1. What is the difference in complexities for an Unsorted list vs a Sorted Array?

   Unsorted List: O(n) for get, set, contains and del
   Sorted Array: O(log n) for get and contains, O(n) for set and del

2. How would we implement an AVL Tree?

   To implement an AVL Tree, each time an item is added or removed from the tree, we rebalance the tree.
   Note that only ancestors of the (added/deleted/moved) node can become unbalanced, so we only need to search up the tree.

3. What is a Tree?

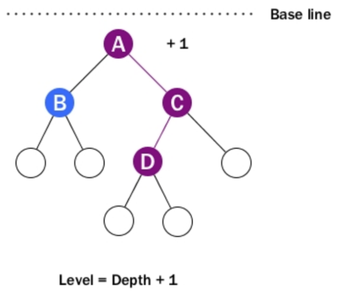
   A tree is a **connected undirected simple graph with no cycles.**

4. How do Hash Tables work?

   Hash tables **locate** {key,value} pairs based on a *hash* value for the key(integer created using the hash() function) and a compression function, which locates the hash value in a **restricted length list.**

5. Define the Level of a node in a binary search tree.

   The Level of a node is defined by 1 + the number of connections between the node and the root.

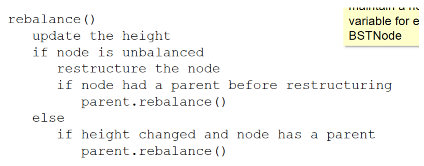


6. What are the properties of an AVL Tree?

   The height of an AVL Tree for n items is O(log n)
   search is O(log n)
   add is O(log n)
   delete is O(log n)
   find_min, find_max is O(log n)
   traversing all nodes is O(n)

7. Give pseudocode descriptions for adding a node to an AVL Tree.

   #After each addition



```
rebalance()
   update the height
   if node is unbalanced
      restructure the node
      if node had a parent before restructuring
         parent.rebalance()
   else
      if height changed and node has a parent
         parent.rebalance()
```

8. For an arbitrary BST, what is the worst case complexity of finding an element in the tree?

   To find the node is O(height of tree), and the worst case is when it is a **leaf on the longest branch.**

9. What is a Binary Search Tree?

   A Binary Search Tree is a rooted tree where
   - Every node has at most two children and the children of a node are ordered from left to right.
   - All left descendants have values lower than their parent's node value
   - All right descendants have values greater than their parents node.

10. What is the expected search complexity for hash tables? WHat is the the worst case search complexity?

    The **expected** search complexity for hash tables is O(1).
    The **worst** search complexity may be O(n/N), where N is the size of the list.

11. Explain the log linear constant function, and give an example of its implementation. (Big O)

    f(n) = n log n
    Typically, the algorithm is doing something like binary search, repeated once for each element in the list.

12. Define the depth of a binary search tree

    The depth of a binary tree is the length of the longest path from the root node to a leaf node.

13. What is a Queue? How can we make a queue more efficient?

    Queues are first-in, first-out (FIFO), meaning if we want to take an item we take it from the front, and if we want to add an item, we add it onto the back.
    We can make a queue more efficient by introducing a **head** and a **tail** variable which point to the beginning and end of a queue respectively.
    These are updated after each use of enqueue(item) and dequeue().
    When we use dequeue(), we assign None to the element that the head is pointing to, reducing complexity from 0(n) to 0(1).
    When the end of the list is reached, instead of extending the list, we place new items at the front of the queue.

14. What is a leaf?

    A leaf is a node with no children.

15. What is an internal node?

    An internal node is one that is **not** the root node or a leaf.

16. What is an AVL Tree?

    An AVL Tree is a binary search tree in which no node is unbalanced.

17. What is a Map? What is the complexity requirement for building a full map?

    A map is a storage and look-up structure maintaing {key, value} pairs(uses dict data type). **Each key is unique.** Values can be retrived from this structure by specifying the corresponding key.
    The complexity requirement for building a full map is O(n^2)

18.	How does a hash tables handle collisions?	A collision may occur when different objects get the same location(i.e. two ormore values point to the same key). The **bucket array** places colliding objects into a ('bucket') list.
19.	How does the compression function work with regard to the hash function and hash tables?	The hash function tries to avoid collisions as much as possible by giving distinct items distinct hashes. However, with a fixed size output, collisions are inevitable. The compression function **tries to distribute the hash values as evenly as possible across the range.** We used modular arithemtic and Linear Probing i.e. hash(k), then (hash(k) + 1 % N), then (hash(k) + 2 % N)
20.	What is a Map ADT? (or Dictionary) What are its standard operations?	A Map is a **storage/look-up** structure containing (key,value) pairs, with each key appearing on the map **at most once.** getitem(key) = return the value with given key, or None setitem(key) = assign value to element with key contains(key) = return True if map has an element with key delitem(key) = remove element with key, return it's value, or None length() = return the number of elements in the map is_empty() = return True if no items in the map

21.	What are the classes and functions of the Doubly Linked List ADT?	class DLLNode:
		
		class DLinkedList:
		add_between(e, before, after) = add an item between before and after.
		add_first(e) = add an item to the start of the list.
		get_first() = return the item at the start of the list.
		remove_node(node) = remove an item from the list.
		remove_first() = remove the item at the start of the list.

