Spring 2023 CS 1332 StudyPalooza Worksheet

Link To Doc: https://bit.ly/3At3vXg

CS 1332 FINAL EXAM TOPIC LIST

- Arrays
- Lists
    - ArrayLists
    - LinkedLists
        - Singly
        - Doubly
        - Circular Singly
        - Circular Doubly
- Linear Data Structures
    - Stacks, both linked and array-backed
    - Queues, both linked and circular array-backed
    - Deques, both linked and circular array-backed
- Binary Trees
    - Shape Properties (balanced, full, complete, degenerate)
    - Traversals (preorder, inorder, postorder, levelorder)
- Binary Search Trees
    - Operations: search, add, remove
- Binary Heaps
    - Operations: add, remove, upheap, downheap
    - BuildHeap
- HashMaps
    - Closed Addressing: External Chaining **(With coding)**
    - Open Addressing: Linear Probing & Quadratic Probing
    - Operations: search, add, remove, resize
- SkipLists
    - Operations: search, add, remove
- AVLs
    - Balance information (heights & balance factors)
    - Rotations (single & double)
    - Operations: search, add, remove
- 2-4 Trees
    - Operations: search, add, remove
    - Overflow Handling: promotion
    - Underflow Handling: transfer & fusion
- Sorting

    - Bubble
    - Selection **(With coding)**
    - Insertion
    - Cocktail Shaker **(With coding)**
    - Merge **(With coding)**
    - Quick
    - QuickSelect **(With coding)**
    - LSD Radix **(With coding)**
    - Heap **(With coding)**
    - Properties of all sorting algorithms above
        - Stability
        - Adaptivity
        - In-Place vs Out-of-Place
- Pattern Matching
    - Brute Force
    - Boyer-Moore
        - without Galil rule **(With coding)**
        - with Galil rule
    - KMP **(With coding)**
    - Rabin-Karp **(With coding)**
- Graphs
    - All graph terminology
    - Breadth-First Search **(With coding)**
    - Depth-First Search
        - Recursive version **(With coding)**
    - Dijkstra's Shortest Path **(With coding)**
    - Prim's MST **(With coding)**
    - Kruskal's MST
- Dynamic Programming
    - Longest Common Subsequence
- Big O for all of the topics above

This doc samples questions from various topics and is not cumulative. For practice over specific topics, go to Canvas > Files > Resources > Additional Resources > PLUS Session Worksheets

Big O

Data Structures Table

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.

	add			remove			search		resize
	First index	Last index	Any index	First index	Last index	Any index	At index	For Specific Value	
ArrayLists	O(n)								
SLL, no tail									N/A
SLL, with tail				O(1)					
CSLL						O(n)			
DLL with tail							O(n)		
Stack (array-backed)								N/A	
Queue (array-backed)		O(1)*							
Deque (array-backed)									O(n)

Data Structures Table (continued) - Assume **Worst Case** but feel free to make a table for average case

	add	search	remove
BST			
Heap			
HashMap with External Chaining			
HashMap with Linear/Quadratic Probing			
AVL			
SkipList			
2-4 Tree			

Sorting Algorithms Table

	Big-O Best Case	Big-O Worst Case	stable?	in-place?	adaptive?
Bubble Sort			Yes		
Insertion Sort					
Selection Sort		$O(n^2)$			
Cocktail Shaker Sort					
Merge Sort				No	
Quick Sort					
LSD Radix Sort	O(kn)				
HeapSort					

Pattern Matching Table

	Best Case		Worst Case	
	Single occurrence	All occurrences	Single occurrence	All occurrences
Brute Force				
Boyer-Moore (no Galil Rule)				
Boyer-Moore with Galil Rule (no good suffix rule)				

KMP				
Rabin Karp				

Graphs Table

Data Structure or Algorithm	Worst Case Time or Space Complexity
Adjacency Matrix (space complexity)	
Adjacency List (space complexity)	
Edge List (space complexity)	
Depth First Search	
Breadth-First Search	
Dijkstra’s Algorithm	
Prim’s Algorithm	
Kruskal’s Algorithm	

Big O Practice:

Refer to Socrative Polls, previous exams, Kahoots, 30+ questions from Exam 1 worksheet, Exam 2 worksheet, and Exam 3 worksheet.

