MOHIT SHARMA (2019CS10372)

A1.3.1: n operations performed on DLL.

1) Worst case time complexity of Insert: O(1)
In insert, a new node of the given data is made which takes constant time and then 4 pointer values are initialised which is of constant time.
So, overall time complexity is O(1).

2) • Worst case time complexity of ~~Del~~ getFirst : O(n)
In worst case, DLL will have n nodes after n operations.
If getFirst is called from tail node, then the while loop runs exactly n times. So, T(n) = n = O(n)
Other helper functions used (isEmpty(), isHead() are of constant time complexity).

3) Worst case time complexity of Delete : O(n)
In delete, getFirst function is called which ~~may~~ takes n units of operation in worst case.
Then the while loop would run over all the nodes in DLL (n), in worst case when the last node (before tail node) matches the node to be deleted, so loop runs n times.
The deletion then takes 8 pointer initialization in worst case which is O(1)
getNext function ~~shown~~ used is taking O(1) time (shown below).
So, in worst case, T(n) = n + n + O(1) = O(n).

4) Worst case time complexity of getNext : O(1)
Helper functions used (isEmpty, isTail) do constant number of matches so they have O(1) time complexity.
In getNext, these constant time taking functions are used 3 times in worst case.
So, T(n) = O(1)

Mohit Sharma (2019CS10372).

5) Worst case time complexity of Find : $O(n)$.

The getFirst function is used, it takes n unit of time operation in worst case.

Then the while loop is run n times over all DLL nodes in worst case when no match is found.

So, $T(n) = n + n + O(1) = O(n)$.

6) Worst case time complexity of Sanity : $O(n)$.

In sanity function, the cycleExists function takes $O(n)$ time. Standard slow and fast pointer technique is used in which slow pointer moves a max of $2n$ times.

Other than this, the DLL is traversed in forward and backward direction, taking a total of $n+3$ iterations (head and tail nodes are included).

So, $T(n) = 2n + n + 3 + O(1) = O(n)$.

Mohit Sharma (2019CS10372)

A.1.3.2 : n operations performed of Allocate / Free.

1) Worst case time complexity of Allocate : O(n)

In worst case freeBlk can have $\lfloor n/2 \rfloor + 1$ nodes.
So number of nodes is O(n) in freeBlk. DLL.

In worst case, in Allocate function, a find & delete function is used on freeBlk and two inserts one on allocBlk & one on freeBlk. The calls are made from head so getFirst in find & delete will take O(1) time.

So, $T(n) = \underbrace{\lfloor n/2 \rfloor + 1}_{\text{Find}} + \underbrace{\lfloor n/2 \rfloor + 1}_{\text{Delete}} + \underbrace{O(1)}_{\text{Insertions}}$

$T(n) = O(n)$.

2) Worst case time complexity of Free : O(n).

In worst case, allocBlk can have n nodes.
So find and delete on allocBlk will take n operations.
As call is made from head node, so find and delete, getFirst used takes O(1) time.

So, $T(n) = \underbrace{n}_{\text{Find}} + \underbrace{n}_{\text{Delete}} + \underbrace{O(1)}_{\text{Insertion}}$

$T(n) = O(n)$.