CS 225 - Lecture 4

Scribe : Harsha Srimath Tirumala

1 Learning Goals

↪ Review fundamentals of Linked lists

↪ Implement insertatFront and Index operations

↪ Pointers vs reference-to-pointers

2 Linked Lists

↪ Access in Linked lists is **one-way**. (A different implementation of linked lists allows two-way access : *Doubly Linked lists*).

↪ Data is accessed via reference and links via pointers.

↪ Unless otherwise specified, we will assume our linked lists have n nodes.

3 insertAtFront(data)

1. Create a new ListNode	ListNode* *tmp* = new ListNode(**data**)
2. Set its next to head	*tmp*→next = head_
3. Update head to point to *tmp*	head_ = *tmp*

↪ Runtime : Create-ListNode ($O(1)$) + Set-next ($O(1)$) + Update-Head ($O(1)$) = $O(1)$

4 _index()

↪ Conceptual - Must return the link to the required index to establish existence of data (being searched). Very helpful if link leading to said data can be manipulated (eg : for insert/delete at that location)

↪ Key Idea - Return type **ListNode* &** allows modification of *link to index* as well as *data in next ListNode*.

↪ Note - Return type **ListNode*** cannot modify link to index.

↪ Runtime : $O(n)$ as it involves scanning from head through index one node at a time.

```
template <typename T>
typename List<T> :: ListNode *& List<T> :: _index(unsigned index) {
 if (index == 0) return _head;
 else{
     ListNode* curr = _head;
     for (unsigned int i = 0; i < index - 1; i++) {
        curr = curr -> next;
     }
     return curr->next;
     }
 }
```