CS 1332 Studypalooza Worksheet

The final exam is cumulative and covers everything taught this semester! This document has many, many questions but does not touch on every topic, so we would recommend reviewing any missing topics.

Document outline:

1. Big-O Review Tables
   a. Data Structures Table
   b. Sorting Algorithms Table
   c. Pattern Matching Table
   d. Graph Algorithms Table

2. Multiple Choice Questions

3. Diagramming Questions
   a. Tree Diagramming (BST, AVLs, 2-4 Trees)
   b. BFS + DFS
   c. Dijkstra’s Algorithm
   d. Prim’s Algorithm
   e. Kruskal’s Algorithm
   f. LCS

4. Coding Questions
   a. Linked List Coding Question
   b. BST Coding Question
   c. Sorting Algorithms Coding Question
   d. Pattern Matching Coding Question
   e. Graphs Coding Question

5. Scenario Questions
## Data Structures Big-O Table

Note: Since these data structures all do different things, some of the categories may not apply (i.e. search for Stack) - write "N/A" if it does not apply. Some of the table is already filled out. We are assuming **worst-case time complexity with amortized analysis** (denoted with an asterisk). Feel free to copy this table and fill it out for average-case analysis.

<table>
<thead>
<tr>
<th></th>
<th>add</th>
<th>remove</th>
<th>search</th>
<th>resize</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>first index</td>
<td>last index</td>
<td>any index</td>
<td></td>
</tr>
<tr>
<td>Arrays</td>
<td>O(1)*</td>
<td></td>
<td></td>
<td>O(n)</td>
</tr>
<tr>
<td>ArrayLists</td>
<td></td>
<td></td>
<td></td>
<td>O(n)</td>
</tr>
<tr>
<td>SLL, no tail</td>
<td>O(1)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SLL, tail</td>
<td></td>
<td>O(n)</td>
<td></td>
<td>O(n)</td>
</tr>
<tr>
<td>CSLL</td>
<td>O(1)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DLL</td>
<td></td>
<td>O(1)</td>
<td></td>
<td>O(n)</td>
</tr>
<tr>
<td>Stack</td>
<td>O(1)</td>
<td></td>
<td></td>
<td>N/A</td>
</tr>
<tr>
<td>Queue</td>
<td></td>
<td></td>
<td></td>
<td>N/A</td>
</tr>
<tr>
<td>BST</td>
<td>O(n)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Heap</td>
<td></td>
<td></td>
<td></td>
<td>N/A</td>
</tr>
<tr>
<td>Hash Map</td>
<td></td>
<td></td>
<td></td>
<td>O(n)</td>
</tr>
<tr>
<td>AVL</td>
<td></td>
<td></td>
<td></td>
<td>O(log n)</td>
</tr>
<tr>
<td>Skip List</td>
<td>O(n)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2-4 Tree</td>
<td></td>
<td></td>
<td></td>
<td>O(log n)</td>
</tr>
</tbody>
</table>
### Sorting Algorithms Table

<table>
<thead>
<tr>
<th>Sorting Algorithm</th>
<th>Big-O</th>
<th>stable?</th>
<th>in-place?</th>
<th>adaptive?</th>
</tr>
</thead>
<tbody>
<tr>
<td>bubble sort</td>
<td></td>
<td>yes</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>cocktail shaker sort</td>
<td></td>
<td></td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>selection sort</td>
<td>$O(n^2)$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>insertion sort*</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>merge sort</td>
<td>$O(n \log n)$</td>
<td></td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>quick sort</td>
<td>$O(n^2)$</td>
<td>no</td>
<td></td>
<td></td>
</tr>
<tr>
<td>radix sort</td>
<td></td>
<td></td>
<td>no</td>
<td></td>
</tr>
</tbody>
</table>

*Insertion sort is not on the final exam for HB

### Pattern Matching Table

<table>
<thead>
<tr>
<th></th>
<th>best-case</th>
<th>worst-case</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>single occur.</td>
<td>all occur.</td>
</tr>
<tr>
<td>Brute Force</td>
<td>$O(m)$</td>
<td>$O(mn)$</td>
</tr>
<tr>
<td>Boyer-Moore (no Galil rule)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>KMP</td>
<td>$O(m + n)$</td>
<td>$O(m + n)$</td>
</tr>
<tr>
<td>Rabin-Karp</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Graph Algorithms Table

For simplicity, represent $|E|$ as E and $|V|$ as V. However, on the exam, make sure to include the cardinality sign!