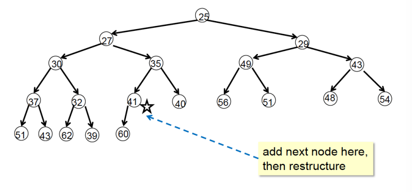
22.	What is the size of a tree?	The total number of nodes.
23.	Explain what a stack is then list and explain its functions.	Stacks are last-in, first-out. Functions: push() = place element on top pop() = get 1st element from top and remove top() = report top element (dosen't remove) length() = report how many elements in stack is_empty()= report if stack is empty

24.	What are the classes and standard operations of the queue ADT?	class Queue:
		enqueue() = place an element onto the end of the queue
		self._body.append(element)
		dequeue() = take the fist element off the front of the queue
		return None if self._length() == 0 else self._body.pop()
		front() = report the front element from the queue
		return None if self.length() == 0 else self._body[-1]
		first() = return the first element of the queue
		return None if self.length() == 0 else self._body[0]
		length() = report how many elements are in the queue
		return len(self._body)
		is_empty = report whether or not the queue is empty
		return len(self._body) == 0

25.	What is a rooted tree?	A rooted tree is a tree in which one of the vertices is designated the root, and all edges are then directed away from that root.
26.	What is a Full Binary Tree?	A Full Binary tree is a binary tree in which **all interior nodes have two children** and **all leaves have the same height.**

27.	Give pseudocode descriptions for the removing a node from an AVL Tree.	#After each removal
		`if removed node was leaf`
		`   leaf.parent.rebalance()`
		`else if removed node was semileaf`
		`   semileaf.parent.rebalance()`
		`else   #internal, moved 'biggest' up, removed biggest`
		`   node = the original parent of biggest`
		`   node.rebalance()`

28.	Explain the Logarithmic standard function. (Big O)	f(n) = $\log_b n$, for some fixed constant value b The log of a number is the power to which the base must be raised to give the number. $\log_b n = x$ if and only if $b^x = n$
29.	Explain the Linear standard function. (Big O)	f(n) = n Typically, this comes from an algorithm with a single loop that iterates over each element of an input list.

30.	Explain the Cubic standard function. (Big O)	f(n) = n^3 Typically this comes from algorithms with a doubly nested loop where we are iterating over the same structure. We can go higher, to any arbitrary degree.
31.	Describe a Binary Search Tree, and give a clear definition of the properties that define it.	A Binary Search Tree is a representation of an ordered sequence of elements where: - All left descendants of a node have values less than the node's value. - All right descendants of a node have values greater than the node's value. - Every node has at most 2 children and the children of a node are ordered from left to right.
32.	What is a Set?	A set is a collection of elements, with no duplicates, and with no particular order among the elements. e.g. {b,c,d,f,g}
33.	What is the difference between a queue and a priority queue?	In a prirority queue, the element with highest priority is removed next.
34.	What is a semi-leaf?	A semi-leaf is a node with only one child.
35.	What is the height of a tree?	The height of a tree is the **total number of links** from the root node to the furthest leaf.
36.	What is the order of the 7 standard functions used to describe the worst-case upper bounds? (Big-O Notation)	1. Constant: f(n) = c, for some fixed constant value c 2. Logarithmic: f(n) = $\log_b n$, for some fixed constant value b 3. Linear: f(n) = n 4. Log Linear: f(n) = n log n 5. Quadratic: f(n) = n^2 6. Cubic: f(n) = n^3 7. Exponetial: f(n) = c^n, for some constant c
37.	Explain the Quadratic standard function. (Big O)	f(n) = n^2 Typically this comes from algorithms with a nested loop where we are iterating over the same structure.
38.	Define an AVL tree. How would we implement it?	An AVL Tree(or a balanced tree) is a Binary Search Tree in which no node is unbalanced. To implement an AVL Tree, each time an item is added or removed from the tree, we rebalance the tree.
39.	Explain the constant standard function. (Big O)	f(n) = c, for some constant fixed value c The running time of the function is independent of the size of the input e.g. a function which returns the value in the first position in a list. Note: We say such a function is O(1)
40.	What is a Binary Heap?	A Binary Tree where: - Every node has a lower key than its children - every level(except maybe the last) is complete - The lowest level is always filled from the left.  25 27 29 30 35 49 43 37 32 41 40 56 51 48 54 51 43 62 39 60 add next node here, then restructure
41.	Explain the Exponential standard function. (Big O)	f(n) = c^n , for some constant c Typically this comes from algorithms with a loop where the number of operations increases by a factor of c each time round the loop. Considered inefficient.
42.	Define the height of a node in a binary search tree.	**The height of a node is the number of edges on the longest downward path between that node and a leaf.** height(node) = 0 if node is a leaf height(node) = 1 + max(height(left), height(right))
43.	For an arbitrary BST, what is the worst case complexity of removing an element in the tree?	To remove it is O(height of tree). Find the biggest item less than it, which is O(height of node). Constant number of operations to replace.
44.	What are the complexities for AVL Tree Properties? (adding, searching, deleting etc.)	Height of an AVL tree for n items is O(log n) Search is O(log n) Add is O(log n) Delete is O(log n) find_min, find_max is O(log n) traversing all nodes is O(n)
45.	How does the Python hash function work? (Hash Tables)	The hash() function converts any **immutable** data into an integer. **This does not work for floats!**
46.	What characterises a Binary Tree?	Every node has at most 2 children and the children of a node are ordered from left to right.

47. What is a Doubly Linked List?	A doubly-linked list is a list structure where **every element has a link to both the previous and next element,** and a **head element indexing the start of the list** and a **tail element indexingthe end of the list.** First element comes **after** the head. Last element comes **before** the tail. Size of the list is stored as a variable and updated after each addition and removal.