ComSSA DSA Exam Revision

Updated: 9th November, 2019

Note: This worksheet was put together by the ComSSA student volunteers. These questions are based around examples from previous tests, exam papers and the lecture materials. This worksheet is designed to be for learning purposes, not as a mock assessmen.

1. Stacks & Queues

(a) Convert the following infix expression to their postfix equivalents using a stack. In your answer, show each step of building the postfix expression, including the contents of the operator stack and the postfix output built up so far at each step.

(i) (A - B / C * (D + E/F)) + G * H

Infix Char	Postfix Written So Far	Operator Stack Contents

2. General & Recursion

(a) What are the 3 necessary properties of a Recursive algorithm

(b) What is the purpose of a *wrapper method*?

(c) Write a Recursive function (and wrapper if necessary) to calculate the value of the i'th fibonacci number. (For imported integer i) Fibonacci sequence is {0,1,1,2,3,5...}

(d) When reading in a file of customer purchases, each with a unique customer ID. What sort of data structure or abstract data type should you store all your Customer objects in for maximum efficiency.

(e) Recursion adds the risk of stack overflow. Why would we ever use a recursive solution over an iterative one?

3. Sorting

(a) What is the difference between an in-place and a not in-place sort. Give an example of each

(b) What is the difference between a stable and an unstable sort. Give an example of each

(c) Why would a **leftmost** element pivot selection strategy perhaps be chosen over a **median of 3** strategy? Why would it be better to instead take a **middle** element pivot selection strategy rather than a leftmost? Try to discuss the **pros and cons** of each strategy

(d) What sort of time complexity does each type of array element ordering make each sorting algorithm have? (Assume quick sort is using 'right most' pivot selection)

Table 1: Sorting Algorithm Time Complexities

	Sorted	Reverse Sorted	Random
Bubble			
Insertion			
Selection			
Merge			
Quick			

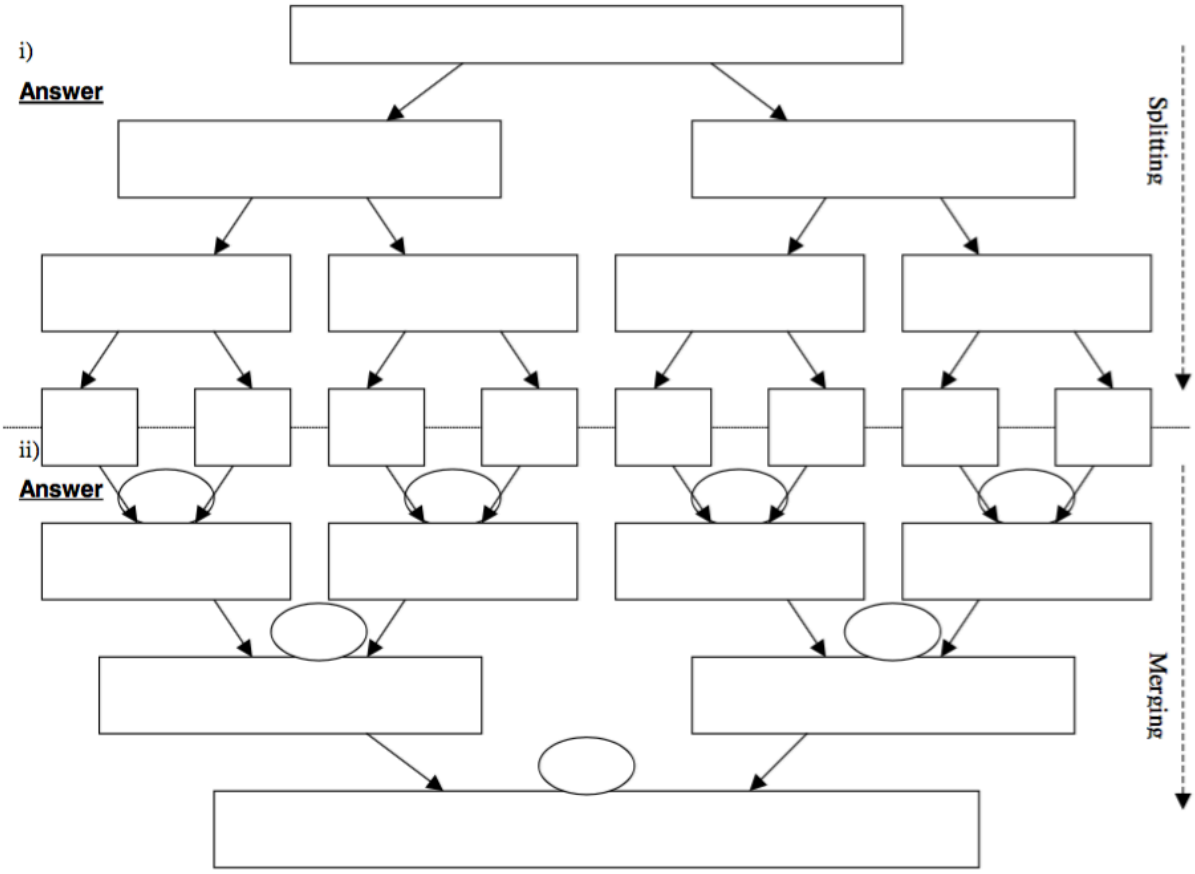
(e) Given the following array of numbers show how MergeSort would sort them:

