|  |  |  |  |
| --- | --- | --- | --- |
|  | Time Complexity | Space Complexity | Understanding  (space complexity in bold) |
| def append() | O(n) | O(1) | This has O(n) of time complexity in case if we have to go for last element, we have to traverse the entire list.  **It only requires constant amount of extra memory to store new node.** |
| Def prepend() | O(1) | **O(1)** | It takes constant amount of time to add new node in the front of the list regardless of size of linked list.  **It only requires a constant amount of memory to create a new node and store its value.** |
| Def find() | O(n) | **O(1)** | This has O(n) of time complexity in case if we have to iterate through all the elements to find the key element.  **It only requires constant amount of space to store the current node during the iteration.** |
| Def delete() | O(n) | **O(1)** | The method needs to traverse the entire linked list in order to find the element to be deleted.  **It only uses a few variables to keep track of the current and previous nodes during the traversal.** |
| Def delete\_front() | O(1) | O(1) | It only involves a single process and takes constant time.  **Use few variables to store the current and previous nodes.** |
| Def delete\_last() | O(n) | O(1) | method needs to traverse the entire linked list to find the second-to-last node, which requires O(n) time.  **Use few variables to store the current and previous nodes.** |
| Def build\_slist\_from\_list() | O(n) | O(1) | It uses for loop and iterates through all the elements  **Only use a constant amount of additional memory to store variables** |
| Is\_empty() | O(1) | O(1) | It checks the value of the first instance variable, which takes constant time.  **Only use a constant amount of additional memory to store variables.** |
| Reverse() | O(n) | O(1) | This uses while loop to iterates through all the elements and reverse their order by next pointer.  **Only use a constant amount of additional memory to store variables.** |
| find\_mid\_point() | O(n) | O(1) | It iterates through all the elements, count the number of elements in order to give midpoint.  **Takes constant space to store variables**. |
| getI | O(n) | O(1) | It uses a for loop that iterates through the specified number of elements in order to get the element at the specified index.  **Takes constant space to store variables**. |
| \_\_len\_\_ | O(n) | O(1) | Method iterates through each node in the linked list, incrementing the count by 1 each time.  **Additional space used is for the count and final variables, which do not grow with the size of the linked list.** |
| \_\_str\_\_ | O(n) | O(1) | Method iterates through each node in the linked list and appends the value of each node to a list, which is later joined to form a string.  **Additional space used is for the count and final variables, which do not grow with the size of the linked list.** |

As it majorly takes time complexity as O(n) and it does not grow with the size of the linked list, it takes constant space of 0(1).