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Time (or Space) Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adjacency Matrix (space complexity)</td>
<td>$O(V^2)$</td>
</tr>
<tr>
<td>Adjacency List (space complexity)</td>
<td></td>
</tr>
<tr>
<td>Edge List (space complexity)</td>
<td>$O(E)$</td>
</tr>
<tr>
<td>Depth-First Search</td>
<td></td>
</tr>
<tr>
<td>Breadth-First Search</td>
<td></td>
</tr>
<tr>
<td>Dijkstra's Algorithm</td>
<td>$O((V + E) \log V)$</td>
</tr>
<tr>
<td>Prim's MST Algorithm</td>
<td></td>
</tr>
<tr>
<td>Kruskal's MST Algorithm</td>
<td></td>
</tr>
</tbody>
</table>
Multiple Choice Questions

1. Given the following tree, select all options that describe the tree shown. You may need to select more than one answer.

   - Binary Tree
   - Binary Search Tree
   - Heap
   - AVL

2. Assume you have an empty deque backed by an array with initial capacity 7. What is the resulting array after the following operations:
   addFirst(5)
   addLast(7)
   addFirst(8)
   addLast(1)

   a. 
      
      7 1
      8 5

   b. 
      
      8 5
      1 7

   c. 
      
      5 8
      7 1

   d. 
      
      8 5 7 1
3. Given the MinHeap below, what is the resulting array after removing 6 from the heap?

```
null  6  10  44  23  15  60
```

