Complexity Questions by Kate Barnes

` 1. a. A queue is a FIFO (first in first out) data structure, so it adds to the end of the list and takes off of the beginning.

b. A stack is a FILO (first in last out) data structure, so it adds and removes from the end of the list.

2. An array will be good if the task requires us to access items at specific intervals frequently (like the fetch method) because the labeled indexes in arrays make this method have a constant time complexity function. Arrays are also good if the task requires a list of fixed length. A linked list on the other hand is good for a task that requires a list of shifting sizes or a lot of removing items from various places on the list. For an array, removing items in the middle of the list requires shifting all of the following cells back one and for a linked list you just have to find the item and remove the pointers.

1. a. **Constant O(1)** time complexity, because we just reference the last index on the list and add an element to the following index.

b. **Linear O(n)** time complexity at worst, the further from the end the element is the longer the operation takes because we have to shift each cell after the removed cell back one index.

c. **Constant O(1),** because we just have to reference the index and return the value.

1. a. **Constant O(1)** as long as we have a tail pointer for the list because we just access the tail pointer and link the new element to this cell.

b. **Linear O(n),** because we have to search through the list one by one until we arrive at the right index, and once there we just have to remove the pointers to this cell.

c. **Linear O(n)** because we have to search through the list one by one until we find the index we’re looking for, and then we just return this index.

1. a. **Linear O(n)** because you have to search through each element one by one to find the value.

b. Same time complexity for the same reason.

c. The lower bound is **logarithmic O(logn)** because each time you make a move on the tree you eliminate half of the items you have to search through. The upper bound is **linear O(n)** with the linearity corresponding to the height of the tree not the number of nodes, because in the worst case you have to go all the way to the bottom of the tree.

d. Doesn’t change things.

6. Yes because a dictionary backed by a binary search tree will have logarithmic time complexity **O(logn)** for fetching elements instead of linear, because each time you make a move on the tree you eliminate half of the choices. VS. with a list you have to search through every item in a linear fashion to find anything on the list.

The Assignment Questions:

1. Queues and stacks can be thought of as specializations of lists that restrict which elements can be accessed.  
1a. What are the restrictions for a queue?

1b. What are the restrictions for a stack?

2. We have looked at lists backed by arrays and links in this class. Under what circumstances might we prefer to use a list backed by links rather than an array? (Your argument should include asymptotic complexity).

3. Give the asymptotic complexity for the following operations on an array backed list. Also provide a brief explanation for why the asymptotic complexity is correct.

3a. Appending a new value to the end of the list.

3b. Removing a value from the middel of the list.

3c. Fetching a value by list index.

4. Give the asymptotic complexity for the following operations on a doubly linked list. Also provide a brief explanation for why the asymptotic complexity is correct.

4a. Appending a new value to the end of the list.

4b. Removing the value last fetched from the list.

4c. Fetching a value by list index.

5. One of the operations we might like a data structure to support is an operation to check if the data structure already contains a particular value.

5a. Given an unsorted populated array list and a value, what is the time complexity to determine if the value is in the list? Please explain your answer.

5b. Is the time complexity different for a linked list? Please explain your answer.

5c. Given a populated binary search tree, what is the time complexity to determine if the value is in the tree? Please give upper and lower bound with an explanation of your answer.

5d. If the binary search tree is guaranteed to be complete, does the upper bound change? Please explain your answer.

6. A dictionary uses arbitrary keys retrieve values from the data structure. We might implement a dictionary using a list, but would have O(n) time complexity for retrieval. Since we expect retrieval to occur more frequently than insertion, a list seems like a poor choice. Could we get better performance implementing a dictionary using a binary search tree? Explain your answer.