studentId;
18      numUngraded = howManyUngraded() + 1;
19      taLock.acquire(); // Make sure each TA has something to grade.
20      for (int i = 0; i < tas.length; i++) {
21        if (tas[i] < 0) {
22          getNextSubmission(taId);
23        }
24      }
25      taLock.release();
26      submissionsLock.release();
27    }
28    numUngradedLock.release();
29  }
30
```

```
31  public void finishGrading(int taId) {
32     // Report that a TA has finished what they were assigned to grade
33     // and assign them something new to grade.
34    taLock.acquire();
35    if (tas[taId] < 0) {
36      System.out.println("This TA is not grading");
37    } else { // Report that TA finished what they were assigned to grade
38      int currStudent = tas[taId];
39      tas[taId] = -1;
40      System.out.println("Submission #" + currStudent + "is ready!");
41      numUngradedLock.acquire()
42      if (howManyUngraded() > 0) { // Assign something new to grade
43        getNextSubmission(taId);
44      }
45      numUngradedLock.release();
46    }
47    taLock.release();
48  }
49
50  public void getNextSubmission(int taId) {
51     // Assign a submission to this TA
52    taLock.acquire();
53    numUngradedLock.acquire();
54    if (tas[taId] >= 0) {
55      System.out.println("TA #" + taId + " is grading submission #" + tas[taId]);
56    } else if (howManyUngraded() <= 0) {
57      System.out.println("All submissions so far have been graded");
58    } else {
59      submissionsLock.acquire();
60      int nextStudent = submissions[nextSubmission];
61      nextSubmission = (nextSubmission + 1) % submissions.length;
62      numUngraded = howManyUngraded() - 1;
63      tas[taId] = nextStudent;
64      submissionsLock.release();
65      System.out.println(taId + " is now grading submission #" + nextStudent);
66    }
67    numUngradedLock.release();
68    taLock.release();
69  }
70
71  public int howManyUngraded() {
72    return numUngraded;
73  }
}
```

Fields

`tas` is an array of size 9, where each TA is assigned an index. The contents at that index will be the student id of the submission the TA should be grading right now or -1 if the TA does not currently have something to grade.

`submissions` holds the list of submissions, where the contents of the array are the student ids corresponding to the submitted assignments.

Methods:

`newSubmission()` will be called whenever a student submits the assignment. This method will add the submission to the end of the queue, and will ensure that each TA is assigned a submission to grade unless there are no submissions left.

`finishGrading()` will be called by the TA after they are done grading their submission. If there are more submissions, the TA will be assigned the next submission in the queue, otherwise, they will be marked as not grading, -1.

`getNextSubmission()` will be called by the TA to get the next submission in the queue, or will let the TA know which submission they are currently assigned to.

`howManyUngraded()` will return the number of ungraded submissions.

For ease of explanation, each method has line numbers.

This series of questions is **not** about debugging the specific methods. It **is** about debugging the concurrency issues. You may assume that each method, if run sequentially, would behave as expected.

Q5.1 Grade Them Quickly

6 Points

Each TA can call `finishGrading()`, `getNextSubmission()`, and `howManyUngraded()` in parallel while the students can call `newSubmission()` in parallel. The teaching staff has started to use locks for our system, as shown above, but we are coming across some concurrency issues. Which of the following concurrency issues are we susceptible to?

- [ ] race condition
- [ ] potential deadlock
- [ ] data race
- [ ] bad interleaving
- [ ] exception thrown
- [ ] none of these

Justify each of your selections; you may need to refer to line numbers in your explanation. For each of the concurrency issues, write at most 3 sentences of justification. **No credit will be given for a selection above if no justification is given.**

Enter your answer here

Save Answer

Q5.2 Grade All of Them

10 Points

Keeping the same locks, restructure where the locks are acquired and released such that the concurrency issues no longer exist as well as making each method as concurrent as possible. You should not modify the methods aside from where the locks are acquired and released.

Feel free to copy and paste the methods from above and change the lines where the locks are acquired and released. (We will ignore line numbers when grading, so if you cut and paste, no need to go back and fix them all.)

