SOLUTIONS

Section 10: Final Review

Q: Runtime Analysis

For each of the following code blocks, what is the worst-case runtime? Give a big-Θ bound.

(a)
```
int result = 0;
for (int i = 0; i < n; i++) {
    for (int j = 0; j < i; j++) {
        result++;
    }
}
```

(b)
```
public IList<String> repeat(DoubleLinkedList<String> list, int n) {
    IList<String> result = new DoubleLinkedList<String>();
    for(String str : list) {
        for(int i = 0; i < n; i++) {
            result.add(str);
        }
    }
    return result;
}
```

(c)
```
public void foo(int n) {
    for (int i = 0; i < n; i++) {
        for (int j = 5; j < i; j++) {
            System.out.println("Hello!");
        }

        for (int j = i; j >= 0; j -= 2) {
            System.out.println("Hello!");
        }
    }
}
```

(d)
```
public int num(int n){
        if (n < 10) {
            return n;
        } else if (n < 1000) {
            return num(n - 2);
        } else {
            return num(n / 2);
        }
    }
```

For this last example, also practice writing a recurrence relation that represents the behavior of this code.

(e)
```
public int foo(int n) {
        if (n <= 0) {
            return 3;
        }
        int x = foo(n - 1);
        System.out.println("hello");
        x += foo(n - 1);
        return x;
    }
```

Q: Recursive Patterns and Tree Method

Consider the following recursive methods, analyzing in terms of parameter *n*:

```
public int f1(int n) {
        if (n <= 1) {
            return 1;
        } else {
          return f1(n / 2) + 1;
        }
    }

public int f2(int n) {
        if (n <= 1) {
            return 1;
        } else {
          return f2(n / 2) + f2(n / 2) + 1;
        }
    }
```

Which of the following statements are true about the recursive call trees for these two methods, as would be visualized when applying the tree method? **Select all that apply.**

- f1 and f2 differ in the height of their recursive call trees.
- f1 and f2 differ in the total number of nodes in their recursive call trees.
- f1 and f2 differ in the total amount of work done in their recursive call trees.
- f1 and f2 differ in the total amount of work done in the last level of their recursive call trees.

Q: TST vs. Trie Insertion

Given the phrase **"cats can call cars"...**

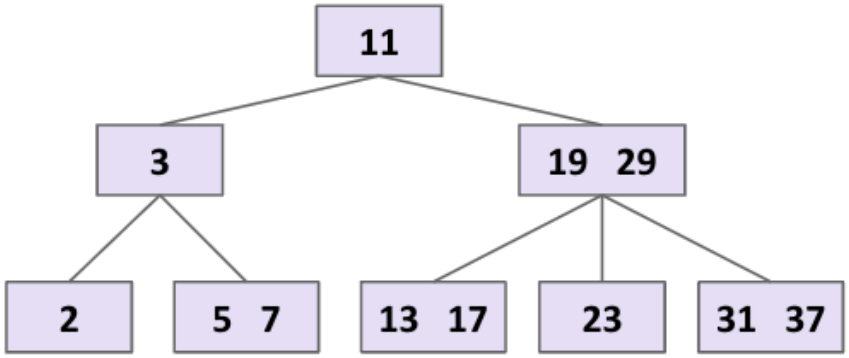
1. Draw the corresponding TST that results from inserting these strings in the above order.
2. Draw the corresponding Trie. *You can omit full arrays and null references.*

Hint: Be sure to adhere to proper alphabetical order!

3. After completing 1) and 2), insert a string of your choice to the Trie and TST that would **increase the height of the TST**, but not the height of the Trie.

Q: Fill Factor

Consider the below tree constructed **by inserting [2, 11, 13, 19, 31, 3, 5, 17, 23, 7, 29, 37]**.



1. Provide **4** numbers that, when inserted sequentially, would **maintain** the current tree height.

2. Provide **4** numbers that, when inserted sequentially, would **increase** the current tree height.

Q: Heaps

1. Suppose you have an array representation of a heap. Must the array be sorted?

2. Suppose you have a sorted array (in increasing order). Must it be the array representation of a valid min heap?

3. You have an array representation of a min-heap. If you reverse the array, does it become an array representation of a max-heap?

Q: Hashing Points

Considering the following code:

```
public class Point {
	public final int x, y;

	public Point(int x, int y) {
		this.x;
		this.y;
	}

	public int hashCode() {
		return this.x + this.y;
	}
}
```

```
public boolean equals(Object o) {
	Point other = (Point) o;
	return this.x == other.x;
}
```

A. Check all the points that will collide with `Point(1, 2)` on a hash table with M = 2 buckets. *Explain why each point does or does not collide by using its hashCode to find the index of its bucket.*

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

- `Point(1, 5)`

B. Assume `Point(1, 2)` is put into an empty hash table with M = 2 buckets using `Point.hashcode`. **If two points are equal if and only if their x-values are equal, what would calling `contains(Point(1, 3))` on the hash table return?**

Explain your answer by explaining how contains uses both the hashCode and equals methods to search for objects.

