DSA Quizzes Safko Summer 2023

Quiz 1: Linked Lists

Question 1: Given a head pointer, discuss in detail the steps for adding a node to the head of the list. Also discuss the process when there is a single node in the list, and when the list is empty.

If there is only one node in the list, the new node will become the new head, while the existing node will be the second node in the list. Create a new node with the desired data value. Set the "next" pointer of the new node to point to the existing node. Set the head pointer to point to the new node. If the list is empty, there are no existing nodes. In this case, the new node becomes the head of the list. Create a new node with the desired data value. Set the "next" pointer of the new node to NULL (since there are no other nodes). Set the head pointer to point to the new node.

Question 2: Discuss in detail the process for deleting a node from the middle of a linked list.

To delete a node from the middle of a linked list, we'll have to go through some steps. First, we'll have to locate the node to be deleted. This involves traversing the linked list starting from the head node. Keep track of the current node and the previous node as you move through the list. You'll stop traversing when you find the node to be deleted or reach the end of the list. Next, we'll adjust the pointers after locating the node that needs to be deleted. We'll set the next pointer of the previous node to point to the node after the one being deleted. If the node is the head and that node is being deleted, update the head pointer to the next node. As a small adjustment of this process, you can free the memory that was occupied by the node by adjusting the pointers. This will avoid any memory leaks.

Quiz 2: ArrayLists, LinkedLists

Question 1: In class we discussed the Stack and Queue Data Structures. Say a few words about stack and queues. Discuss the difference between pop and top, and discuss the acronyms LIFO and FIFO. Also, give real-life examples of a stack and a queue. Note: you can't use plates at a buffet or waiting in line for tickets as your examples

Answer: A stack is similar to a stack of CDs, the last in is the first out. A stack is a LIFO structure for that exact reason. LIFO being 'last in first out'. A queue is similar to a car wash, where the first in and on the conveyor belt is the first out. A queue is a FIFO structure for that exact reason. FIFO being 'first in, first out'. Pop() removes the element at the head, beginning, or top of a structure. Top() is similar to peek where it looks at whatever is at the top and returns it.

Question 2: A collection object (such as a stack or queue) can be written using either java.util.ArrayList or java.util.LinkedList. Discuss the pros and cons of using java.util.ArrayList or java.util.LinkedList for a collection. Do not use antipodal (opposite) reasons.

Answer:
Pros of java.util.ArrayList:
Is an array, and contiguous
It inherits from List class
It's better for storing and accessing

Cons of java.util.ArrayList:
Slow
With every deletion, the array needs to be rebuilt
Default capacity of 10

Pros of java.util.LinkedList:
Fast
It uses doubly linked list
It can act as a list and a queue
It's better for data manipulation

```
Cons of java.util.LinkedList:
Not contiguous
```

Quiz 3: Big O, Sorting Algorithms

Question 1: Given the following big-O notation time complexities: O(n lg n) O(n!) O(n) O(n2) Sort these in order from least time complexity to greatest time complexity

From least time complexity to greatest time complexity:

O(n) < O(n lg n) < O (n^2) < O(n!)

Question 2: Discuss in detail the big O time complexity value for linear search, and the big O time complexity value for binary search, and why those values are correct for those searching algorithms 2 positive reasons for using an array, and 2 negative reasons for using an array as a container to add elements to the beginning of the array and at the end of the array.

For Linear Search: Big O Time Complexity: O(n) A linear search is a simple searching algorithm that sequentially checks each element in a list until a match is found or the whole list has been traversed. The time complexity of a linear search is O(n) because, in the worst-case scenario, it may need to iterate through all n elements in the list to find the desired element or determine that it doesn't exist. The time it takes to find the element grows linearly with the size of the list.
Binary Search:
Big O Time Complexity: O(log n) Binary search is an efficient searching algorithm that works on sorted arrays. It repeatedly divides the search space in half by comparing the middle element with the target element. If the middle element is the target, the search is successful. Otherwise, the search continues in the left or right half, depending on whether the target element is smaller or larger than the middle element. The time complexity of binary search is O(log n) because with each step, the search space is reduced by half. This makes binary search much faster

than linear search, especially for large lists, as the number of remaining elements to be searched decreases exponentially.
2 positive reasons for using an array is that you can have random access to elements and the memory can be contiguous. 2 negative reasons to use an array for containers is insertion at the beginning and the insertion at the end (which is dynamic arrays).

Question 3: Given the following eight (8) element array:

14, 52, 35, 16, 72, 11, 48, 51

What will the array look like after the first pass of a bubble sort?