a.  
```
null  10  15  23  44  60
```

b.  
```
null  10  44  23  15  60
```

c.  
```
null  10  15  44  23  60
```

d.  
```
null  10  44  23  15  60
```

4. Suppose you have the HashMap below with the collision strategy of quadratic probing. Each <key, value> pair is <data, data>. Assume the hashcode is the same as the key (e.g. hash(10) = 10). The hashcode is then compressed to fit within the bounds of the HashMap. Suppose you add the data 14. Which index should the data be added to?

```
<table>
<thead>
<tr>
<th>Index</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entry</td>
<td>12</td>
<td>25</td>
<td>4</td>
<td></td>
<td></td>
<td>18</td>
<td></td>
<td>20</td>
<td>10</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

a.  0  
b.  2  
c.  5  
d.  6  
e.  8
5. You are given the starting array [3, 14, 25, 26, 37, 19] and perform an unknown sorting algorithm. After 1 iteration, the array becomes [3, 14, 25, 26, 37, 19]. Which sorting algorithm could have produced the array after 1 iteration? Select all that apply.
   a. Bubble Sort
   b. Selection Sort (selecting for minimum element)
   c. Cocktail Shaker Sort
   d. QuickSort
   e. LSD Radix Sort

6. Perform 2 iterations of LSD Radix Sort on the initial array below. What is the resulting array?

   | 160 | 35 | 49 | 9  | 222 | 85 | 100 |

   a. | 160 | 100| 222| 35 | 85  | 49 | 9  |
   b. | 9   | 35 | 49 | 85 | 100 | 160| 222|
   c. | 35  | 85 | 49 | 9  | 100 | 222| 160|
   d. | 100 | 9  | 222| 35 | 49  | 160| 85 |

7. Suppose you have the text “turingcomplete” and you are searching for the pattern “pnp”. Which pattern matching algorithm below would minimize the number of comparisons?
   a. Brute Force
   b. Boyer Moore
   c. KMP
   d. Rabin Karp (assume BASE is a large prime number)
Diagramming Questions:

Tree Diagramming:

1. BST traversals
   a. Construct a BST so that the Pre-Order traversal is [20, 16, 33, 45, 40]
   b. Construct a BST so that the In-Order traversal is [13, 21, 27, 31, 49]
   c. Construct a BST so that the Level-Order traversal is [6, 3, 10, 1, 5, 13]
   d. Construct a BST so that the Post-Order traversal is [20, 18, 27, 31, 39, 22]

Questions 2-4 apply to the 2-4 tree above. If needed for any operations, use the predecessor node. You should perform the operation based on the result of the previous part (for example, in Q3, you should remove 5 from the result in Q2)

2. Remove 11 from the following 2-4 tree
3. Now, remove 5 and determine the resulting 2-4 tree.
4. Now, remove 20 and determine the resulting 2-4 tree.
Questions 5-8 apply to the tree above. If necessary for any operations, use the successor node.

5. Remove 100 treating the tree as an AVL.

6. (Hard) Using the same tree in the image above and treating it as an AVL (without removing 100), remove 60 instead.

7. Using the same tree in the image above, remove 60 treating it as a BST instead.

8. Using the same tree in the image above, remove 29 treating it as a BST.
For the DFS, BFS, and Dijkstra’s problems below, if there exists a tie on the next vertex to traverse, always choose the vertex that comes first alphabetically.

1. Run BFS on the following graph beginning at vertex B and determine the order in which the vertices are visited.

![Graph for BFS](image1.png)

2. Run DFS on the following graph beginning at vertex H and determine the order in which the vertices are visited.

![Graph for DFS](image2.png)
3. Run Dijkstra’s on the following graph beginning at vertex G and determine
   a. The order in which the vertices are visited.
   b. The distance map to all other vertices

4. Run Prim’s Algorithm on the following graph and determine
   a. The order in which the vertices are visited.
   b. The MST produced

For Prim’s:
If you need to break ties, always add the edge based on which vertex you have added
to the visited set. For example, if you have already visited vertex R and S and need to
break a tie between RG and SA, note that R comes before S alphabetically. So you
would choose RG to traverse first.
For Kruskals:
If you need to break ties, compare edges alphabetically after sorting the edges alphabetically. For example, if you need to break a tie between edges YA and GB, first sort the edges alphabetically to AY and BG. Note that AY comes before BG, so AY should be selected.

5. Run Kruskal's Algorithm on the following graph and determine
   a. The order in which the edges are visited
   b. The MST produced
Do NOT continue adding edges once an MST has been formed. If you need a start vertex, use vertex A.

6. Longest Common Subsequence
   a. List 2 LCS for the two strings below. You must produce the table to receive credit!
      FROZONE
      BRONZE
   b. (Hard) Find 4 LCS for the two strings below. You must produce the table to receive credit!
      GCGAGTCTAT
      CAGGATTAC
      (You will not be asked anything this difficult on the final but if you can do this problem correctly, you should feel comfortable with LCS!)
Coding Questions

Linked List Coding Question

Implement the functions addLast and removeLast for a **Singly-Linked List with a Tail**. Here is some starter code:

```java
public class LinkedList {
    int size;
    Node head;
    Node tail;

    class Node {
        int data;
        Node next;

        public Node(int data, Node next) {
            this.data = data;
            this.next = next;
        }
    }

    public void addLast(int data) {
        // For you to implement!
    }

    public int removeLast() {
        // For you to implement!
        // returns the data that was contained in the Last Node.
    }
}
```
A node in a BST is a **proud parent** if the sum of all of its children (including grand-children, great-grand-children, etc.) is divisible by 4 and greater than 0.

For example, in the following tree, the nodes at 10 and 12 are both proud parents:

- Node 10: $3 + 1 + 4 + 12 + 11 + 13 = 44$ (divisible by four!)
- Node 12: $11 + 13 = 24$ (divisible by four!)

Write a function, `countProudParents()`, that counts the number of proud parents in a BST. You must implement this recursively. Notice the class `IntegerWrapper` - we'll have to keep track of two values during recursion, the sum of a node's children as well as the total number of proud parents. Using the `IntegerWrapper` will let you keep an integer value across different function calls, how should you use this class for your solution? You can instantiate a `IntWrapper` with the default constructor and set the value attribute using the `.value` attribute. Here is some starter code:

```java
public class BST {
    int size;
    Node root;

class Node {
    int data;
    Node right;
    Node left;
    // constructor not shown
}

// counts the number of nodes whose children's sum is nonzero and divisible by four.
public int countProudParents() {
    // For you to implement!
}
}

class IntWrapper {
    int value;
}
```
**Sorting Algorithms Coding Question**

You are given the following starter code for the QuickSort algorithm. Fill in the code so that this code is a valid QuickSort algorithm.