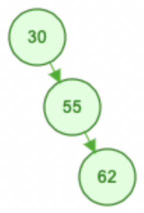
One question which you should know: What is the runtime of running the LCS algorithm on any arbitrary text or pattern.

T/F:

1. The data in an ArrayList must be contiguous but the data in an array does not.
2. In a SLL-backed Queue, elements are added to the back and removed from the front
3. The location of adding to the front of an array-backed deque is (Front - 1) % array.length
4. The average space complexity of a SkipList assuming it was constructed using a fair coin and the number of levels is capped at log(n) is O(nlog(n))
5. Heaps are always full, complete, and balanced
6. KMP performs better for large alphabets with repeating characters in the text while Boyer Moore performs better for small alphabets.
7. Running Dijkstra's on the MST produced by either Prim's or Kruskal's will return the same result as running Dijkstra's on the original graph.

Multiple Choice Questions

1. Given the following tree, select all options the tree could be. **You may need to select more than one answer.**



   a. Binary Tree
   b. Binary Search Tree
   c. Heap
   d. 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 <14, 14>. Which index should the data be added to?

Index	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Entry		12		25	4			18		20	10

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 (1 iteration is putting pivot in the right place)
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

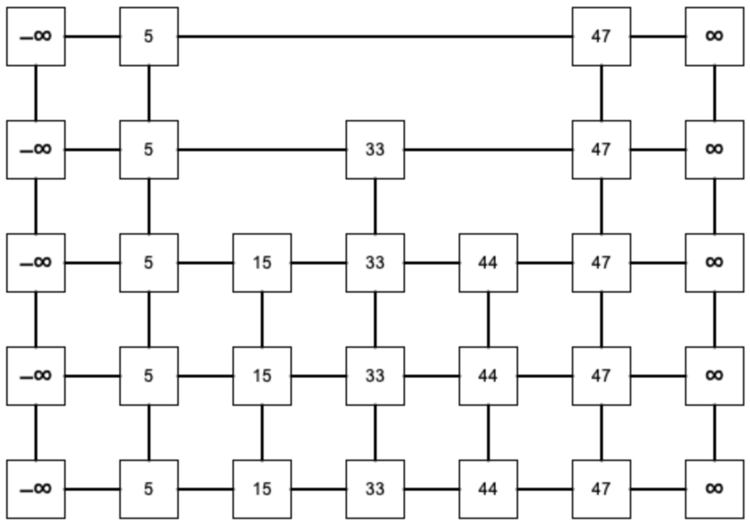
c.

35	85	49	9	100	222	160

d.

100	9	222	35	49	160	85

7. Given the following SkipList, select the path taken to determine whether 46 exists. Only nodes that are traversed over should be included in the path. The nodes in the path are written as (level number, data).

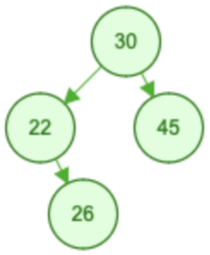


a. (4, -∞), (4, 5), (4, 47), (3, 47), (2, 47), (1, 47), (0, 47)
b. (4, -∞), (4, 5), (3, 5), (3, 33)
c. (4, -∞), (4, 5), (3, 5), (3, 33), (2, 33), (2, 44), (1, 44), (0, 44)
d. (4, -∞), (3, -∞), (2, -∞), (1, -∞), (0, -∞), (0, 5), (0, 15), (0, 33), (0, 44)

8. Give the following scenario, select the sorting algorithm that would perform best. Assume that the integers are being sorted in ascending order:
Efficiency is prioritized and no memory constraints exist. The relative order of duplicates must be maintained. The input data has 20 elements, with the largest element being 1234567890 and the smallest element being 0.

