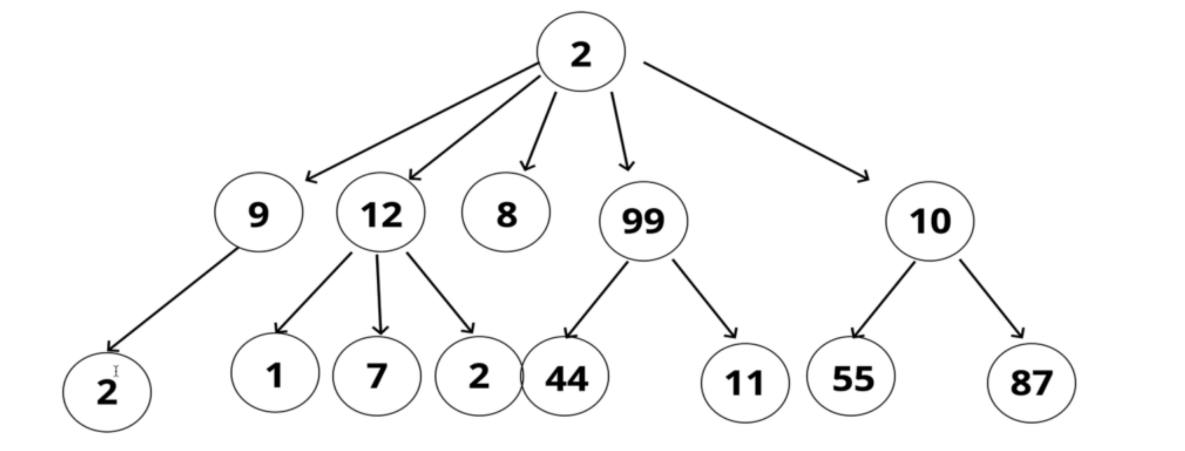
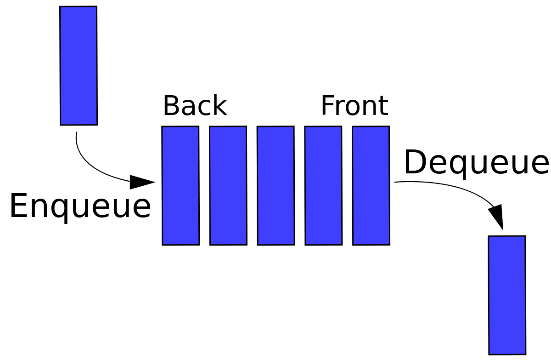
Queue Implementation

1. **class** Node:
2. **def** \_\_init\_\_(self, data):
3. self.data = data
4. self.next = None
6. **class** Queue:
7. **def** \_\_init\_\_(self):
8. self.first = None
9. self.last = None
10. self.length = 0
12. **def** enqueue(self, data):
13. node = Node(data)
14. **if** **not** self.first:
15. self.first = node
16. self.last = node
17. **else**:
18. self.last.next = node
19. self.last = node
20. self.length += 1
21. **return** self.length
23. **def** dequeue(self):
24. **if** **not** self.first:
25. **return** None
26. current = self.first
27. **if** self.first == self.last:
28. self.last = None
29. self.first = self.first.next
30. self.length -= 1
31. **return** current.data



* Abides by **FIFO** (first in first out), **first element** added to the stack will be the **first removed** from the stack.
* Can use a list for a stack. list.insert(0, data) to add onto the beginning and li.pop() to remove the item.
* Analogy: waiting in line to get coffee; gaming waitlist



### Queue Big O

* Insertion = O(1)
* Deletion = O(1)
* Searching = O(n)
* Access = O(n)