```java
import java.util.Random;
public class Sorting {

    public static void swap(int[] arr, int a, int b) {
        int temp = arr[a];
        arr[a] = arr[b];
        arr[b] = temp;
    }

    public static void quickSort(int[] arr) {
        if (arr == null || arr.length <= 1) {
            return;
        } else {
            quickSort(arr, 0, arr.length);
        }
    }

    private static void quickSort(int[] arr, int start, int end) {
        if (Math.abs(start - end) <= 1) {
            return;
        } else {
            int pivotIndex = rand.nextInt(end - start) + start;
            int pivot = arr[pivotIndex];
            swap(arr, start, pivotIndex);
            int i = start + 1;
            int j = end - 1;

            // Your code here!
            // Your code here!
            // Your code here!

            // Restore pivot and recurse
            swap(arr, start, j);
            quickSort(arr, start, j);
            quickSort(arr, j + 1, end);
        }
    }
}
```
Pattern Matching Coding Question

KMP Failure Table Coding

Given the following code snippets for the KMP Failure Table algorithm, fill in the blanks in the provided code with the appropriate answer. The method should build and return the failure table used in the KMP pattern matching algorithm. You can use more than 1 line of code for question 4 and question 5 if necessary, **but only for those questions**.

```java
public static int[] buildFailureTable(CharSequence pattern, CharacterComparator comparator) {
    int[] failureTable = 1._________
    int i = 0; //Hint: Base Case?
    int j = 1;
    while (3. ______) {
        if (4. ____________________) {
            i++;
            failureTable[j] = i;
            j++;
        } else if (i == 0) {
            5. ________
        } else {
            6. ________
        }
    }
    return failureTable;
}

Hint: try to make the failure table for a pattern, i.e. ababcdcd, and think of what is happening in terms of code to get those values
```
Graphs Coding Question

Implement the BFS function with the following starter code.

```java
public class Graph {

    class Vertex {
        int data;
        // constructors and other methods not shown
    }

    class Edge {
        Vertex u;
        Vertex v;
        int weight;
        // constructors and other methods not shown
    }

    class Graph {
        Set<Vertex> vertices;
        Set<Edge> edges;
        Map<Vertex, List<Vertex>> adjacencyList;
        // constructors and other methods not shown
    }

    class Queue<T> {
        public void push(T data) { /* implementation not shown */ }
        public T pop() { /* implementation not shown */ }
        public int size() { /* implementation not shown */ }
    }