Newly modified `newSubmission()`:

Enter your answer here

Newly modified `finishGrading()`:

Enter your answer here

Newly modified `getNextSubmission()`:

Enter your answer here

Newly modified `howManyUngraded()`:

Enter your answer here

Save Answer

Q6 Planet RARSWHJKDH-332

24 Points

332 TAs are secretly astronauts. One day, they land on a new planet, *RARSWHJKDH-332*.

Planet *RARSWHJKDH-332* consists of independent empires. Each empire is made up of multiple cities, connected by bi-directional roads. Inhabitants can get from any city to any other city by way of a route consisting of one or more roads within a empire. However, there is no road connecting two different empires together, since each empire is isolated.

The TAs are given a map of **all** the roads on the planet. The map consists of a list of pairs, in the form: **(cityA, cityB)**. Each pair shows that the city named A is connected to a different city named B by a single road. Each city name is **planetary unique**.

Q6.1 Hello world

6 Points

The TAs are curious to find out how many independent empires there are on the planet. Your task is to design an algorithm to do so, using the planet map given to you.

Describe your algorithm in 3 - 4 sentences.

Enter your answer here

Save Answer

Q6.2 Spaceship landing

6 Points

The TAs choose the empire *Hansa* for further exploration and are given a map containing just the roads for *Hansa*. Interestingly, the roads in *Hansa* do not form a cycle.

The TAs need to find out **which city** in *Hansa* is the best for landing their spaceship.

Upon landing, the TAs will send out robots from the city to explore the rest of the empire. Every single day, a single robot can reach and explore **one** city via one connected road (thus all roads should be considered the same length for this problem). Suppose the TAs have an **infinite** number of robots, the goal is to find the **minimum** number of days for the robots to finish exploring all of the cities in *Hansa*. Design an algorithm to determine which city is the best to land the spaceship.

Describe your algorithm in 3 - 4 sentences.

Enter your answer here

Save Answer

Q6.3 Hit the road

6 Points

The TAs make a tourist stop in the city *Windijef* in the empire *Keshuth*.
Now, they want to start a roadtrip to travel to *Richushi*, a touristy city in *Keshuth*.
Ruchushi is renowned for rich-ness of history and beautiful shi-neries.

For each of the following parts, describe your idea in 1 - 2 sentences.

1. Design an algorithm to find **all possible paths** from *Windijef* to *Richushi*.

Enter your answer here

2. Suppose the TAs can travel to a new city every day, how do you find a path that will take them the least number of days to get to *Richushi* ?

Enter your answer here

3. Now, suppose that the TAs are made aware that the roads are different lengths, and it takes longer time to travel between two cities if the city is further away. How would you change your algorithm accordingly?

Enter your answer here

Save Answer

Q6.4 City exploration

6 Points

The TAs finally arrive at *Richushi*.
They decide to visit some history museums to understand the city's past.
However, they cannot just visit the museums in any order they want, they need to visit them respecting the ordering suggestions of the *Richushi* tourism bureau.

You are given suggestions of the tourism bureau as a list of pairs in the form of **(MuseumA, MuseumB)**.
The pair means that the museum named A needs to be visited before visiting the museum named B.

Here are three pieces of additional information:

1. Each museum name is unique.
2. TAs cannot visit the two museums in the same pair on the same day.
3. Otherwise, they can visit as many museums as they want on the same day.

Your task is to come up with a museum touring plan that minimizes the number of days to visit every single museum in *Richushi*. It is guaranteed that at least one valid schedule exists.

Describe your algorithm in 3 - 4 sentences.

Enter your answer here

Save Answer

Q7 P vs NP

5 Points

For the following problems, select **ALL** the sets they (most likely) belong to.

Q7.1

1 Point

Finding the shortest path from one vertex to another vertex in a weighted directed graph

- [ ] NP
- [ ] P
- [ ] NP-complete
- [ ] None of these

Save Answer

Q7.2

1 Point

Finding a cycle that visits each edge in a graph exactly once

- [ ] NP
- [ ] P
- [ ] NP-complete
- [ ] None of these

Save Answer

Q7.3
1 Point

Determining if a program will run forever

- [ ] NP
- [ ] P
- [ ] NP-complete
- [ ] None of these

Save Answer

Q7.4
1 Point

Finding the prefix sum of an array in parallel using 10 processors

- [ ] NP
- [ ] P
- [ ] NP-complete
- [ ] None of these

Save Answer

Q7.5
1 Point

Finding a path that starts and ends at the same vertex that visits every vertex exactly once

- [ ] NP
- [ ] P
- [ ] NP-complete
- [ ] None of these

Save Answer