a. QuickSort
b. MergeSort
c. LSD Radix Sort
d. Bubble Sort

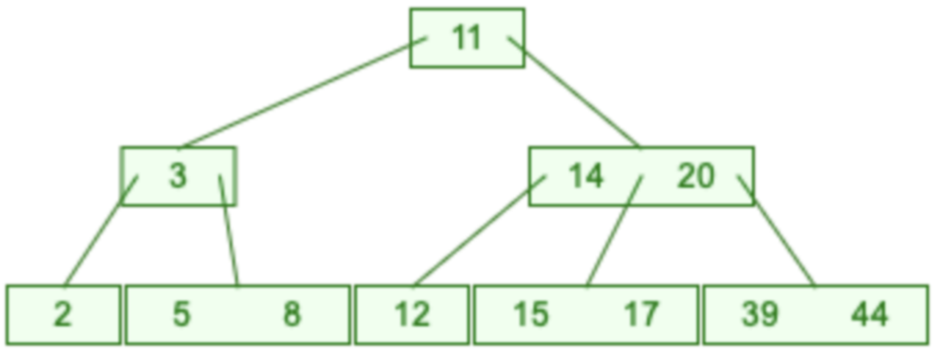
9. Given the following AVL and the add operation, describe the sequence of rotation operations that are performed after the add operation.
Add(24)



a. Right-Left rotation
b. Left-Right rotation
c. Right rotation
d. Left rotation
e. No rotations needed

Diagramming Questions

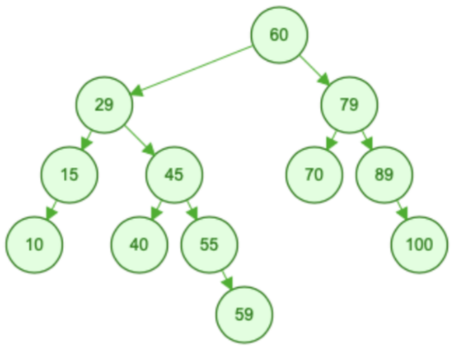
1. Create a MaxHeap from the following list of data: [25, 11, 84, 33, 46, 50, 77, 100]
Tree Diagramming:
2. BST traversals (Challenge!)
a. Construct a BST so that the Pre-Order traversal is [20, 16, 33, 45, 40]
b. Construct a BST so that the In-Order traversal is [13, 21, 27, 31, 49]
c. Construct a BST 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]



.

3. Questions a-c 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 b, you should remove 5 from the result in a)

   a. Remove 11 from the following 2-4 tree
   b. Now, remove 5 and determine the resulting 2-4 tree.
   c. Now, remove 20 and determine the resulting 2-4 tree.



Questions 4-7 apply to the tree above. If necessary for any operations, use the **successor** node.

4. Remove 100 treating the tree as an AVL.

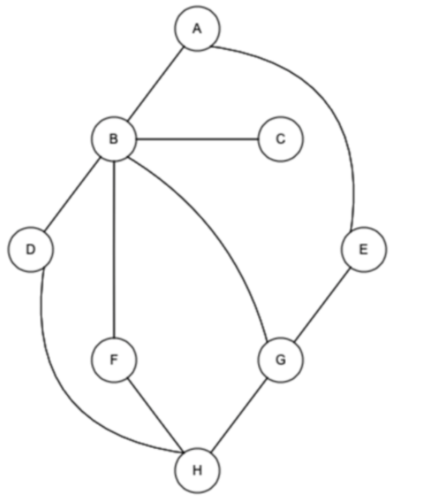
5. (Hard) Using the same tree in the image above and treating it as an AVL (without removing 100), remove 60 instead.

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

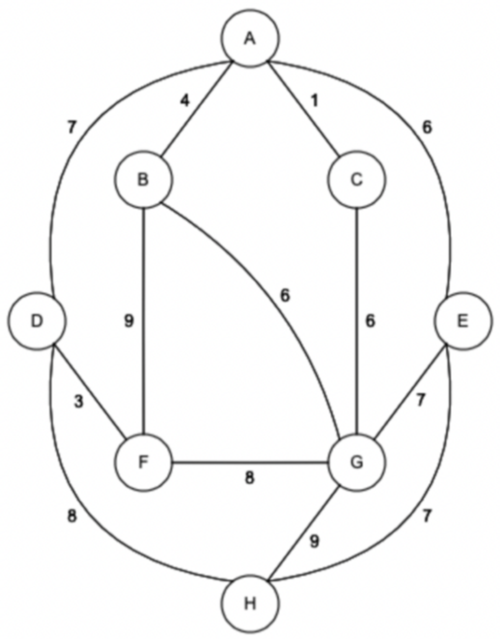
7. Using the same tree in the image above, remove 29 treating it as a BST.

