A 1.3

2019CS50768 ABHINAV SINGHAL

1) i) **Insert**: In insert operation on the Doubly linked list, we are just changing a few pointers of the current node on which insert method is called and the next to current node to insert the new node in between these 2 nodes. Hence, Insert Method is O(1), i.e, Constant time complexity.

ii) **Delete**: First we need to search for the correct node that is to be deleted. We will have to traverse all of the nodes in the DLL to search the Node. This will take O(n) time, where n is the number of nodes in the DLL.
If we find the node to be deleted, we just have to change a few pointers of its Previous and next nodes, which has O(1) Time Complexity.

Hence, Deletion is O(n) worst case Time Complexity.

iii) **Find**: To find a node with a given key K, or a node with key >= K (depends on 'exact' input), we will again have to traverse the whole DLL in the worst case and check if we find the correct node. Hence, Find method has O(n) time complexity, where n = number of nodes in DLL.

Scanned with CamScanner

A1.3

ABHINAV SINGHAL 2019CS50768

I) iv) **getFirst**: To get the first Node of the DLL, we will have to go backwards from the Node on which the method has been called till we reach the Node just ahead of the Head sentinel node. In the worst case, we may have to traverse the whole DLL Backwards, Hence, worst case T.C is $O(n)$, where, n = number of Nodes in DLL.

v) **getNext**: We just have to give the next Node of the Node on which getNext is called, or null if Node is the tail Sentinel node or one behind the tail sentinel. This takes constant time. Here, T.C is $O(1)$.

vi) **Sanity**: For various tests in the Sanity method, we are repeatedly performing a while loop, and each takes $O(n)$ time, because we have to traverse the whole DLL and visit all nodes atleast once. Hence, for all the tests put together, the Sanity function takes $O(n)$ time for verifying all the Tests.

Scanned with CamScanner

A1.3

2019CS50768 ABHINAV SINGHAL

II i) **Allocate**: In the worst case of Allocate, first we spend O(n) time in finding a freeblock of size greater than or equal to required size. Then, once found, we will have to split in the worst case and perform insert and delete operations in the two DLL → freeblk and AllocBlk, which takes O(n) time.

Here, Allocate method also takes O(n) Time.

ii) **Free**: In the worst case, we may have to traverse the whole AllocBlk DLL to find the Node to be deleted, which takes O(n) time.

Then, when found, we need to delete it from the AllocBlk (O(n)) and insert in freeBlk (O(1)). This also takes O(n) net time.

In total, Free method also takes O(n) time.

Scanned with CamScanner