- **11, 52, 35, 16, 72, 14, 48, 51**
- **14, 35, 16, 52, 11, 48, 51, 72**
- **11, 14, 52, 35, 16, 72, 48, 51**
- **14, 52, 16, 35, 11, 72, 48, 51**

Answer:
14, 35, 16, 52, 11, 48, 51, 72

Question 4: Given the following eight (8) element array:

14, 52, 35, 16, 72, 11, 48, 51

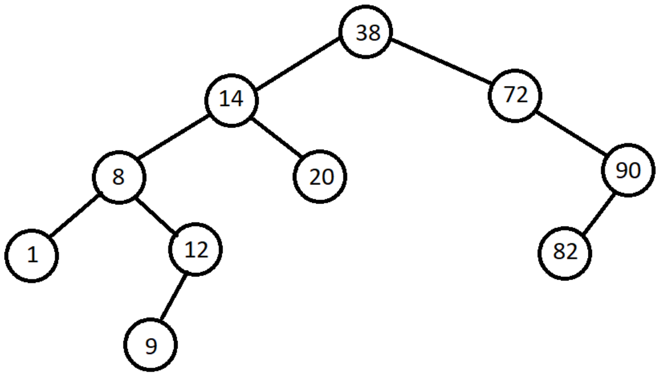
What will the array look like after the first pass of a selection sort?

- **11, 52, 35, 16, 72, 14, 48, 51**
- **14, 35, 16, 52, 11, 48, 51, 72**
- **11, 14, 52, 35, 16, 72, 48, 51**
- **14, 52, 16, 35, 11, 72, 48, 51**

Answer:
11, 52, 35, 16, 72, 14, 48, 51

Quiz 4: Binary Search Trees

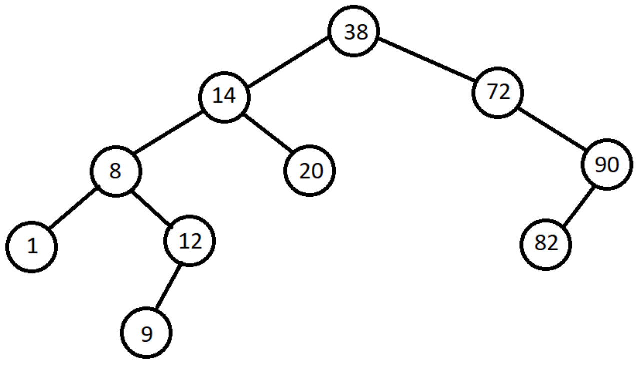
Question 1: Given the following BST:



Discuss the process for adding the number 15 to this tree

Answer: The process in adding the number 15 to this tree involves you having to check whether the node value is greater or less than the value you want to add. For example, since 15 is less than the root node (38), we will be going to the left side of the tree. Now we perform a check on 14. Since 15 is greater, we move to the right of 14. Now we have to perform a check on 20. Since 15 is less than 20, we will add the node 15 to the left of 20. Therefore, the process is completed and the node has been successfully added to the tree.

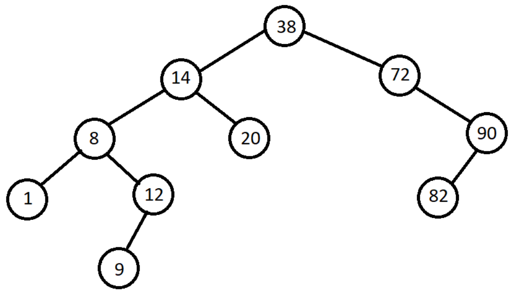
Question 2: Given the following BST:



Discuss any process of your choice for deleting the number 72 from this tree

Answer:
When a parent node is deleted, the first child/successor next in line becomes that parent only if it's value can fit in as a valid replacement (i.e. is greater or less than the parent above it) in this case since 90 is greater than 38 it can simply move up one node and replace 72 upon its deletion. This will move the right side of the BST up by one node effectively making 90 the child of 38, and 82 the child of 90.

Question 3: **Given the following BST:**



Which of the following is the correct output for a Pre-Order Traversal of this tree?

