Rahul Kadam SUID: 704734820
rkadam@syr.edu

CSE 674 Advanced Data Structures and Algorithms Spring 2017

Assignment No. 1:

Markov Text Generator

Doubly Linked List Markov:

-> "add method -> O(n)"

Algorithm:

1. Search for Node t1.

2. If Node t1 found:

    - add t2 to t1's vector.

  else: (If found)

    - create a new node and add it to tail of doubly linked list

    - add t2 to this newly created node's vector

O(n) is because for searching the Node t1 we traverse the doubly linked list and worst case scenario could be that the Node t1 is at the tail end of the doubly linked list.

-> "generate method -> O(n)"

Algorithm:

1. Get the length of the list.

2. Generate x -> Random Number in between 0 and size of the list.

3. Traverse through the linked list x times to get a Random Node A.

4. Store info of Node A in final_string.

5. Get the Random string B inside the vector of the Node A.

6. Append this string into the final_string.

7. Traverse again through the linked list to find the node with info as B.

8. Loop on steps 4 to 7 until the Markov List contains certain number of nodes. eg: 20 times

O(n) is because for searching the Node t1 we traverse the doubly linked list and worst case scenario could be that the Node t1 is at the tail end of the doubly linked list. Similarly, for searching the Node t2 we traverse the doubly linked list and worst case scenario could be that the Node t1 is at the tail end of the doubly linked list. Thus, O(2n) ~ O(n).

Rahul Kadam SUID: 704734820
rkadam@syr.edu

Multi-Linked List Markov:

"add method -> O(n)"

Algorithm:

1. Search for Node t1.

2. If Node t1 found:

    - add t2 to t1's vector.

  else: (If found)

    - create a new node and add it to tail of doubly linked list

    - add t2 to this newly created node's vector

Complexity is O(n) because for searching the Node t1 we traverse the doubly linked list and worst case scenario could be that the Node t1 is at the tail end of the doubly linked list.

"generate method -> O(n)"

Algorithm:

1. Get the length of the list.

2. Generate x -> Random Number in between 0 and size of the list.

3. Traverse through the linked list x times to get a Random Node A.

4. Store info of Node A in final_string.

5. Get a Random Node B's address inside the vector of the Node A.

6. Append this string into the final_string.

7. Directly go to the Node B (no traversing list here again).

8. Loop on steps 4 to 7 until the Markov List contains certain number of nodes. eg: 20 times

Complexity is O(n) , because for searching the Node t1 we traverse the doubly linked list and worst case scenario could be that the Node t1 is at the tail end of the doubly linked list. But, for searching the Node t2 we don't traverse the doubly linked list as we are storing addresses of Nodes in vector and not strings we did in Doubly Linked List.

Thus, O(n).

Rahul Kadam SUID: 704734820
rkadam@syr.edu

Skiplist Markov:

“add method -> O(log n)”

“generate method -> O(log n)”

The worst case search time for a sorted linked list is O(n) as we can only linearly traverse the list and cannot skip nodes while searching. In Skip List, we create multiple layers so that we can skip some nodes.

Eg: The upper layers works as an “express lane” which connects only main outer stations, and the lower layer works as a “normal lane” which connects every station.

Because of these Layers the complexity of add(), remove() and find() are O(log n).

References:

1. For Vector Implementation -> Array List Implementation from Professor’s notes.

2. For Doubly Linked List -> IDLL implementation from Professor’s notes.

3. For Skip List -> Open Data Structures by Pat Morrin.