### Programming Guide for Advanced List Structures

#### 1. Skip Lists

**Overview:** A Skip List is a data structure that allows fast search within an ordered sequence of elements. It does this by maintaining multiple layers of pointers, skipping several elements in a single step.

**Key Operations:** - **Insertion:** Place the element in the appropriate position while maintaining the skip list property. Randomly determine the level of the new node. - **Search:** Start from the top level and move horizontally as far as possible before moving down a level, until the element is found or not. - **Deletion:** Similar to insertion but involves removing nodes and updating pointers.

**Pseudo Code for Skip List:**

CLASS SkipListNode:  
 FUNCTION \_\_init\_\_(key, level):  
 Initialize key and forward pointers for each level  
  
CLASS SkipList:  
 FUNCTION \_\_init\_\_(max\_level, probability):  
 Initialize max\_level and probability  
 Create a header node with max\_level  
  
 FUNCTION insert(key):  
 Randomly determine the level for the node  
 Insert the node by updating forward pointers  
  
 FUNCTION search(key):  
 Start from the highest level and move horizontally and down  
 Return True if the element is found  
  
 FUNCTION delete(key):  
 Locate the node and remove it by updating pointers

#### 2. Self-Adjusting Lists

**Overview:** Self-Adjusting Lists rearrange their elements based on access frequency. The most commonly accessed elements are moved towards the front of the list.

**Key Operations:** - **Access:** Move the accessed element to the front of the list. - **Insertion:** Add new elements to the front of the list. - **Deletion:** Remove elements as required.

**Pseudo Code for Self-Adjusting List:**

CLASS SelfAdjustingList:  
 FUNCTION \_\_init\_\_():  
 Initialize an empty list  
  
 FUNCTION access(key):  
 IF key is in the list:  
 Move key to the front  
 ELSE:  
 Return False  
  
 FUNCTION insert(key):  
 Add key to the front of the list  
  
 FUNCTION delete(key):  
 Remove key from the list if it exists

#### Applications

* **Skip Lists:** Used in databases and file systems for efficient indexing and search operations. They provide a balance between the complexity of balanced trees and the simplicity of linked lists.
* **Self-Adjusting Lists:** Useful in caching mechanisms where frequently accessed items need to be retrieved quickly. Also used in some memory allocation strategies.