1-8-9-12-14-20-38-72-82-90

1-9-12-8-20-14-82-90-72-38

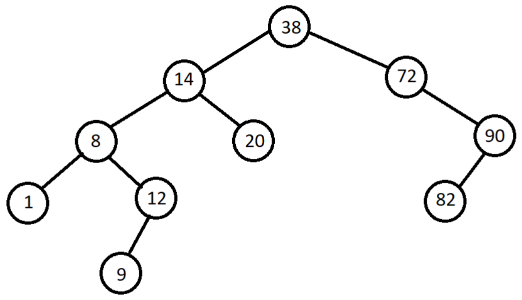
38-14-8-1-12-9-20-72-90-82

38-14-72-8-20-90-1-12-82-9

Answer:

38-14-8-1-12-9-20-72-90-82

Question 4: **Given the following BST:**



Which of the following is the correct output for a Post-Order Traversal of this tree?

1-8-9-12-14-20-38-72-82-90

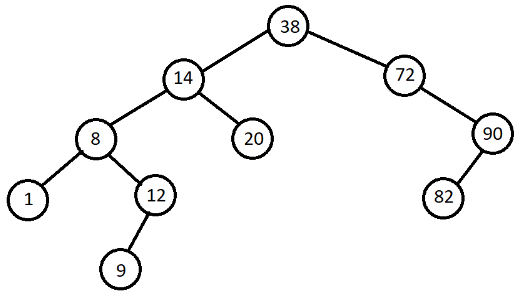
1-9-12-8-20-14-82-90-72-38

38-14-8-1-12-9-20-72-90-82

38-14-72-8-20-90-1-12-82-9

Answer:
1-9-12-8-20-14-82-90-72-38

Question 5: **Given the following BST:**



Which of the following is the correct output for an In-Order Traversal of this tree?

1-8-9-12-14-20-38-72-82-90

1-9-12-8-20-14-82-90-72-38

38-14-8-1-12-9-20-72-90-82

38-14-72-8-20-90-1-12-82-9

Answer:
1-8-8-12-14-20-38-72-82-90

Quiz 5: Trees, Graphs, Prim, Kruskal

Question 1: Given the statement: All trees are graphs.

Is the statement true or false? Explain your answer.

Answer:
This statement is true because trees are simply a visual representation of the traversal of several nodes. if I have a tree:

```
    100
   /   \
  20   10
```

This can be expressed via

	100	10	20
100	0	0	0
20	1	0	0
10	1	0	0

Question 2: Given the statement: All graphs are trees.

Is the statement true or false? Explain your answer.

Answer: Correct answers will discuss that this is false; a tree must have n Vertices and *n-1* edges, but a graph can have cycles, which therefore means more edges than vertices. A Graph can also have isolated vertices, which a tree cannot have.

Question 3: Given a Heap with exactly 7 elements, and a Binary Search Tree (BST) with at exactly 7 elements, discuss at least three similarities and differences between the layout and data placement in the trees. In your discussion, consider comparing and contrasting the following: what possible shapes could they be? What are their possible depths? How easy is it to find a value on either tree?

Answer: A heap with 7 vertices is a Perfect Tree, while a BST with 7 vertices can take any shape

A heap with 7 vertices can only have a max depth of 2, while a 7 vertex BST in its worst layout could have a depth of 6

In a heap, we know the max value is always at the top. In a BST, the max value is always found at the bottom of the rightmost branch

A value in a 7 element BST can be found in at most O(n lg n) searches. The only way to find a particular value in a heap is to use a Breadth First or Depth First Search.

Question 4: Given a set of vertices V. and a set of edges E that make up a graph G, and the sizes of the sets of vertices and edges are denoted by |V| and |E| respectively, which of the following is true?

A. If G is a Tree, then |E| + 1 = |V|

B. If G is a Tree, then |V| + 1 = |E|

C. If G is a Tree, then |E|+ 1 > |V|

D. If G is a Tree, then |V| + 1 > |E|

Question 5: Given a connected Graph that has at least one cycle, what of the following is true about the MCST (Minimal Cost Spanning Tree) that is generated by this Graph?

A. The MCST has more Vertices than the Graph

B. The MCST has more Edges than the Graph

C. The MCST the same number of Edges as the Graph

D. The MCST the same number of Vertices as the Graph

Question 6: Which of the following is true about MCSTs (Minimal Cost Spanning Trees)

A. They can always be isomorphed (redrawn) as a single line of nodes with no children
B. There is only one path that leads from one vertex to another
C. They must be Binary Trees
D. The sum of their vertex values will be minimal

Question 7: Which of the following is true about Prim's Algorithm and Kruskal's Algorithm:

A. Kruskal always checks for cycles being produces, but Prim does not
B. Both algorithms produce a tree with the sum of the edge weights being the lowest in value
C. They will always produce the same Minimal Cost Spanning Tree
D. Prim always checks for cycles being produces, but Kruskal does not