    // returns an ArrayList with vertices in order of exploration for the BFS algorithm.
    public static ArrayList<Vertex> bfs(Vertex start, Graph graph) {
        // For you to implement!
    }
}
```
Graphs Coding Question Pt 2

Prim's/Kruskal's Coding

Below you are given a bank of unordered code lines from A to Q where some of the code lines from the code bank complete Prim's or Kruskal's algorithm as implemented in the homework.

Students in HB's section should write the order of the steps for Prim's algorithm. Students in Moss' section should write the order of the steps for Kruskal's algorithm.

Write the letter corresponding to each line in the order they are to be called within the method. Separate individual letters in the list using spaces. For example, your answer should be in the format of: “R S T U V W X Y”

NOTE: You will not need to use every step. No step is used more than once. There are multiple correct orderings.

A few points to note:
- Your ordering should be based on the homework implementation
- Each edge stores an edge weight and a pair of vertices (u, v).
- Assume that the edge set output is initialized for both Prim's and Kruskal's.
- Assume the disjoint set have already been initialized for Kruskal's.
- For Kruskal's, recall that two sets are disjoint if they have no element in common. This is equivalent to checking if the representative of the sets are different (using the find() method).

A. Create a while loop that terminates when the MST has 2* (|V| - 1) or the data structure that stores edges is empty
B. Create a queue and add all edges adjacent to the start vertex to the queue
C. For all edges adjacent to u, add to the data structure that stores edges only if the corresponding pair v has not already been visited
D. Return the edge set output if the size is 2 * (|V| - 1), else return null
E. Create a priority queue and add all edges adjacent to the start vertex to the priority queue
F. Union each vertex in the edge
G. Traverse back to the beginning of the while loop and recheck the terminating condition
H. Create a priority queue and add all edges in the graph to the priority queue
I. Return the edge set output
J. Add edge and reverse edge to MST edge set output if the vertex u and vertex v are in disjoint sets
K. Create a while loop that terminates when the data structure that stores edges is empty
L. Create a while loop that terminates when the MST has $2^* (|V| - 1)$
M. Remove the first element in the data structure that stores edges
N. Add edge and reverse edge to MST edge set output if the vertex $v$ has not been visited before (you have already visited vertex $u$)
O. Create a queue and add all edges in the graph to the queue
P. Add edge and reverse edge to MST edge set output
Q. Add edge and reverse edge to MST edge set output if the vertex $u$ and vertex $v$ are not in disjoint sets
R. Initialize a visited set and add the start vertex
S. Add the edge vertex $v$ to the visited set (you have already visited vertex $u$)
Scenario Questions:

1. Suppose you are a coffee shop owner, and you want to determine all nearby shops. You want to drive to all closest shops first, then the shops closest to those, and so on in a 2 mile radius. Which graph algorithm best describes this scenario?

2. Your algorithm in the previous question visited all nearby shops but did not record the minimum distance to those shops. Which graph algorithm would best solve this problem?

3. In the previous 2 questions, you mapped out all nearby shops and the distances to those shops. Now, you want to record a path to all nearby shops starting at your shop but don’t want to include any unnecessary roads in your graph. Which graph algorithm best describes this scenario?

For questions 4 - 8, you have the following list of options:

- a. Array
- b. ArrayList
- c. Singly-Linked List
- d. Doubly-Linked List
- e. Circular Linked List
- f. Stack
- g. Queue
- h. Deque
- i. Tree
- j. Binary Tree
- k. Binary Search Tree
- l. Min Heap
- m. Max Heap
- n. SkipList
- o. AVL
- p. 2-4 Tree
- q. HashMap
4. GT Athletics is assigning seats to the football games in a new way to use social distancing but to also get more fans in the stands. The stadium has the capacity to have thousands of seats in the stands for a game, but they are trying something new. All fans will pay a single price to get into the game. When the fans arrive at the stadium, they are assigned a number that always ends in a zero and is preceded by a random number of 1 digits. The more 1’s a fan gets, the higher up the fan’s seat is. The fans are also arranged alphabetically by name in the system and added and removed as they show up and leave the game. What structure is best suited to store this information?

5. bEtsy is a new online store that connects creative, hand crafted goods from vendors with customers. They have vendors all over the country making goods to sell. bEtsy keeps track of the vendor sales each month in a special system. The vendor sales data is entered at the end of the month. The system quickly identifies what vendor has the largest sales that month. With this information, bEtsy executives offer the vendor discounts for using their online store. What structure is best suited to store this information?

6. AutoKarma is a new car sales business that sells cars online. They have storage facilities all over the country filled with cars. AutoKarma keeps track of its inventory by assigning each car a unique vehicle ID. The data structure that holds the ID and car info allows the company to retrieve the vehicle information very quickly. What structure is best suited to store this information?

7. Single-celled organisms often reproduce by splitting in half, forming two copies of the original organism. Each of the copies also can split, but some of the organisms die before they are able to do this. What structure best represents the hierarchy of reproduction?

8. Ringples Chips have recently started selling chips in an innovative new packaging. Instead of a bag, chips are sold in a long cylinder that can be opened from either end, and chips are stacked together on the inside of the cylinder. Because the cylinder is only wide enough for the chips to barely fit, you can only access the chips at the extreme ends of the cylinder. What structure is best suited to represent the new packaging?