PROJECT REPORT

Abstract:

This project is to implement the SkipList data structure. This data structure allows fast search within an ordered sequence of elements. Fast search is made possible by maintaining a linked hierarchy of subsequences, each skipping over fewer elements. Searching starts in the sparsest subsequence until two consecutive elements have been found, one smaller and one larger than the element searched for. Via the linked hierarchy these two elements link to elements of the next sparsest subsequence where searching is continued until finally we are searching in the full sequence. The elements that are skipped over may be chosen probabilistically or deterministically, with the former being more common.

Problem Statement:

1.Implement SkipList data structure without using any list libraries of Java.

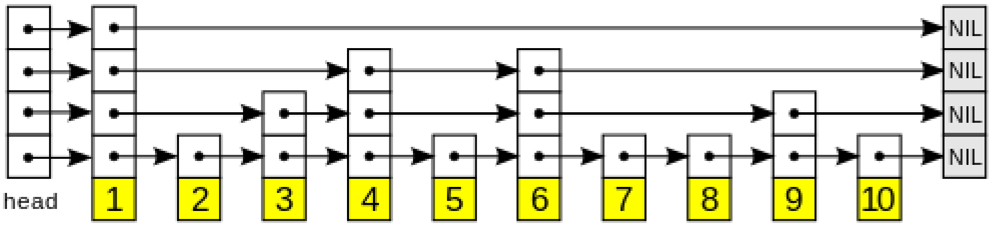
2.Implement following operations which are resent in SkipList interface:

3.Add, Remove, Contains, First, Last, Ceiling, Floor, IsEmpty

4.Compare the SkipList runtime performance with TreeSet performance.

Methodology:

SkipList has a structure for every node. Each node have actual data and for reference next pointer.



A schematic picture of the skip list data structure. Each box with an arrow represents a pointer and a row is a linked list giving a sparse subsequence; the numbered boxes at the bottom represent the ordered data sequence.

1. SkipList is build in layers. The bottom layer is an ordinary ordered liked list. Each higher lane acts as an express lane for list below where an element in layer i appears in layer i+1 with some fixed probability p.

2.To reach a particular node we need to traverse from topmost next pointer of head node.If the data value is less than the value we expect move ahead.

3.But if the value we expect is less than data value which came on the way we will do one level down and then check. Then continue the same process until we find whether element is present. At 0th level we will have the actual data which we are trying to search.

4.If it is not present then head pointer will be connected to tail pointer for all levels.

5.We have developed a helper function find which stores the way we are moving in array prevs[ ].This array is present in a PreviousData structure.

6.The last node in prevs[] array has the actual data which we use to perform operations like Add, Remove, Contains.

Add: We create a new node with some random level. We use prevs[] array to modify the next[] pointer of new node. We call helper function Find to find the location where we can add new element in SkipList. Find function returns null along with reference to one before where we can add new element in the SkipList

Contains: We call the helper function find. If the element is not present it returns null. So them we get to know that element is not present in the list. Otherwise it returns the positions one before the present element position.

Remove: We call helper function to check whether the element is present in the List. If it is present then we will remove from the SkipList as we know the location from where we need to delete from prevs[].

First: Display first element in level 0 of SkipList.

Last: Display last element in level 0 of SkipList

IsEmpty : Check whether SkipList is empty. If it is empty head and tail we will be directly connected to each other at all levels.

Floor: Find the element in SkipList which is just before the specified element at level 0.

Ceiling: Find the element in SkipList which is just after the specified element at level 0.

Development Platform:

Operating System:Windows8

Hardware Specifications: Intel CORE i5 processor,8GB RAM

Development IDE: Eclipse

Comparison Results:

Following table gives the difference between execution time from Skip List and Tree Set.

Input_size	Skip_List	TreeSet
input_50	12	4
input_100	13	12
input_200	20	28
input_1000	69	39
input_10000	298	156
input_50000	566	463
input_100000	894	517
input_1000000	6811	3649

Conclusion:

1.The time taken by Skip List is more as compared to Tree Set. It is almost twice the Tree Set runtime. Actually the Runtime of Skip List must be less when compared with Tree Set.

2.We can improve the performance by removing the redundant code written. By doing proper implementation we can achieve maximum performance for Skip List.

References:

Classnotes

http://en.wikipedia.org/wiki/Skip_list

http://www.drdobbs.com/cpp/skip-lists-in-c/184403579