A. TRUE
B. FALSE
C. NOT ENOUGH INFORMATION

Q: Sorting Cases

When choosing an appropriate algorithm, there are often several trade-offs that we need to consider. Inspect the chart for the following sorting algorithms. Then, for each sort, provide an example of **both** a best-case and worst case input.

	Runtime (best)	Runtime (worst)	Stable? (Y/N)	**Best- and Worst-Case Input**
A: Selection Sort	$\Theta(N^2)$	$\Theta(N^2)$	No	
B: Insertion Sort	$\Theta(N)$	$\Theta(N^2)$	Yes	

C: Merge Sort	$\Theta(N \log N)$	$\Theta(N \log N)$	Yes	
D: Quicksort	$\Theta(N \log N)$	$\Theta(N^2)$	No	
E: Heapsort	$\Theta(N)$	$\Theta(N \log N)$	No	

Q: MSD and LSD Radix Sort

Explain, in plain English, how each of these counting sorts work. Indicate what their worst-case runtimes are, and what each of the variables in your provided runtime refer to.

For each variable, diagram how they individually impact the worst-case runtime.

Q: Sorting Design

For each of the following scenarios, say which sorting algorithm you think you would use and why. There may be more than one right answer.

1. Suppose we have an array where we expect the majority of elements to be sorted "almost in order". What would be a good sorting algorithm to use?

2. You are a triage nurse and you are in charge of the queue that determines who sees the doctor first. You keep a single sorted list in a spreadsheet, and as new patients arrive you insert them into the queue based on the severity of their injury. If two patients have the same severity, the patient who arrived first should get to see the doctor first.

3. You're writing the backend for the website SortMyNumbers.com, which sorts numbers given by users.

4. Your artist friend says for a piece she wants to make a computer sort every possible ordering of the numbers $1, 2, \ldots, 15$. Your friend says something special will happen after the last ordering is sorted, and you'd like to see that ASAP.

5. You are a teacher and you release your students for lunch. The students know they are supposed to let younger students go first, but in their excitement, they forget and line up all out of order. Students don't like letting younger ones cut but are willing to swap line positions with a younger student if you ask them. There is not space in the cafeteria for an extra line, so you have to direct the students within the same line.

Q: Dijkstra's Debugging

Your best friend tried their hand at writing a more complete Dijkstra's pseudocode. However, you noticed something isn't quite right. **Find the bug(s), explain why the behavior isn't correct, and suggest a fix.**

```
dijkstraShortestPath(G graph, V start, V end)
    Set known; Map edgeTo, distTo;
    initialize distTo with all nodes mapped to infinity, except start to 0

    while (there are unknown vertices):
        let u be the closest unknown vertex
        known.add(u)
        for each edge (u,v) to unknown v with weight w:
            if (v == end)
                return path // We found the goal node, we're done!
            oldDist = distTo.get(v)
            newDist = distTo.get(u) + w
            if (newDist < oldDist):
                distTo.put(v, w)
                edgeTo.put(v, u)
```

Q: Graph Design 1

Suppose we are trying to design a maze within a 2d top-down video-game. The world is represented as a grid, where each tile is either an impassable wall, an open space a player can pass through, or a *wormhole*. On each turn, the player may move one space on the grid to any adjacent open tile. If the player is standing on a wormhole, they can instead use their turn to teleport themselves to the other end of the wormhole, which is located somewhere else on the map.

Now, suppose that there are several coins scattered throughout the map. Your goal is to design an algorithm that finds a path between the player and some coin in the fewest number of turns possible.

1. **Describe how you would represent this scenario as a graph** (what are the vertices and edges? Is this a weighted or unweighted graph? Directed or undirected?).

2. Then, **describe how you would implement an algorithm to complete this task.**

Q: Graph Design 2

After 4 snow days last year, UW has decided to improve its snow response plan. Instead of doing "late start" days, they want an "extended passing period" plan. The goal is to clear enough sidewalks that everyone can get from every classroom to every other **eventually** but not necessarily very quickly.

Unfortunately, UW has access to only one snowplow. Your goal is to determine which sidewalks to plow and whether it can be done in time for the first 8:30 AM lecture of the day.

You have a map of campus, with each sidewalk labeled with the time it will take to plow to clear it.

(a) Describe a graph that would help you solve this problem. You will probably want to mention at least what the vertices and edges are, whether the edges are weighted or unweighted, and directed or undirected.

(b) What algorithm would you run on the graph to figure out which sidewalks to plow? Explain why the output of your algorithm will be able to produce a "extended passing period" plowing plan.

(c) How can you tell whether the plow can actually clear all the sidewalks in time?

Q: MSTs Knowledge Check

Answer each of these true/false questions about minimum spanning trees:

1. A MST contains a cycle.

1. If we remove an edge from a MST, the resulting subgraph is still a MST.

2. If we add an edge to a MST, the resulting subgraph is still a MST.

3. `If there are V vertices in a given graph, a MST of that graph contains |V | − 1 edges.`