54, 32, 9, 15, 58, 84, 18, 70

(i) Fill in the recursion tree depicting the splits and sub-arrays to be processed at each branch (do not show the merging of the sub-arrays yet though).

(ii) When the recursion unwinds, MergeSort will begin merging the different sub-arrays, comparing elements as it merges. Complete your diagram from part (i) so that merges are shown

(iii) In the ellipses provided in the diagram, write down how many comparison operations (ie: x<y) are involved at each merging branch.



Note: Don't forget to fill in the number of comparisions in the ellipses on the above diagram

4. Linked List, Iterators

(a) How do the following differ in their configurations and time complexities when inserting and removing from the end? (Use diagrams to assist in your explanation)

(i) Singly ended, singly linked

(ii) Doubly ended, singly linked

(iii) Doubly ended, doubly linked

(b) Which kind of linked list would be the bare minimum for implementing a queue and why? What about the bare minimum while being the most efficient?

(c) Why is traversing a linkedlist from start to finish more efficient using an iterator rather than accessing each element index by index.

(d) What would the benefit be to include an iterator in a Queue or a Stack?

5. Binary Search Trees

(a) Define the terms complete, almost complete and completely degenerate in regards to Binary Search Trees.

(b) Draw the binary tree that results from inserting each of the numbers in the order shown. State if the tree created is Complete, Almost-complete or Degenerate. For each write the Pre, In and Post order transversal.

(i) 56, 78, 60, 36, 41, 91, 18, 3, 19, 10. Then remove 3

(ii) 'D', 'F', 'B', 'G', 'A', 'E', 'C'.

(iii) 41, 49, 44, 47, Then remove 41

(c) What is the worst case time complexity for finding an element in a binary search tree? Why? Include a diagram in your explanation. What data structure is this type of tree similar to?

(d) The recursive insert and delete methods for Binary Trees return the current node they were 'updating' after ever level of recursion. How can this be utilised in the wrapper method?

(e) Write the code (including the wrapper method) for post order traversal (via printing the keys of each node) of a binary tree.

(f) When deleting a node you must choose a successor. The following code is broken, how would you fix it?

```
SUBMODULE promoteSuccesor
IMPORT: cur
EXPORT: successor

1. successor = cur
2. IF cur.getLeft == null
3.    successor = cur
4. ELSE
5.    IF cur.getLeft NOT null
6.       successor = promoteSuccessor <- cur.getLeft
7.       IF successor == cur.getLeft
8.          cur.setLeft <- successor.getLeft
9.       ENDIF
10.   ENDIF
11.ENDIF
```

6. Advanced Trees

(a) Draw the red black tree, 2-3-4 tree and B tree with a max degree of 5 (block size of 4) that correspond with the insertion following sequences. You may choose to simply to do one of each tree.

