Midterm Review

1 - Quickest contains

- Set is fastest to find is an element exists
- Collection is an abstract class

2 - Quickest find

- SortedSet, if you store a string ({lastname},{firstname})

5 - ArrayList

- get() is O(1) in ArrayList
- remove() in O(1 + n-i)

6 - Linked List

7 - ArrayList vs LinkedList

- Same because from front of array

8 - ArrayList vs LinkedList

- ArrayList is faster at random locations

9 - ArrayList vs LinkedList

- same at the back

10 - ArrayList vs LinkedList

- ArrayList has to shift all elements
- LinkedList is faster

11 - ArrayList vs LinkedList

- same since in the middle?

12 - ArrayList vs LinkedList

- LinkedList is faster since we have pointer to middle
- Don't need to iterate to get to middle

13 - ArrayStack

- simple math

14 - ArrayStack

- a.length/2 - a.length/3 = a.length/6
- 3n < a.length
- n < a.length/3
- thus, (a.length/6) / (a.length/3) = n / 2

15 - ArrayStack

- memorize At most 2m elements are copied by grow() and shrink()

16 - ArrayDequq

- O(1+min{i,n−i})

17 - Binary

18 - Binary

- x%m is same as x&(m-1)

19 - DualArrayDeque

- front arraydeque is reversed

21 - RootishArrayStack

- 10 * 11/2

22 - RootishArrayStack

- get(3)[4];

23 - SinglyLinkedList (SLL) as a Queue

- enqueue is add tail
- dequeue is remove head

24 - SinglyLinkedList (SLL) as a Stack

- push is add head
- pop is remove head

25 - SLL

- in O(1 + min(i, n(n − i − 1)) time
- This catches the condition when head == tail
- i.e. if i = n-1 => n(n − i − 1) becomes 0

26 - DLL

- dummy node is used to point to front and back
- dummy.next is front
- dummy.prev is back

28 - DLL

- add() in O(1 + min(i, n−i)
- remove() in O(1 + min(i, n−i))

29 - HashTable [HashMap]

- O(n/m)
- Elements are evenly spaced out