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.

8. Run DFS and BFS on the graph below beginning at vertex B and determine the order in which the vertices are visited.



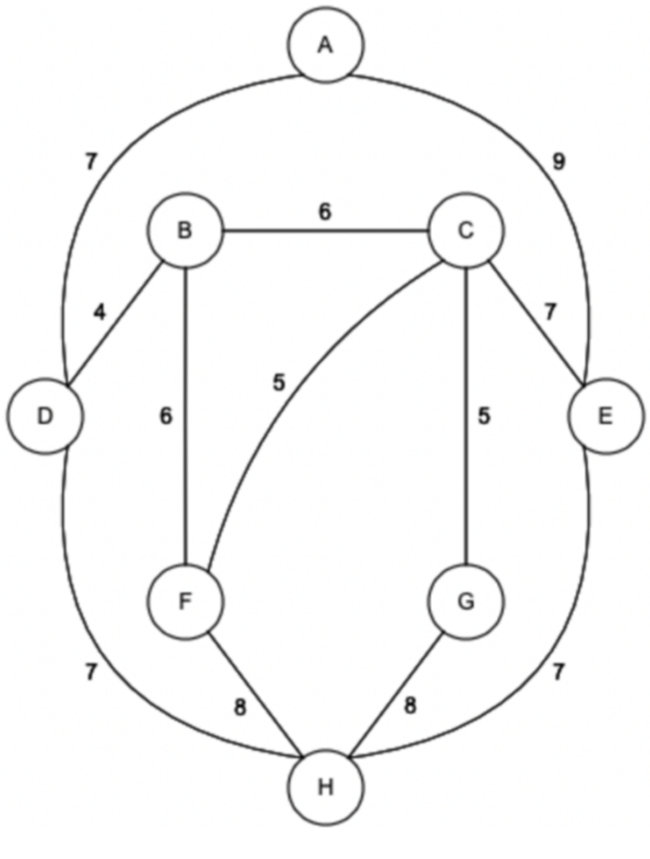
9. Run Dijkstra's on the graph below beginning at vertex G and determine
   a. The order in which the vertices are visited.
   b. The distance map to all other vertices



10. Kruskal's Algorithm
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.
Run Kruskal's Algorithm on the following graph and determine

a. The order in which the edges are visited
b. The final MST produced

Do NOT continue adding edges once an MST has been formed. If you need a start vertex, use vertex A.



11. Longest Common Subsequence

a. List 2 LCS for the two strings below. You must produce the table and fill in the entries.
   FROZONE
   BRONZE
b. Find the LCS for the two strings below. You must produce the table and fill in the entries.
   BRUNCHING
   BRORANCH

Coding Questions

1. Closed Addressing: External Chaining **(With coding)**
2. Selection **(With coding)**
3. Cocktail Shaker **(With coding)**
4. Merge **(With coding)**
5. QuickSelect **(With coding)**
6. LSD Radix **(With coding)**
7. Heap **(With coding)**
8. Boyer-Moore without Galil rule **(With coding)**
9. KMP **(With coding)**
10. Rabin-Karp **(With coding)**
11. Breadth-First Search **(With coding)**
12. Depth-First Search **- Recursive version (With coding)**
13. Dijkstra's Shortest Path **(With coding)**
14. Prim's MST **(With coding)**

For the algorithm questions (2 - 14), there is really only one way to implement the algorithm **as taught in this class**. Refer to your homework and consider redoing any problem you need practice on!

For the only data structure question (1), know the general approach of the methods (put, remove, get, containsKey, keyset, etc.) and be prepared to extend what you know to a new method since you could be asked something not directly on the HW. For example, you could be asked to implement methods like

- entrySet (see Exam 3 PLUS Session Document here or on Canvas)
- getOrDefault()
- containsValue()
- putIfAbsent()
- replace()

All methods above can be found in the documentation.