(i) 56, 36, 91, 45, 84, 34, 28, 64, 77, 13 (Recommend 2-3-4 Tree)

(ii) 13, 21, 30, 44, 32, 25 (Recommend RB Tree)

(iii) 47, 60, 52, 33, 49, 90, 67, 50, 51 (Recommend B Tree)

7. Heaps

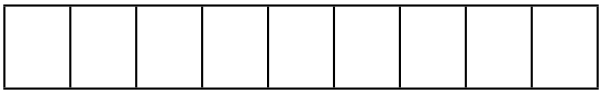
(a) Given the following list of numbers:

57, 31, 25, 41, 81, 59, 37, 24

(i) Draw the heap that results from each of the numbers being inserted into a max heap

(ii) Represent the heap as an array

(iii) Show the heap as an array after inserting "43"



(b) State the time complexity of sorting an array of N items with heapsort. Can you explain why this is so?.

(c) What is the difference between a min and a max heap? Could you use a max heap as a min heap via removing from the bottom instead of the top?

8. Hash Tables

(a) Given the following hash function:

```
SUBMODULE: HASH
IMPORT: key (integer)
EXPORT: hashIdx (integer)
ALGORITHM:
        hashIdx = key MOD table.length;
```

Using the above hash function, insert the following key:value pairs into the hash table below. Use the method of **quadratic probing** to handle collisions.

28:"First" 21:"Second" 6: "Third: 39:"Fourth" 10:"Fifth"

Index	Key	Value
0		
1		
2		
3		
4		
5		
6		
7		
8		
9		
10		

Note: Remember that each step after collision starts from the original index. Each step just has larger and larger step sizes

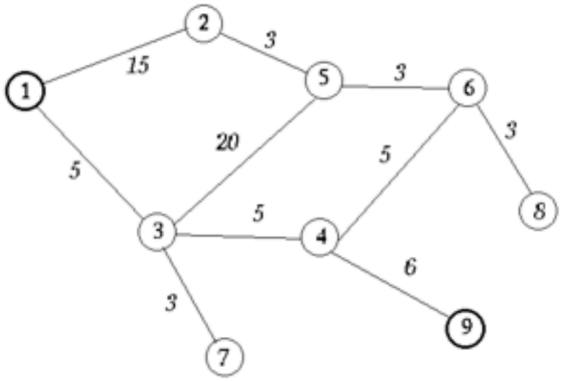
(b) List the four properties of a good hash function

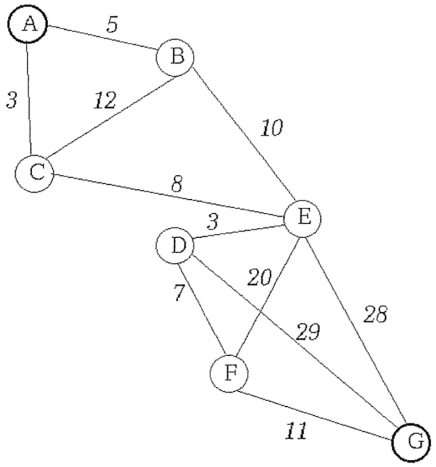
(c) Explain the difference between primary and secondary clustering.

(d) Explain how Double Hashing collision handling works, and how it avoids primary and secondary clustering.

9. Graphs

(a) Convert the following graphs into both an adjacency list and adjacency matrix format. Then, perform a depth-first and breadth-first search, showing the **list of edges** e.g. {(A,B),(B,D)} in the search tree. Assume nodes are sorted alphabetically.





End of Worksheet