Generic Programming (UE14CS337)

Mini Project

Generic Skip List Implementation in C++

Sushrith S. Arkal
01FB14ECS262

Anush S Kumar
01FB14ECS037

Tejas S Kasetty
01FB14ECS267

Abstract

The skip list data structure is an ordered linear data structure (linked list container) that also allows fast search (better than O(N), where N is the number of elements in the list). This is made possible by allowing the search algorithm to "skip" intermediate nodes until it reaches the node that it is looking for. The average case time complexity can be shown to be O(logN), and even more impressive is that it can be shown that, "with high probability", the time complexity is O(logN).

This is because a skip list data structure maintains a linked hierarchy of subsequences, with each subsequences skipping over fewer elements than the previous one.

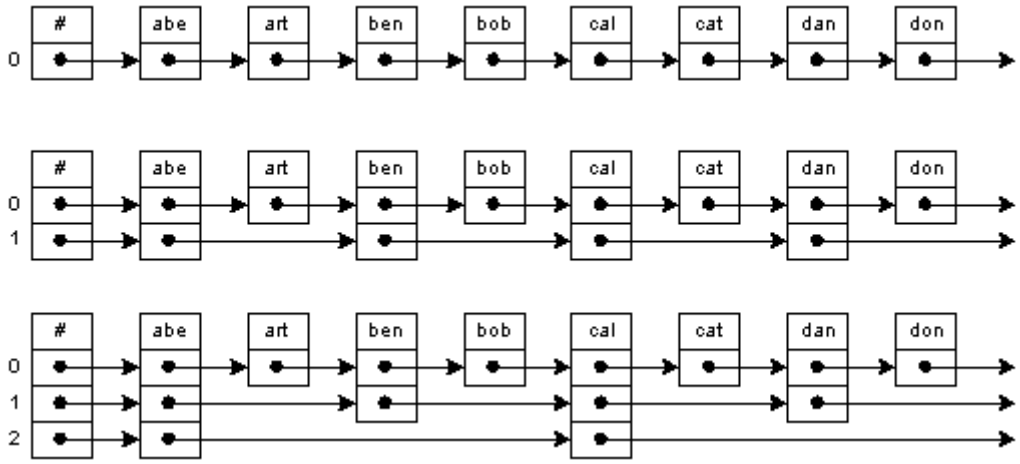

Figure I : Vanilla Linked List, and Skip Lists with 2 and 3 levels

Thus, in the search algorithm, taking into account the ordered nature of the data structure, we can traverse the top layer lists, which is sparse, allowing us to skip nodes in the base list (the list with all elements), until we find the key, or a key that is ordered above the required key, at which point we can drop down a layer and continue the search until the key is found or we reach the base layer, at which point we can conclude that the key is not present in the list.

In our project, we plan to implement the skip list data structure in a generic manner, so that is will work with pre-existing algorithms present as part of the Standard Template Library of C++. We will implement iterators to this effect to support majority of these functions.

As of now, our plan is to implement a list that allows duplicate elements, although it should be noted that, considering the advantage that skip list provides us, implementing a set data structure is also a real probability, and would provide that advantage of possibly cache friendly set iteration (assuming sequential contiguous memory allocation, although in certain cases this assumption is broken, leading to cache unfriendly behaviour) and also, on average and "with high probability", O(logN) insertion operation.